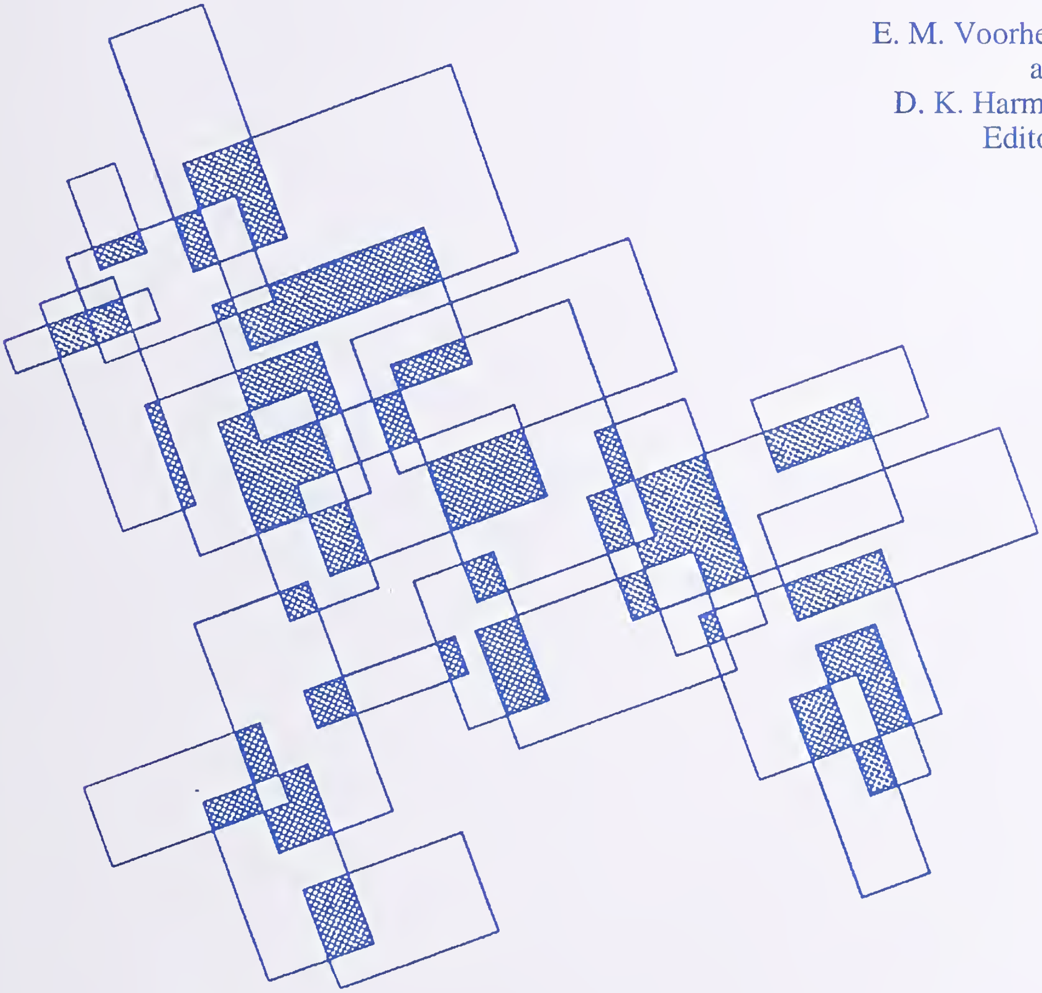
NIST Special Publication 500-250

Information Technology:

The Tenth Text Retrieval Conference, TREC 2001



E. M. Voorhees
and
D. K. Harman
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 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 Services
- Information Services

Advanced Technology Program
- Economic Assessment
- Information Technology and Applications
- Chemistry and Life Sciences
- Materials and Manufacturing Technology
- 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]

Materials Science and Engineering Laboratory
- Theoretical and Computational Materials Science
- Materials Reliability[1]
- Ceramics
- Polymers
- Metallurgy
- NIST Center for Neutron Research

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

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

Manufacturing Engineering Laboratory
- Precision Engineering
- Automated Production Technology
- Intelligent Systems
- Fabrication Technology
- Manufacturing Systems Integration

Building and Fire Research Laboratory
- Applied Economics
- Structures
- Building Materials
- Building Environment
- Fire Safety Engineering
- Fire Science

Information Technology Laboratory
- Mathematical and Computational Sciences[2]
- Advanced Network Technologies
- Computer Security
- Information Access and User Interfaces
- High Performance Systems and Services
- Distributed Computing and Information Services
- Software Diagnostics and Conformance Testing
- Statistical Engineering

[1] At Boulder, CO 80303.

[2] Some elements at Boulder, CO.

NIST Special Publication 500-250

Information Technology:
The Tenth Text Retrieval Conference, TREC 2001

E.M. Voorhees and
D.K. Harman
Editors

*Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20890-8980*

May 2002

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 500-250
Natl. Inst. Stand. Technol.
Spec. Publ. 500-250
968 pages (May 2002)
CODEN: NSPUE2

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 2002

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 2001 edition of the Text REtrieval Conference, TREC 2001, held in Gaithersburg, Maryland, November 13–16, 2001. The conference was co-sponsored by the National Institute of Standards and Technology (NIST), the Defense Advanced Research Projects Agency (DARPA), and the Advanced Research and Development Activity (ARDA). Approximately 175 people attended the conference, including representatives from 21 different countries. The conference was the tenth in an on-going series of workshops to evaluate new technologies for text retrieval and related information-seeking tasks. Eighty-seven 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 cited 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,
Donna Harman
May 6, 2002

TREC 2001 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 2001 PAPERS BY TASK/TRACK

Cross-Language

Filtering

Fudan University

Independent Consultant

IRIT-SIG

Johns Hopkins University

Korea Advanced Institute of Science and Technology (KAIST)

Microsoft Research

NIST

Oracle

SER Technology

Tampere University of Technology

University of Nijmegen

Interactive

CSIRO

NIST

Oregon Health Sciences University

Queens College, CUNY

Question Answering

Video

Web

CSIRO

Electronics and Telecommunications Research Institute

Fondazione Ugo Bordoni

Fudan University

Fujitsu Laboratories, Ltd.

Hummingbird

IBM Almaden Research Center

IBM Labs, Haifa University

Illinois Institute of Technology

Imperial College of Science, Technology and Medicine

IRIT-SIG

Johns Hopkins University

JUSTSYSTEM Corporation

Kasetsart University

Microsoft Research

NCR Corporation

NextTrieve

Abstract

This report constitutes the proceedings of the 2001 edition of the Text REtrieval Conference, TREC 2001, held in Gaithersburg, Maryland, November 13–16, 2001. The conference was co-sponsored by the National Institute of Standards and Technology (NIST), the Defense Advanced Research Projects Agency (DARPA), and the Advanced Research and Development Activity (ARDA). Eighty-seven groups including participants from 21 different countries were represented.

TREC 2001 is the latest in a series of workshops designed to foster research in text retrieval and related technologies. A new video "track" that focused on supporting content-based access to digital video was introduced this year. The other tracks included in TREC 2001 were web retrieval, cross-language retrieval, question answering, interactive retrieval, and filtering.

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 2001 proceedings web site also contains system descriptions that detail the timing and storage requirements of the different runs.

Overview of TREC 2001

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

1 Introduction

The tenth Text REtrieval Conference, TREC 2001, was held at the National Institute of Standards and Technology (NIST) November 13–16, 2001. The conference was co-sponsored by NIST, the Information Technology Office of the Defense Advanced Research Projects Agency (DARPA/ITO), and the US Department of Defense Advanced Research and Development Activity (ARDA).

TREC 2001 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 2001 contained six areas of focus called "tracks". These included the Cross-Language Retrieval Track, the Filtering Track, the Interactive Retrieval Track, the Question Answering Track, the Video Retrieval Track, and the Web Retrieval Track. This was the first year for the video track, which was designed to investigate content-based retrieval of digital video. The other tracks were run in previous TRECs, though the particular tasks performed in some of the tracks changed for TREC 2001.

Table 1 lists the groups that participated in TREC 2001. Eighty-seven groups submitted retrieval results, an increase of approximately 25 % over the previous year. The participating groups come from twenty-one 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 2001

Ajou University
Alicante University
BBN Technologies
Carnegie Mellon U. (3 groups)
Chinese Academy of Sciences
Clairvoyance Corp.
CLIPS-IMAG
CL Research
Conexor Oy
CSIRO
Dublin City University
EC Wise, Inc.
Fondazione Ugo Bordoni
Fudan University
Fujitsu
Harbin Institute of Technology
Hummingbird
IBM-Almaden
IBM-Haifa
IBM-T.J. Watson (3 groups)
Illinois Institute of Technology
Imperial College of Science, Tech. & Medicine
InsightSoft-M
IRIT/SIG
ITC-irst
Johns Hopkins University, APL
Justsystems Corp.
KAIST
Kasetsart University
Katholieke Universiteit Nijmegen
KCSL
Kent Ridge Digital Labs
Korea University
Language Computer Corp.
David Lewis
LIMSI
Microsoft Research China
Microsoft Research Ltd.
MITRE
Moscow Medical Academy
National Taiwan University
New Mexico State University
NexTrieve
NTT Communication Science Labs
Oracle
Oregon Health and Science University
Pohang University of Science and Technology
Queens College, CUNY
RICOH
Rutgers University (2 groups)
SER Technology Deutschland GmbH
Sun Microsystems Labs
Syracuse University
Tampere University of Technology
Tilburg University
University of Twente
TNO-TPD & Universite de Montreal
University of Alberta
University of Amsterdam/ILLC
U. of Amsterdam & CWI & TNO & U. Twente
University of California, Berkeley
University of Glasgow
University of Illinois, Urbana/Champaign
University of Iowa
University of Maryland (2 groups)
University of Massachusetts
University of Michigan
University of Neuchatel
University of North Carolina, Chapel Hill (2 groups)
University of North Texas
University of Padova
University of Pennsylvania
University of Pisa
University of Sheffield
University of Southern California, ISI
University of Toronto
University of Waterloo
University of York
Virginia Tech
Yonsei University

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 ad hoc task of the web 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 title field has been used differently in topics 451–550, the web track's ad hoc topics, as described below.) 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:  508
<title> hair loss is a symptom of what diseases

<desc> Description:
Find diseases for which hair loss is a symptom.

<narr> Narrative:
A document is relevant if it positively connects the loss of head hair in humans with a
specific disease.  In this context, "thinning hair" and "hair loss" are synonymous.  Loss
of body and/or facial hair is irrelevant, as is hair loss caused by drug therapy.
```

Figure 1: A sample TREC 2001 topic from the web 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.

This standard procedure for topic creation was tweaked to create the topics for the ad hoc task in the web track. Participants in the web track were concerned that the queries that users type into current web search engines are quite different from standard TREC topic statements. However, if participants were given only the literal queries submitted to a web search engine, they would not know the criteria by which documents would be judged. As a compromise, standard TREC topic statements were retrofitted around actual web queries. This year's actual web queries were taken from a MSNSearch log that NIST obtained from Sue Dumais of Microsoft. A sample of queries that were deemed acceptable for use in a government-sponsored evaluation was given to the assessors. Each assessor selected a query from the sample and developed a description and narrative for that query. The assessors were instructed that the original query might well be ambiguous (e.g., "cats"), and they were to develop a description and narrative that were consistent with any one interpretation of the original (e.g., "Where is the musical Cats playing?"). They then searched the web document collection to estimate the likely number of relevant documents for that topic. The "title" field of topics 451–500 (TREC-9 web topics) contains the literal query that was the seed of the topic. For this year's topics 501-550, NIST corrected the spelling of the words in the MSNSearch queries, but otherwise left the queries as they were submitted, leaving other errors such as punctuation or grammatical mistakes in the title fields. The description and narrative fields use correct (American) English.

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 [11].

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 [14]. 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 [13]. 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 [11].

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 two of the tasks in TREC 2001, the ad hoc task in the web 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 2001.

3 TREC 2001 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 2001 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

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
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
Spanish	—	—	4	10	7	—	—	—	—	—
Confusion	—	—	—	4	5	—	—	—	—	—
Database Merging	—	—	—	3	3	—	—	—	—	—
Filtering	—	—	—	4	7	10	12	14	15	19
Chinese	—	—	—	—	9	12	—	—	—	—
NLP	—	—	—	—	4	2	—	—	—	—
Speech	—	—	—	—	—	13	10	10	3	—
Cross-Language	—	—	—	—	—	13	9	13	16	10
High Precision	—	—	—	—	—	5	4	—	—	—
Very Large Corpus	—	—	—	—	—	—	7	6	—	—
Query	—	—	—	—	—	—	2	5	6	—
Question Answering	—	—	—	—	—	—	—	20	28	36
Web	—	—	—	—	—	—	—	17	23	30
Video	—	—	—	—	—	—	—	—	—	12
Total participants	22	31	33	36	38	51	56	66	69	87

able to retrieve relevant documents regardless of the language a document happens to be written in. The TREC 2001 cross-language track used Arabic documents and English or French 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 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.

Twenty-five 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 twenty-five 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 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 entire set of 25 English topics was also translated into French by Sylvain Soliman of the Délégation Générale pour l'Armement. The three different versions of the topics were then 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-eight runs from ten different groups were submitted to the track. Twenty-eight of the runs were cross-language runs, including three runs that used French as the topic language. One monolingual run and one cross-language run were manual runs.

The assessment pools were created using three runs from each group (based on the group's assigned priorities) and using the top 70 documents from each run. The average size of the pools was 910 documents. Fourteen monolingual runs and sixteen cross-language runs were added to the pools. None of the runs whose topic language was French was judged. The LDC assessors judged each document in the pools using binary (relevant/not relevant) assessments.

The average number of relevant documents over the 25 topics is 164.9 with five topics having more than

300 relevant documents. The combination of fairly broad topics and ten participating groups resulted in pools that were not as complete as they ideally would be: for seven topics, more than half of the relevant documents were retrieved by only one group, and for another six topics between 40 and 50 % of the relevant documents were retrieved by only one group. The test whereby a group's unique relevant documents are removed from the relevance judgments shows that mean average precision scores decrease by an average of 8 %, with a maximum difference of 21 %. These differences are a little larger than those that have been reported for other TREC cross-language collections [13]. They suggest that experimenters who find many unjudged documents in the top-ranked list of only one of a pair of runs to be contrasted should proceed with care.

While the effectiveness of cross-language runs is traditionally reported as a percentage of monolingual effectiveness, the most effective run submitted to the track was a a cross-language run, BBN10XLB submitted by BBN. The difference between the effectiveness of BBN10XLB and the corresponding monolingual run, BBN10MON, is small (mean average precision scores of .4639 and .4537 respectively), but this is the second year that a cross-language run was better than all submitted monolingual runs. In the TREC-9 cross-language track the task was retrieving Chinese documents with either English or Chinese topics. Several groups, including BBN, submitted cross-language runs that were more effective than their monolingual counterparts.

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. There were three tasks in the TREC 2001 filtering track, an adaptive filtering task, a batch filtering task, and a routing task. The main focus of the track was the adaptive filtering 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 2001 filtering task used the recently released "Reuters Corpus, Volume 1, English language, 1996-08-20 to 1997-08-19" collection from Reuters (http://about.reuters.com/researchandstandards/corpus/) as training and test data. 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. A hierarchy of the Reuters category codes is included with the corpus.

Each topic for the filtering track was a category code. The topic statement included both the category identifier (so systems could use the hierarchical information about categories if desired) and the name of category. Eighty-four topics were created for the track by selecting category codes that were assigned to at least 2 training documents, but no more than 5 % of the training documents. The training documents were the set of documents from from August, 1996.

A document was considered to be relevant to a topic if the document was assigned the topic's category code. Since all this data is part of the Reuters corpus, no relevance judgments were made at NIST for the track.

Research into appropriate evaluation methods for filtering runs (which do not produce a ranked list and therefore cannot be evaluated by the usual IR evaluation measures) has been a major thrust of the filtering track. The earliest filtering tracks used linear utility functions as the evaluation metric. With a linear utility function, 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. Utility functions are attractive because they directly reflect the experience of a user of the filtering system. Unfortunately, there are drawbacks to the functions as evaluation measures. Utilities do not average well because the best possible score for each topic is a function of the number of relevant documents for that topic, and the worst possible score is essentially unbounded. Thus topics that have many relevant documents will dominate an average, and a single poorly performing topic can eclipse all other topics. Furthermore, it is difficult to know how to set the relative worth of relevant and irrelevant documents.

Two different measures were used as the primary measures for the TREC 2001 filtering tasks. The first was a linear utility function that rewarded systems two points for retrieving a relevant document and penalized systems one point for retrieving an irrelevant document. To compute average utility over the 84 topics, the utility score for each topic was first scaled between an upper and lower bound (2 × number-relevant and -100, respectively). The second measure was a version of van Rijsbergen's F measure [10]. This measure is a function of recall and precision plus a variable, β, that controls the relative importance of precision over recall. For the filtering track, β was set to .5, which emphasizes precision. If R^+ is the number of relevant documents that were retrieved, R^- the number of relevant documents that were not retrieved, and N^+ the number of non-relevant documents that were retrieved, the F score used in the track is defined as

$$\mathrm{T10F} = \begin{cases} 0 & \text{if } \mathrm{R}^+ = \mathrm{N}^+ = 0 \\ \dfrac{1.25\mathrm{R}^+}{.25\mathrm{R}^- + \mathrm{N}^+ + 1.25\mathrm{R}^+} & \text{otherwise} \end{cases}$$

Routing runs were evaluated using mean average precision since routing runs produce a ranked list of documents.

Sixty-six runs from nineteen different groups were submitted to the filtering track. Of these, 30 runs were adaptive filtering runs, 18 were batch filtering runs, and 18 were routing runs.

Because of the way the topics and relevance judgments were defined, the topics used in this year's filtering track had a much larger average number of relevant documents than the collections that have been used in previous years' tracks. Indeed, some topics had considerably more than 1000 relevant documents (so routing results for those topics are likely not meaningful since routing submissions were limited to a maximum of 1000 documents per topic). When there are relatively few relevant documents, retrieving very few documents can produce a good utility score. For this year's task, however, retrieving few documents produced a poor score. Retrieving no documents produced an average scaled utility score of 0.03. Many of the adaptive filtering submissions had a scaled utility greater than 0.2, with the best adaptive filtering run, oraAU082201 submitted by Oracle, obtaining a scaled utility score of 0.29.

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 2001 track was the first 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]. During this first year of the plan, TREC 2001 participants performed observational studies of subjects using publicly-accessible tools and the live web to accomplish a search task. Participants devised their own objectives for the studies, though a common objective for all the studies was to suggest a testable hypothesis for use in TREC 2002. Each searcher who participated in a study performed four searches, two from a list of fully specified search problems and two from a list of partially specified search problems. The lists of search problems were defined by the track and came from four domains:

- finding consumer medical information,
- buying an item,

Table 3: Focus of particular study done in the interactive track by each participant

CSIRO:	the correlation between searching/presentation mechanisms and search tasks
Glasgow:	the extent to which implicit evidence of relevance can be substituted for explicit evidence
OHSU:	how subjects use the Web
Rutgers:	ways to obtain longer queries (line vs. box)
U. Michigan:	the effect of domain knowledge in search effectiveness
U. Toronto:	the use of categories vs. standard search statements

- travel planning, and
- collecting material for a project on a given subject.

For example, a fully specified search problem for the travel planning domain was "Identify three interesting things to do during the weekend in Kyoto, Japan." A partially specified search problem for the same domain was "Identify three interesting places to visit in ____." Track participants were required to collect demographic information about the searchers, to collect the URLs of all pages visited during all searches, and to collect whatever other data made sense for the aims of their study.

Six groups participated in the interactive track. The main focus of each group's study is given in Table 3.

3.4 The Question Answering (QA) track

The purpose of the question answering track was to encourage research into systems that return actual answers, as opposed to ranked lists of documents, in response to a question. The TREC 2001 track contained three different tasks: the main task, the list task, and the context task. All of the tasks required completely automatic processing.

The main task was the focus of the track, and was essentially the same as the task in the TREC-8 and TREC-9 QA tracks. Participants received a set of fact-based, short-answer questions and searched a large document set to extract (or construct) an answer to each question. Participants returned a ranked list of five [*document-id*, *answer-string*] pairs per question such that each answer string was believed to contain an answer to the question and the document supported that answer. Answer strings were limited to no more than 50 bytes. Unlike previous years, questions were not guaranteed to have an answer in the collection. The system could return "NIL" as one of the five choices to indicate its belief that the collection did not contain an answer to the question.

As an additional, experimental part of the main task, systems were also required to return a "final answer" for each question. The final answer was either an integer from one to five that referred to a position in the ranked list of responses for that question, or the string "UNSURE" that indicated the system did not know what the answer was.

The main task was evaluated using the mean reciprocal rank measure that was used in previous years. 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 run was then the mean of the individual questions' reciprocal ranks. The correctness of a response was determined by human assessors. The assessors read each response and decided whether the answer string contained an answer to the question. If not, the response was judged as incorrect. If so, the assessor decided whether the answer was supported by the document returned with the string. If the answer was not supported by that document, the response was judged as "Not Supported". If it was supported, the response was judged as correct. The official scoring for the track treated Not Supported answers as incorrect. The NIL response was scored the same as other responses (except that it could never be marked Not Supported). NIL was counted as correct when no correct answer was known to exist in the collection for that question.

The document collection used for all the tasks in the QA track was the set of newspaper and newswire articles on TREC disks 1–5. The questions for the main task continued a progression of using more realistic questions in each of the three runnings of the track. In TREC-8, the majority of the questions were created expressly for the track, and thus tended to be back-formulations of a statement in a document. In TREC-9, the questions were selected from an Encarta log that contained actual questions, and a raw Excite log. Since the raw Excite log did not contain many grammatically well-formed questions, NIST staff used the Excite log as a source of ideas for actual questions. All the questions were created without looking at any documents. The resulting test set of question was much more difficult than the TREC-8 set, mainly because the TREC-9 set contained many more high-level questions such as *Who is Colin Powell?*. For this year's main task, the source of questions was a set of filtered MSNSearch logs and AskJeeves logs. Raw logs were automatically filtered (at Microsoft and AskJeeves) to select queries that contained a question word (e.g., what, when, where, which, etc.) anywhere in the query; that began with modals or the verb to be (e.g., are, can, could, define, describe, does, do, etc.); or that ended with a question mark. NIST did additional human filtering on these logs, removing queries that were not in fact questions; questions that asked for a list of items; procedural questions; questions that asked for the location of something on the web (e.g., pictures of someone); yes/no questions; and questions that were obviously too current for the document collection (e.g., questions about Britney Spears, etc.). The assessors then searched the collection looking for answers for the queries that remained. NIST fixed the spelling, punctuation, and sometimes the grammar of the queries selected to be in the final question set, but except for a very few (less than 10) questions, the content of the question was precisely what was in the log. The few changes that were made were simple changes such as substituting one Greek god for another so that the question would have an answer in the collection.

The final question set for the main task consisted of 500 questions. The question set contained many more definitional questions (e.g., *What are steroids?*) than in previous years, reflecting the content of the logs. Forty-nine of the questions have no known correct answer in the document collection.

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 *Name 8 Chuck Berry songs*. 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. For the Chuck Berry question, each of the answer-strings were to contain the title of a different Chuck Berry song.

The questions 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. A judgment of Correct/Incorrect/Not Supported was made for each individual pair in the list using the same criteria as in the main task. While judging for correctness, the assessor 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. Incorrect responses were always marked as not distinct, but Not Supported responses could be marked distinct. The accuracy score for a list question was computed as the number of distinct instances retrieved divided by the number of requested instances. The score for a run was the average accuracy over the 25 questions.

The context task was intended to test the systems' ability to track discourse objects (the context) through a series of questions. The expected answer types for questions in the context task were the same as in the main task. However, the questions were grouped into different series, and the QA system was expected to track the discourse objects across the individual questions of a series. For example, the following three questions were one series in the test set:

1. In what country was the first telescope to use adaptive optics used?
2. Where in the country is it located?
3. What is the name of the large observatory located there?

To answer the second question, the system needs to resolve "it" to "the first telescope to use adaptive optics", and "the country" to the country that was the answer to the first question. Similarly, "there" in the third question needs to be resolved to the answer of the second question.

NIST staff created ten question series for the context task. Most series contained three or four questions, though one series contained nine questions. There were 42 questions across all series, and each question was guaranteed to have an answer in the document collection. The context task questions were judged and evaluated as in the main task; all questions were judged by the same assessor.

A total of 92 runs was submitted to the QA track, including submissions from 36 different groups. Each group submitted at least one main task run for a total of 67 main task runs. Ten groups groups submitted eighteen runs for the list task, and six groups submitted seven runs for the context task.

The main task systems can be divided into two broad categories: systems that attempt a full understanding of the question, and systems that use a more shallow data-driven approach. The data-driven approaches rely on simpler pattern matching methods using very large corpora (frequently the web) rather than sophisticated language processing. Both approaches were successful in this year's track, with both approaches equally represented in the top systems.

3.5 The Video track

TREC 2001 was the first year for the video track, a track designed to promote progress in content-based retrieval from digital video. As befits the first running of a track, the video track included a broad range of tasks and search strategies so the track could document the current state-of-the-art in video processing.

The video collection used in the track consisted of approximately eleven hours of MPEG-1 recordings. It contains video from "NIST Digital Video Collection, Volume 1" (`http://www.nist.gov/srd/nistsd26.htm`), from the Open Video Project (`http://www.open-video.org/`), and stockshots from the BBC. Much of the video includes a sound track, though the amount and quality of the audio varies. A transcript of the soundtrack was available for the portion of the collection drawn from the NIST Digital Video Collection, and there were keyword descriptions of the BBC stockshots. The collection consists mostly of documentaries, covering a variety of subjects. The collection also contains a variety of different production techniques.

The track included three different tasks, a shot boundary task, a known-item search task, and an ad hoc search task. The two search tasks could use either automatic or manual processing, while the shot boundary task was restricted to automatic processing.

The goal in the shot boundary task was to (automatically) identify the shot boundaries in a given video clip. Since the retrieval objects in the search tasks were shots, shot boundary detection is a fundamental component of the other tasks. It is also the video processing task that has received the most attention, so it made a good basic task for the track. The shot boundary task was performed on a 5 hour subset of the whole collection. System output was evaluated using automatic comparison to a set of reference shot boundaries created manually at NIST.

Topics for the two search tasks were contributed by the participants. The final set contained 74 topics, which was the union of the topics contributed from all participants. A topic statement included a text description of the information wanted (e.g., "scenes that depict the lift off of the Space Shuttle") and possibly some examples (either video, image, or audio, as appropriate) of that type of information. Each topic was also tagged as to whether it was intended for interactive (manual) processing, automatic processing, and/or a known-item search.

For the known item search task, participants could submit up to 100 shots maximum per topic. System results were automatically compared to the list of shots identified during topic development to determine which shots in the system output were correct. (Since there is no official reference set of shot boundaries, matching system responses to known items is a fuzzy matching process. See the video track overview for details about the matching process.) Submissions were evaluated using recall and precision over the retrieved set of shots.

For the ad hoc search task, participants could submit up to 20 shots maximum per topic. NIST assessors reviewed each submitted clip and made a binary judgment as to whether it satisfied the topic. Submissions were evaluated using the precision of the retrieved set.

Twelve groups participated in the video track, submitting a total of 36 runs. Fifteen of the runs were boundary shot runs, and the remaining 21 runs contained search results (known-item search results, ad hoc search results, or both types of search results). The boundary shot results showed that the systems participating in TREC are almost perfect at recognizing boundaries that result from cuts, while other more gradual changes are more difficult to recognize. The search tasks, particularly the known-item task, are challenging problems for automatic systems.

3.6 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. The track contained two tasks in 2001, the "ad hoc" task and the homepage finding task. The ad hoc task was a traditional ad hoc retrieval task where documents (web pages) were ranked by decreasing likelihood of meeting the information need provided in the topic statement. The homepage finding task was a known-item task where the goal was to find the homepage (or site entry page) of the site described in the topic. The homepage finding task was introduced this year to allow exploration of a retrieval task for which link-based methods are known to be beneficial.

The document collection used for both tasks was the WT10g collection (`http://www.ted.cmis.csiro.au/TRECWeb/access_to_data.html`) used in previous web tracks. This collection is a ten gigabyte subset of a snapshot of the web in 1997. The original snapshot of the web was provided by the Internet Archive. The WT10g collection is a sample of the snapshot selected in such a way as to balance a number of competing desiderata including final size, containing some content-heavy pages, having naturally defined sub-collections contained within the collection, and having a good closed set of hyperlinks (see `http://www.ted.cmis.csiro.au/TRECWeb/wt10ginfo.ps.gz`).

The topics used in the main web task were TREC topics 501–550. As described earlier, these topics were created by NIST assessors by retrofitting topic statements around MSNSearch queries. Three-way relevance judgments (not relevant, relevant, and highly relevant) were used again this year since the results of last year's track showed that the relative quality of runs differed depending on whether evaluation was based on all relevant documents are highly relevant documents only [12]. The evaluation results reported in Appendix A of the proceedings are computed using all relevant documents. Evaluation of the ad hoc task runs is based on `trec_eval`.

The topics for the homepage finding task consisted of a single phrase such as "HKUST Computer Science Dept.". For each topic, the system was to return the home page of the entity described by the topic. For the example topic, the system should retrieve the home page for the computer science department of the Hong Kong University of Science and Technology. The home page of a related site of a different granularity, such as the home page for the entire university or a project within the computer science department was counted as incorrect for this task.

NIST assessors developed 145 topics for the homepage finding task in the following way. An assessor was given a list of 100 randomly selected pages from the WT10g collection. For each page, the assessor followed links to get to a page that he considered to be the home page of a site that contained the original page. Randomly selected pages that did not contain links or that contained links that could not be followed were skipped. Once at a homepage, the assessor created a descriptive phrase for it such that he could imagine someone using that phrase as a query. The assessors were instructed that the phrase should be short, but descriptive enough to distinguish a single site (i.e. "cs dept" alone is realistic, but is not descriptive enough for this task).

Participants returned a ranked list of 100 documents per topic. Small pools consisting of the top twelve pages from each judged run were created to check for pages that had different DOCNOs but were equivalent homepages (caused by mirroring and the like). The rank of the homepage whose rank was closest to one was used as the score for each topic. That is, if a topic had two homepages and a system retrieved the pages at ranks three and seven, then only the homepage at rank three was considered in the scoring. The main evaluation measure for the homepage finding task was the mean reciprocal rank of the homepage.

Thirty groups submitted a total 140 runs to the web track. Of those runs, 97 were ad hoc task runs and 43 were homepage finding task runs. The ad hoc task guidelines specified that at least one of the ad hoc task runs from a group should be automatic runs that used only the title portion of the topic statement ("short" runs) since such runs are the most realistic for the web environment. Seventy of the ad hoc task runs were short runs. The remaining twenty runs included two manual runs and automatic runs that used other parts of the topic statement. Since a large majority of the ad hoc runs were short runs, documents that were retrieved only by shorts runs made up 65 % of the judging pools and 26 % of the relevant documents.

Documents that were retrieved by both a short run and some other type of run made up 20 % of the pools and 61 % of the relevant documents. The two groups that did a manual run were the two groups that found the greatest numbers of unique relevant documents. Each of the two manual runs had a decrease of about 3.5 % in mean average precision when evaluated with and without its group's unique relevant documents. No automatic run had a difference greater than 2.5 % (provided the mean average precision score was at least 0.1).

The retrieval results from the track support the hypothesis that the two tasks within the track require different retrieval techniques. For the ad hoc task, content-based methods alone were sufficient for effective retrieval. The homepage finding task, however, required exploiting additional information, specifically URL text or link structure. The highest-ranked content-only homepage run had a mean reciprocal rank score that was only 30 % as good as the best homepage run: 0.774 for `tnout10epCAU` vs. 0.338 for `tnout10epC`, both submitted by the TNO/University of Twente group.

4 The Future

The final session of each TREC workshop is a planning session for future TRECs, in particular to decide on the set of tracks for the next TREC. Each of the TREC 2001 tracks will continue in TREC 2002. In addition, a new "novelty" track will also be included. The goal in the novelty track is to test systems' abilities to recognize repeated information in a document set. Participants in the track will receive a set of topics and a relatively small (less than 30), ranked set of relevant documents for each topic. Systems must process each document in the order given and flag sentences that contain relevant information that is not contained in previous documents. Evaluation of system performance will be a function of the overlap between the sentences flagged by the system and the sentences selected by human assessors.

TREC 2002 will also contain an exploratory effort or "pre-track" on the retrieval of genomic data. The pre-track will take a very broad definition of genomic data, including such things as research papers, lab reports, etc., as well as actual gene sequences. The purpose of the track is to foster collaboration between the retrieval and bioinformatics communities, as well as to provide a retrieval task on a particular kind of structured data.

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] C.J. van Rijsbergen. *Information Retrieval*, chapter 7. Butterworths, 2 edition, 1979.

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

[12] Ellen M. Voorhees. Evaluation by highly relevant documents. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 74–82, 2001.

[13] 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`.

[14] 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-2001 Cross-Language Information Retrieval Track: Searching Arabic using English, French or Arabic Queries

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

Abstract

Ten groups participated in the TREC-2001 cross-language information retrieval track, which focussed on retrieving Arabic language documents based on 25 queries that were originally prepared in English. French and Arabic translations of the queries were also available. This was the first year in which a large Arabic test collection was available, so a variety of approaches were tried and a rich set of experiments performed using resources such as machine translation, parallel corpora, several approaches to stemming and/or morphology, and both pre-translation and post-translation blind relevance feedback. On average, forty percent of the relevant documents discovered by a participating team were found by no other team, a higher rate than normally observed at TREC. This raises some concern that the relevance judgment pools may be less complete than has historically been the case.

1 Introduction

For the 2001 Text Retrieval Conference (TREC-2001), the Cross-Language Information Retrieval (CLIR) task was to utilize English (or French) queries against Arabic documents. Monolingual Arabic experiment designs in which both the queries and the documents were in Arabic were also supported. This was the eighth year in which non-English document retrieval has been evaluated at TREC, and the fifth year in which cross-language information retrieval has been the principal focus of that work. In TREC-3, retrieval of 25 topics against a Mexican newspaper corpus was tested by four groups. Spanish language retrieval was evaluated in TREC-3, TREC-4 (another 25 topics for the same Mexican corpus), and TREC-5 (where an European Spanish corpus was used). In TREC-5, a Chinese language track was introduced using both newspaper (People's Daily) and newswire (Xinhua) sources from People's Republic of China, and 25 Chinese topics with an English translation supplied. The TREC-5 corpus was represented with the GB character set of simplified Chinese. The Chinese monolingual experiments on this collection that were done in TREC-5 and TREC-6 sparked research into the application of Chinese text segmentation to information retrieval using dictionary-based methods and statistical techniques, and simpler overlapping bigram segmentation methods were also found to be effective. TREC-6, TREC-7 and TREC-8 had the first cross language tracks, which focussed upon European languages (English, French, German, and later Italian). Following TREC-8, the venue for European-language retrieval evaluation moved to Europe with the creation of the Cross-Language Evaluation Forum (CLEF), first held in Lisbon in September 2000 [1]. For TREC-9, the CLIR task used Chinese documents from Hong Kong. In distinction from the earlier TREC-5/6 Chinese corpus, these sources were written in the traditional Chinese character set and encoded in BIG5. Following TREC-9 the evaluation of English-Chinese retrieval moved to the NTCIR Evaluation that is coordinated by the National Institute of Informatics in Japan (http://research.nii.ac.jp/ntcir/workshop/work-en.html).

2 Task Description

As in past TREC CLIR evaluations, the principal task for each group was to match topics in one language (English or French, in this case) with documents in another language (Arabic) and return a ranked list of the top 1000 documents associated with each topic. Participating groups were allowed to submit as many as five runs, with at least one using only the title and description field of the topic description. Evaluation then proceeded by pooling ranks and manual examination of the pools by human judges to decide binary (yes/no) relevance for each document in the pool with respect to each topic. A suite of statistics were then calculated, with the mean (over 25 topics) uninterpolated average being the most commonly reported.

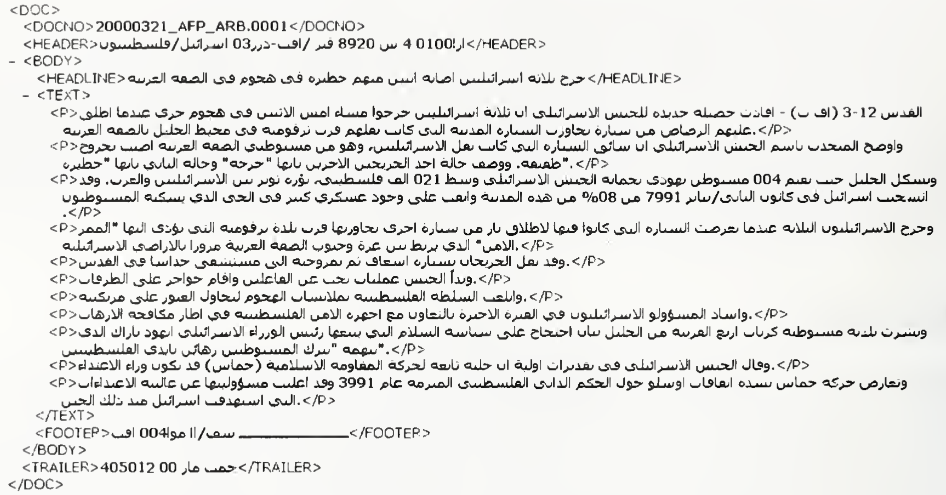
2.1 Topics

Twenty-five topic descriptions (numbered AR1-AR25) were created in English in a collaborative process between the LDC and NIST. An example of a topic description is:

```
<top>
<num> Number: AR22
<title> Local newspapers and the new press law in Jordan
<desc> Description:
Has the Jordanian government closed down any local newspapers due
to the new press law?
<narr> Narrative:
Any articles about the press law in Jordan and its effect on the local
newspapers and the reaction of the public and journalists toward the new
press law are relevant. The articles that deal with the personal suffering
of the journalists are irrelevant.
</top>
```

Through the efforts of Edouard Geoffrois of the French Ministry of Defense, the English topics were translated into French and made available to participants which wished to test French to Arabic retrieval. The French version of the topic shown above is:

```
<top>
<num> Number: AR22
<title> Les journaux locaux et la nouvelle loi sur la presse en Jordanie
<desc> Description:
Le gouvernement jordanien a-t-il interdit un journal local à cause de la nouvelle loi sur la
```



```
<DOC>
  <DOCNO>20000321_AFP_ARB.0001</DOCNO>
  <HEADER>ار!0100 4 س 8920 قبر /اف-درر03 اسرائيل/فلسطسون</HEADER>
- <BODY>
    <HEADLINE>حرح بلانه اسرائيلس اصابه اس منهم خطيره فى هجوم فى الصفه العربيه</HEADLINE>
  - <TEXT>
      <P>القدس 3-12 (اف ب) - افادت حصيله حديده للحيس الاسرائيلى ان ثلاثة اسرائيليس حرحوا مساء امس الاثنس فى هجوم حرى عندما اطلق
        عليهم الرصاص من سيارة يحاورب السياره المدنيه التى كانت بقلهم قرب ترقوميه فى محيط الخليل بالصفه العربيه.</P>
      <P>واوصح المتحدث باسم الحيش الاسرائيلى ان سائق السياره التى كانت بقل الاسرائيليس، وهو من مستوطنى الصفه العربيه اصيب بجروح
        خطيره "طفيفه. ووصف حالة احد الجريحيس الاخريس بانها "حرحه" وحاله الثاني بانها".</P>
      <P>ويسكل الخليل حيث يقيم 004 مستوطن يهودى بحمايه الحيش الاسرائيلى وسط 021 الف فلسطيني، بؤره توتر بين الاسرائيليس والعرب. وقد
        انسحبت اسرائيل فى كانون الثاني/يناير 7991 من 08% من هذه المدينة وابقت على وجود عسكري كبير فى الحى الذي يسكنه المستوطنون
        .</P>
      <P>وجرح الاسرائيليون الثلاثه عندما تعرضت السياره التى كانوا فيها لاطلاق نار من سيارة اخرى يحاوريها قرب بلدة ترقوميه التى تؤدى اليها "الممر
        الامن" الذي يربط بين غزة وجنوب الصفه العربية مرورا بالاراضى الاسرائيليه.</P>
      <P>وقد نقل الجريحان بسياره اسعاف ثم بمروحيه الى مستشفى حداسا فى القدس.</P>
      <P>وبدأ الجيس 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 الفترة الاخيرة بالتعاون مع اجهزه الامن الفلسطينيه في اطار مكافحه الارهاب.</P>
      <P>ونشرت بلديه مستوطنه كريات اربع القريبه من الخليل بيان احتجاج على سياسه السلام التى يتبعها رئيس الوزراء الاسرائيلى ايهود باراك الذى
        يتهمه "بترك المستوطنين رهائن بايدى الفلسطينيين".</P>
      <P>وقال الجيش الاسرائيلى فى تقديرات اولية ان خليه تابعه لحركة المقاومه الاسلامية (حماس) قد تكون وراء الاعتداء.</P>
      <P>وتعارض حركه حماس بشده اتفاقات اوسلو حول الحكم الذاتى الفلسطينى المبرمه عام 3991 وقد اعلنت مسؤوليتها عن غالبيه الاعتداءات
        التى استهدفت اسرائيل منذ ذلك الحين.</P>
    </TEXT>
    <FOOTER>اف 004هو/اا سف ___________</FOOTER>
  </BODY>
  <TRAILER>حمب مار 00 405012</TRAILER>
</DOC>
```

Figure 1: Example Arabic document.

presse?
<narr> Narrative:
Tout article concernant la loi sur la presse en Jordanie et ses effets sur les journaux locaux ainsi que la réaction du public et des journalistes à la nouvelle loi sur la presse est pertinent. Les articles traitant des souffrances personnelles des journalistes ne sont pas pertinents. </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.

2.2 Documents

The document collection used in the TREC-2001 CLIR track consisted of 383,872 newswire stories that appeared on the Agence France Press (AFP) Arabic Newswire between 1994 and 2000. The documents were represented in Unicode and encoded in UTF-8, resulting in a 896 MB collection. A typical document is shown in Figure 1.

3 Relevance Judgments

The ten participating research teams shown in Table 1 together produced 24 automatic cross-language runs with English queries, 3 automatic cross-language runs with French queries, 19 automatic monolingual runs with Arabic queries, and 2 manual runs (one with English queries and one with Arabic queries). From these, 3 runs were selected from each team in a preference order recommended by the participants for use in forming assessment pools. The resulting pools were formed from 15 cross-language runs with English queries, 1 cross-language run with French queries, and 14 monolingual runs with Arabic queries. The top-ranked 70 documents for a topic in each of the 30 ranked lists were added to the judgment pool for that topic, duplicates were removed, and the documents then sorted in a canonical order designed to prevent the human judge from inferring the rank assigned to a document by any system. Each document

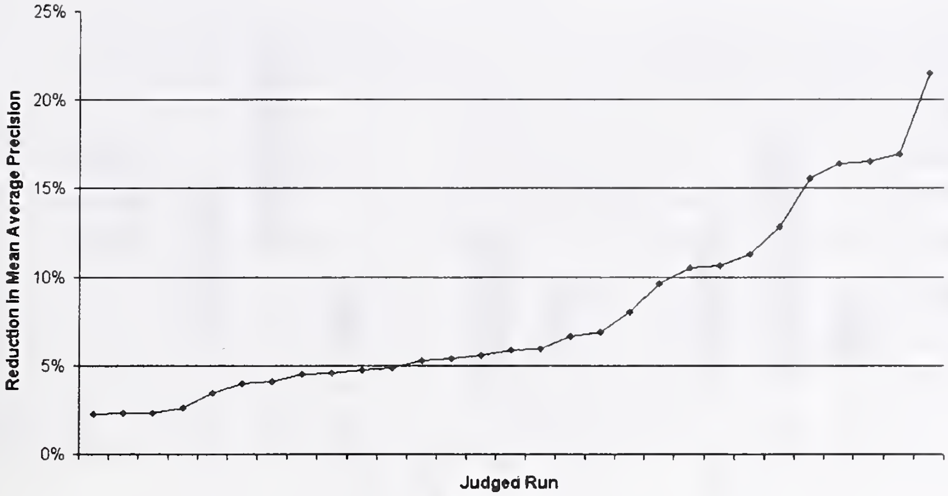


Figure 2: Effect on 29 judged runs of removing "uniques" contributed by that run.

in the pool was then judged for topical relevance, usually by the person that had originally written the topic statement. The mean number of relevant documents that were found for a topic was 165.

Most documents remain unjudged when pooled relevance assessments are used, 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 was typically less than 5% [2]. As Figure 2 shows, this effect is substantially larger in the TREC-2001 Arabic collection, with 9 of the 28 judged automatic runs experiencing a relative reduction in mean uninterpolated average precision of over 10% relative when the "uniques" contributed by that run were removed from the judgment pool.

Figure 3 helps to explain this unexpected condition, illustrating that many relevant documents were found by only a single participating research team. For 7 of the 25 topics, more than half of the known relevant documents were ranked in the top-70 in runs submitted by only a single research team. For another 6 of the 25 topics, between 40 and 50 percent of their relevant documents were ranked in the top-70 by only one team.

These results show a substantial contribution to the relevance pool from each site, with far less overlap than has been typical in previous TREC evaluations. This limited degree of overlap could result from the following factors:

- A preponderance of fairly broad topics for which many relevant documents might be found in the collection. The average of 165 relevant documents per topic is somewhat greater than the value typically seen at TREC (100 or so).

- The limitation of the depth of the relevance judgment pools to 70 documents (100 documents per run have typically been judged in prior TREC evaluations).

- The diversity of techniques tried by the participating teams in this first year of Arabic retrieval experiments at TREC, which could produce richer relevance pools.

- A relatively small number of participating research teams, which could interact with the diversity

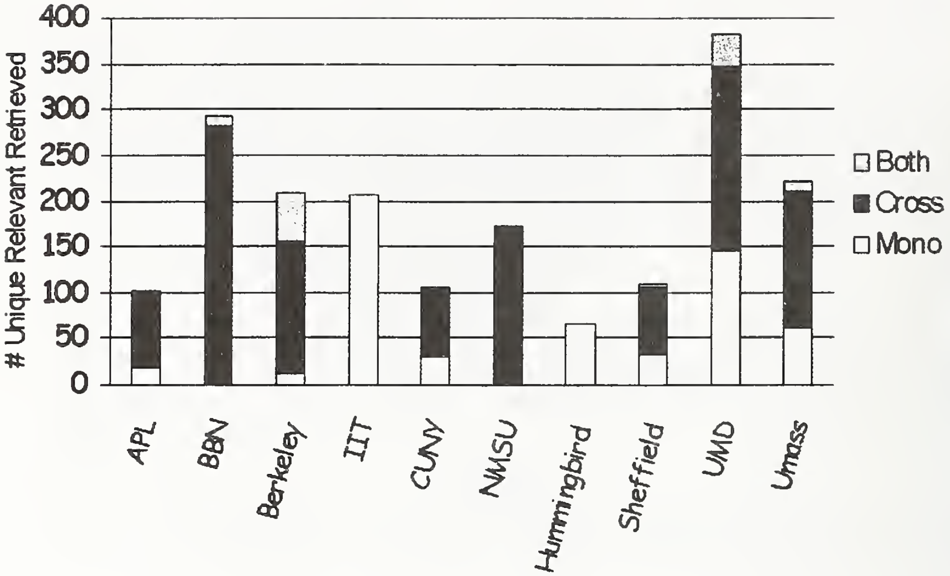


Figure 3: Unique relevant documents, by research team.

of the techniques to make it less likely that another team would have tried a technique that would find a similar set of documents.

The first two factors have occasionally been seen in information retrieval evaluations based on pooled assessment methodologies (TREC, CLEF, and NTCIR) without the high "uniques" effect observed on this collection. We therefore suspect that the dominant factors in this case may be the last two. But until this cause of the high "uniques" effect is determined, relative differences of less than 15% or so in unjudged and post hoc runs using this collection should be regarded as suggestive rather than conclusive. There is, of course, no similar concern for comparisons among judged runs since judgments for their "uniques" are available.

As has been seen in prior evaluations in other languages, manual and monolingual runs provided a disproportionate fraction of the known relevant documents. For example, 33% of the relevant documents that were found by only one team were found only by monolingual runs, while 63% were found only by cross-language runs.

4 Results

Table 1 summarizes the alternative indexing terms, the query languages, and (for cross-language runs) the sources of translation knowledge that were explored by the ten participating teams. All ten participating teams adopted a "bag-of-terms" technique based on indexing statistics about the occurrence of terms in each document. A wide variety of specific techniques were used, including language models, hidden Markov models, vector space models, inference networks, and the PIRCS connectionist network. Four basic types of indexing terms were explored, sometimes separately and sometimes in combination:

Words. Indexing word surface forms found by tokenizing at white space and punctuation requires no language-specific processing (except, perhaps, for stopword removal), but potentially desirable matches between morphological variants of the same word (e.g., plural and singular forms) are

	Arabic Terms Indexed				Query	Translation Resources Used			
Team	Word	Stem	Root	n-gram	Lang	MT	Lexicon	Corpus	Translit
BBN		X			A,E	X	X	X	
Hummingbird		X			A				
IIT	X	X	X		A,E	X	X		
JHU-APL	X			X	A,E,F	X			
NMSU	X	X			A,E		X		
Queens	X			X	A,E	X			
UC Berkeley		X			A,E	X	X		
U Maryland	X	X	X	X	A,E	X			X
U Mass	X	X			A,E	X	X		
U Sheffield	X				A,E,F	X			

Table 1: Configurations tested by participating teams.

precluded. As a result, word indexing yielded suboptimal retrieval effectiveness (by the mean uninterpolated average precision measure). Many participating research teams reported results for word-only indexing, making that condition useful as a baseline.

Stems. In contrast to English, where stems are normally obtained from the surface form of words by automatically removing common suffixes, both prefixes and suffixes are normally removed to obtain Arabic stems. Participating teams experimented with stemming software developed at three participating sites (IIT, NMSU, and U Maryland) and from two other sources (Tim Buckwalter and Shereen Khoja).

Roots. Arabic stems can be generated from a relatively small set of root forms by expanding the root using standard patterns, some of which involve introduction of infixes. Stems generated from the same root typically have related meanings, so indexing roots might improve recall (possibly at the expense of precision, though). Although humans are typically able to reliably identify the root form of an Arabic word by exploiting context to choose between alternatives that would be ambiguous in isolation, automatic analysis is a challenging task. Two participating teams reported results based on automatically determined roots.

Character n-grams. As with other languages, overlapping character n-grams offer a useful alternative to techniques based on language-specific stemming or morphological analysis. Three teams explored n-grams, with values of n ranging from 3–6.

Term formation was typically augmented by one or more of the following additional processing steps:

Character deletion. Some Unicode characters, particularly diacritic marks, are optional in Arabic writing. This is typically accommodated by removing the characters when they are present, since their presence in the query but not the document (or vice-versa) might prevent a desired match.

Character normalization. Some Arabic letters have more than one Unicode representation because their written form varies according to morphological and morphotactic rules, and in some cases authors can use two characters interchangeably. These issues are typically accommodated by mapping the alternatives to a single normalized form.

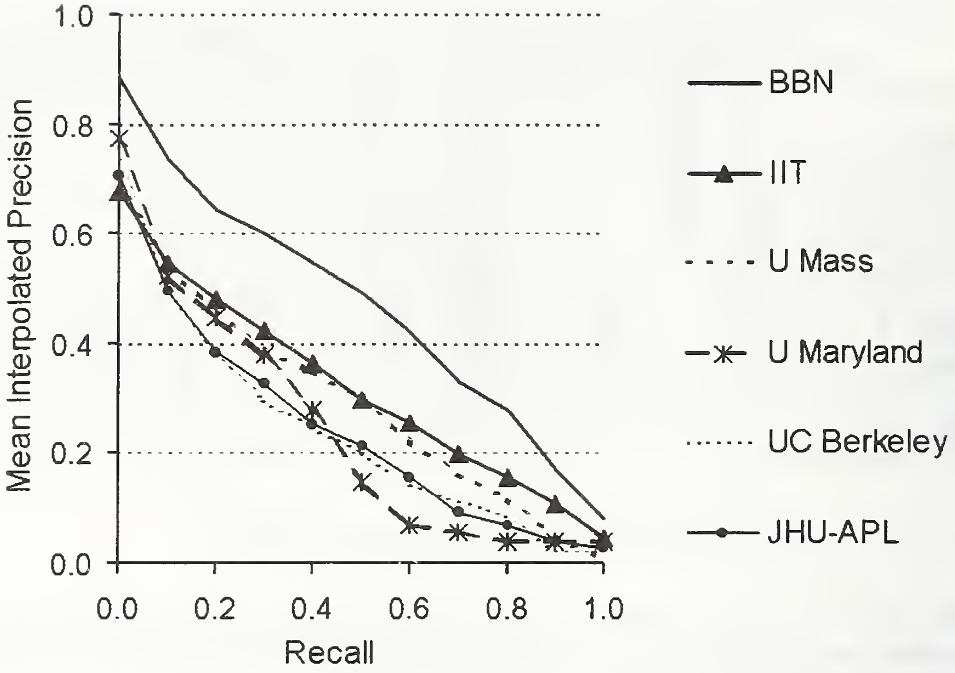


Figure 4: Cross-language retrieval effectiveness, English queries formed from title+description fields, automatic runs.

Stop-term removal. Extremely frequent terms and other terms that system developers judge to be of little use for retrieval are often removed in order to reduce the size of the index. Stop-term removal is most commonly done after stemming or morphological analysis in Arabic because the highly productive morphology would otherwise result in impractically large stopword lists.

Nine of the ten participating research teams submitted cross-language retrieval runs, with all nine using a query-translation architecture. Both of the teams that tried French queries used English as a pivot language for French-to-Arabic query translation, so English-to-Arabic resources were key components in every case. Each team explored some combination of the following four types of translation resources:

Machine Translation Systems. Two machine translation systems were used: (1) a system developed by Sakhr (available at http://tarjim.ajeeb.com, and often referred to simply as "Ajeeb" or "Tarjim"), a system produced by ATA Software Technology Limited (available at http://almisbar.com, and sometimes referred to as "Almisbar" or by the prior name "Al-Mutarjim"). At the time of the experiments, both offered only English-to-Arabic translation. Some teams used a machine translation system to directly perform query translation, others used translations obtained from one or both of these systems as one source of evidence from which a translated query was constructed. A mark in the "MT" column of Table 1 indicates that one or more existing machine translation systems were used in some way, not that they were necessarily used to directly perform query translation.

Translation Lexicons. Three commercial machine readable bilingual dictionaries were used: one marketed by Sakhr (sometimes referred to as "Ajeeb"), one marketed by Ectaco Inc., (typically referred to as "Ectaco"), and one marketed by Dar El Ilm Lilmalayin (typically referred to as "Al Mawrid"). In addition, one team (NMSU) used a locally produced translation lexicon.

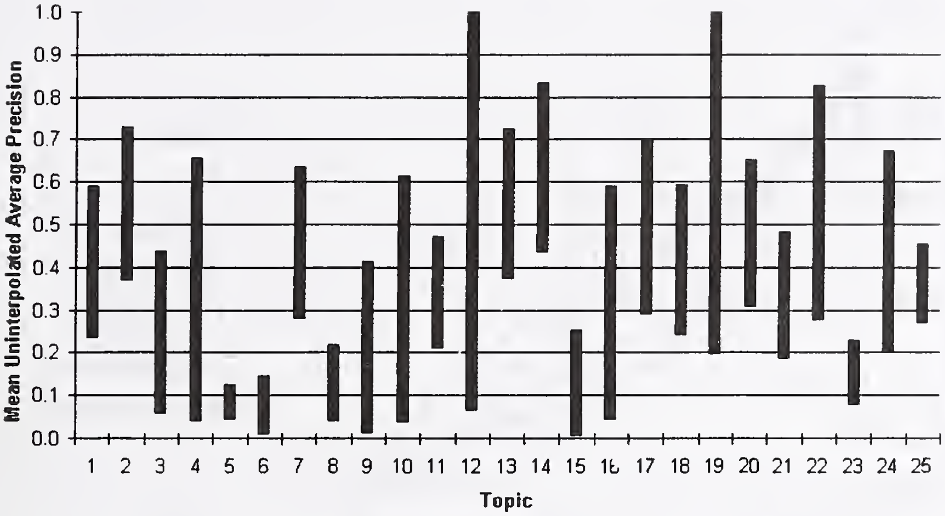


Figure 5: Cross-language topic difficulty, uninterpolated average precision (base of each bar: median over 28 runs, top of each bar: best of the 28 runs).

Parallel Corpora. One team (BBN) obtained a collection of documents from the United Nations that included translation-equivalent document pairs in English and Arabic. Word-level alignments were created using statistical techniques and then used as a basis for determining frequently observed translation pairs.

Transliteration. One team (Maryland) used pronunciation-based transliteration to produce plausible Arabic representations for English terms that could not otherwise be translated.

When multiple alternative translations were known for a term, a number of techniques were used to guide the combination of evidence, including: (1) translation probabilities obtained from parallel corpora, (2) relative term frequency for each alternative in the collection being searched, and (3) structured queries. Pre-translation and/or post-translation query expansion using blind relevance feedback techniques and pretranslation stop-term removal were also explored by several teams.

To facilitate cross-site comparison, teams submitting automatic cross-language runs were asked to submit at least one run in which the query was based solely on the title and description fields of the topic descriptions. Figure 4 shows the best recall-precision curve for this condition by team. All of the top-performing cross-language runs used English queries.

As is common in information retrieval evaluations, substantial variation was observed in retrieval effectiveness on a topic-by-topic basis. Figure 5 illustrates this phenomenon over the full set of cross-language runs (i.e., not limited to title+description queries). For example, half of the runs did poorly on topic 12, which included specialized medical terminology, but at least one run achieved a perfect score on that topic. Topic 5, by contrast, turned out to be problematic for all systems.

No standard condition was required for monolingual runs, so Figure 6 shows the best monolingual run by team regardless of the experiment conditions. Several teams observed surprisingly small differences between monolingual and cross-language retrieval effectiveness. One site (JHU-APL) submitted runs under similar conditions for all three topic languages, and Figure 7(a) shows the resulting recall-precision graphs by topic language. In that case, there is practically no difference between English-topic and Arabic-topic results. There are two possible explanations for this widely observed effect:

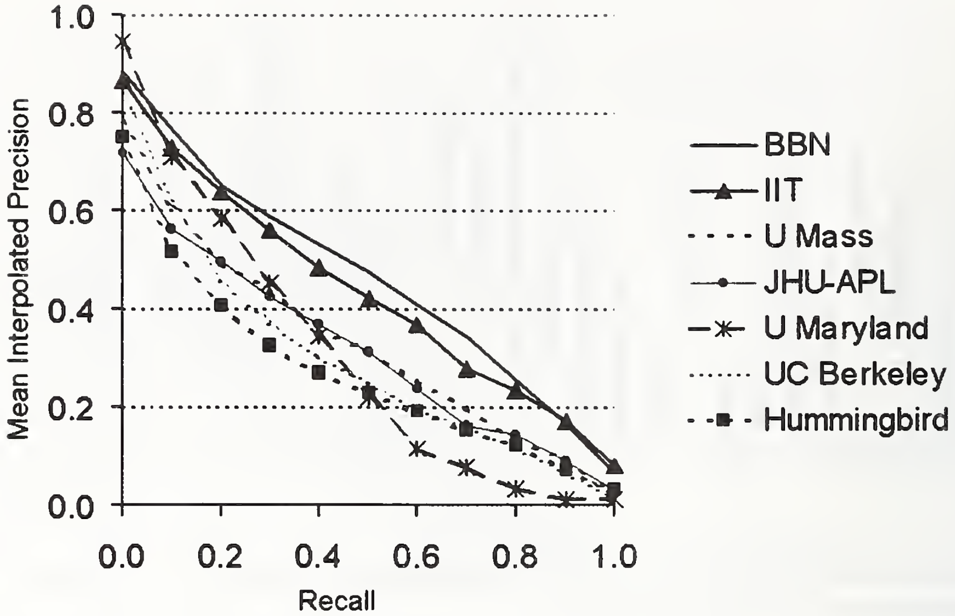


Figure 6: Monolingual retrieval effectiveness, Arabic queries formed from title+description fields (except JHU-APL and UC Berkeley, which also used the narrative field), automatic runs (except U Maryland, which was a manual run).

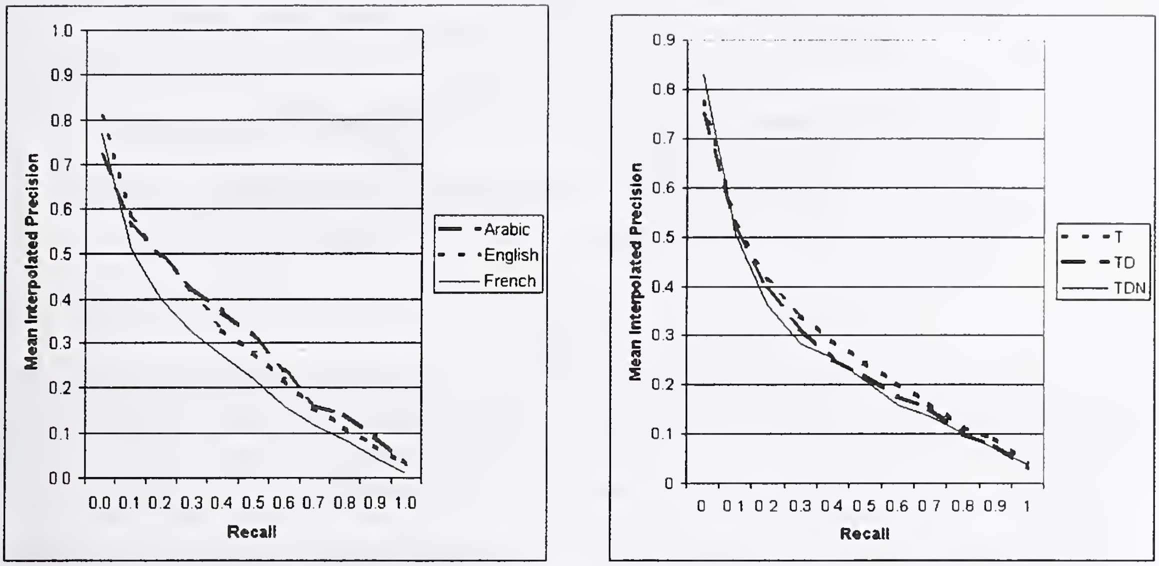
- No large Arabic information retrieval test collection was widely available before this evaluation, so the monolingual Arabic baseline systems created by participating teams might be improved substantially in subsequent years.

- The 25 topics used in this year's evaluation might represent a biased sample of the potential topic space. For example, relatively few topic descriptions this year included names of persons.

Several teams also observed that longer queries did not yield the improvements in retrieval effectiveness that would normally be expected. One site (Hummingbird) submitted runs under similar conditions for three topic lengths, and Figure 7(b) shows the resulting recall-precision graphs. In this case, longer queries showed no discernible benefit; indeed, it appears that the best results were achieved using the shortest queries! The reasons for this effect are not yet clear, but one possibility is that the way in which the topic descriptions were created may have resulted in a greater concentration of useful search terms in the title field. For example, the title fields contains an average of about 6 words, which is about twice as long as is typical for TREC.

5 Summary and Outlook

The TREC-2001 CLIR track focussed this year on searching Arabic documents using English, French or Arabic queries. In addition to the specific results reported by each research team, the evaluation produced the first large Arabic information retrieval test collection. A wide range of index terms were tried, some useful language-specific processing techniques were demonstrated, and many potentially useful translation resources were identified. In this paper we have provided an overview of that work in a way that will help readers recognize similarities and differences in the approaches taken by the



(a) Topic language effect, title+description+narrative.

(b) Query length effect, Arabic queries.

Figure 7: Comparing runs under comparable conditions (T=title, D=Description, N=Narrative).

participating teams. We have also sought to explore the utility of the test collection itself, providing aggregate information about topic difficulty that individual teams may find useful when interpreting their results, identifying a potential concern regarding the completeness of the pools of documents that were judged for relevance, and illustrating a surprising insensitivity of retrieval effectiveness to query length.

The TREC-2002 CLIR track will continue to focus on searching Arabic. We plan to use 50 new topics (in the same languages) and to ask participating teams to also rerun the 25 topics from this year with their improved systems as a way of further enriching the existing pools of documents that have been judged for relevance. We expect that the result with be a test collection with enduring value for post hoc experiments, and a community of researchers that possess the knowledge and resources needed to address this important challenge.

Acknowledgments

We are grateful to Ellen Voorhees for coordinating this track at NIST and for her extensive assistance with our preliminary analysis, to the participating research teams for their advice and insights along the way, and to Kareem Darwish for his comments on an earlier version of this paper.

References

[1] Carol Peters. *Cross-Language Information Retrieval and Evaluation.* Springer: Lecture Notes in Computer Science: LNCS 2069, Germany, 2001.

[2] Ellen M. Voorhees. Variations in relevance judgments and the mesaurement of retrieval effectiveness. In C.J. Van Rijsbergen W. Bruce Croft, Alistair Moffat, editor, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 315–323. ACM Press, August 1998.

The TREC 2001 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–10 filtering track measures the ability of systems to build persistent user profiles which successfully separate relevant and non-relevant documents. 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 large 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–10 are very similar to those investigated in TREC–7 through TREC–9.

Traditional adhoc 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–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) [7]
- TREC–4 http://trec.nist.gov/pubs/trec4/t4_proceedings.html (#11) [6]

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–10 Task Description

For those familiar with previous TRECs, the basic filtering tasks in TREC–10 are similar to those investigated in TREC–7 through TREC–9. The batch-adaptive task has been abandoned (in the interests of reducing the number of different tasks in the track), and a new evaluation measure has been introduced, in lieu of the target-based measure introduced in TREC–9. The corpus and topics are again somewhat different from those used previously. 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

For the second year, the TREC–10 filtering experiments went outside the usual TREC collections. The new corpus recently provided by Reuters for research purposes [5] was used. This is a collection of about 800,000 news stories, covering a time period of a year in 1996-7. The items are categorised according to a standard set of Reuters categories, some of which were selected as discussed below to form the "topics" for filtering (in a similar fashion to the way MeSH headings were used in TREC–9).

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 12 days' items, 20 through 31 August 1996, were taken as the training set (which could be used in ways specified below). The remainder of the collection formed the test set.

The category codes applied by Reuters are of three kinds: topic, region and industry. Only the topic codes were used; as with MeSH codes last year, the idea was to treat these categories as topics in the TREC sense, and regard the assignment of category codes by Reuters indexers as relevance judgements. One problem was the range of frequencies of assignment – some codes are extremely rare and some are applied to a substantial proportion of the collection. It was decided to limit this range at both ends, on the basis of the training set. Any code that occurred in more than 5% of the training set was rejected (this is too far removed from the usual TREC topic relevance set size). Also, since the tasks required some relevance data in the training set, specifically 2 relevant items for adaptive filtering, any code that occurred not at all or once only in the training set was rejected. The remaining 84 codes formed the basis for the filtering topics.

The Reuters Corpus is supplied with a list of codes, with a short text heading or description for each one (generally 1–3 words), and the numerical code. The numbering of the codes implies some hierarchical structure. Participants were provided with the text heading as the text of each topic (in the style of TREC 'short topics'), but were also able to make use of the hierarchical relations implied.

2.2 Tasks

The adaptive filtering task is designed to model the text filtering process from the moment of profile construction. In TREC–10, 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, a random sample of two of the relevant documents in the training set was selected and 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 two provided for each topic) to generate collection frequency statistics (such as IDF) 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 (there was no batch-adaptive task in TREC–10). 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 to open 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–10 for this purpose (as alternative sub-tasks). One was essentially the linear utility measure used in previous TRECs, and described below. The other was new to the track for TREC–10: it is a version of the van Rijsbergen measure of retrieval performance.

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 2001, 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:

$$\text{T10F} = \frac{1.25 \times \text{No. of relevant retrieved docs}}{\text{No. of retrieved docs} + 0.25 \times \text{No. of relevant docs}}$$

(T10F is set to 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:

$$\text{T10U} = 2R^{+} - N^{+}$$

which corresponds to the retrieval rule:

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

Filtering according to a 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-10 is as in TREC–8. That is, the topic utilities are scaled between limits, and the scaled values are averaged. The upper limit is the maximum utility for that topic, namely

$$\text{MaxU} = 2 \times (\text{Total relevant})$$

The lower limit is some negative utility, MinU, which may be thought of as the maximum number of non-relevant documents that a user would tolerate, with no relevants, over the lifetime of the profile. If the T10U value falls below this minimum, it will be assumed that the user stops looking at documents, and therefore the minimum is used.

$$\text{T10SU} - \frac{\max(\text{T10U}, \text{MinU}) - \text{MinU}}{\text{MaxU} - \text{MinU}}$$

for each topic, and

Mean T10SU = Mean T10SU over topics

Different values of MinU may be chosen: a value of zero means taking all negative utilities as zero and then normalising by the maximum. A value of minus infinity is equivalent (in terms of comparing sytems) to using unnormalised utility. The primary evaluation measure for TREC–10 has

$$\text{MinU} = -100$$

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:

- Macro average recall
- Macro average precision
- Mean T10SU (scaled utility) over topics, over the whole period and broken down by time period for adaptive filtering
- Mean T10F (F-beta) over topics. Note that this is referred to as F-beta, but has beta set to 0.5, as above
- Zeros, that is, the number of topics for which no documents were retrieved over the period.

For routing: the usual range of ranked-output performance measures, and the number of topics which scored P@1000 > 0.9.

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 T10F or T10SU – 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 T10SU 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. It was also possible to make limited use of other Reuters data – again, groups were asked to declare this.

3 TREC–10 results

Nineteen groups participated in the TREC–10 filtering track (five more than in TREC–9) and submitted a total of 66 runs (slightly less than last time, because of the substantially reduced number of options). These break down as follows: 12 groups submitted adaptive filtering runs, 10 submitted to batch filtering, and 11 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 recognize which runs belong to which groups in the plotted results.

	Abbreviation	Run identifier
Johns Hopkins University Applied Physics Lab	apl-jhu	apl10
Fudan University	Fudan	FDUT10
IRIT-SIG	IRIT	mer10
Justsystem Corporation	Justsystem	jscbtafr
Korea Advanced Institute of Science and Technology	KAIST	KAIST10
David D. Lewis, Independent Consultant	Lewis	DLewis01
Moscow Medical Academy	MCNIT	MMAT10
SER Technology Deutschland GmbH	SER	ser
Tampere University of Technology	Tampere	Visa
Chinese Academy of Sciences	chinese_academy	ICTAdaFT10
Clairvoyance Corporation	clairvoyance	CL01
Carnegie Mellon University	cmu-cat	CMUCAT
Carnegie Mellon University	cmu-lti	CMUDIR
Kent Ridge Digital Labs	kent_ridge	krdlT10
Microsoft Research Ltd	microsoft	ok10
Katholieke Universiteit Nijmegen	KUN	KUN
Oracle	oracle	ora
Rutgers University	rutgers-kantor	RU
University of Iowa	uiowa	UIowa

3.1 Summary of approaches

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

JHU APL participated in the routing and batch filtering tasks. Their routing runs used statistical language modeling with either character n-gram, stem, or word features. For batch filtering, they used support vector machines with different choices of feature vectors, kernels, score thresholding, and training skew factors.

Fudan University participated in the batch and adaptive filtering tasks. Their filtering profiles were a Rocchio-style weighted sum of positive training documents, with mutual information used to select terms. For adaptive filtering, their filtering procedure added pseudo-relevant documents to the initial profile, and modified both the profile and the threshold using positive and negative feedback.

IRIT-SIG participated in the routing, batch, and adaptive filtering tasks. Their Mercure system uses a spreading activation network to compute a relevance score. For routing and batch filtering, they trained their profiles using backpropagation. For adaptive filtering, they began with simple term-frequency weighted profiles and adapted both the profile and the decision threshold.

KAIST participated in the batch filtering task. They clustered the training documents in each topic into subtopics, trained a support vector machine for each subtopic, and OR'ed the binary classifier outputs to form a final decision for a topic.

David Lewis participated in the routing and batch filtering tasks. He used support vector machines with different weights for positive and negative training examples. The weighting parameter was chosen using n-fold cross-validation.

SER Technology participated in all three tasks. They used their commercial text classification system. For adaptive filtering they kept a constant decision threshold, and retrained their classifier using the top documents seen so far.

Tampere University of Technology participated in the routing task. They employed a novel profile learning technique that encodes feature terms and bins the coded values according to a statistical distribution, yielding a histogram for each word and sentence in a training document.

Chinese Academy of Sciences (CAS-ICT) participated in the adaptive filtering task. They used a vector-space approach with a profile adaptation function similar to Rocchio.

Clairvoyance participated in all three tasks. For batch filtering, they experimented with "ensembles" of simple filters in parallel or series rather than a single monolithic profile. For adaptive filtering, they used the same system as in TREC-8 to explore how well it performed on the new Reuters data.

The CMUCAT group from Carnegie Mellon University participated in the batch and adaptive filtering tasks. They compared kNN classification to the standard Rocchio method, and further explored issues with the utility metric.

The CMUDIR group from CMU participated in the adaptive filtering task. The used a language modeling approach to learn the maximum likelihood of the relevant documents discovered during filtering.

Microsoft Research Cambridge participated in the adaptive filtering task. They used their Keenbow system from last year, adding optimisation for the F-beta measure.

Katholieke Universiteit Nijmegen participated in all three tasks. Their system uses a version of Rocchio which decays terms in the profile over time, and a threshold optimisation method based on EM that models the distributions of the scores of relevant and non-relevant documents.

Oracle participated in all three tasks. They used the Oracle Text RDBMS/text retrieval system. They assigned concepts from a thesaurus to documents, summed the concept vectors from relevant training documents and selected the best concepts to represent each topic.

3.2 Evaluation results

Some results for the various participating groups are presented in the following graphs. Figures 1 and 2 show the adaptive filtering results for the two optimisation 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.

Figure 3 shows the adaptive filtering utility scores broken down into four document periods, to illustrate learning effect. For readability, only the run with the best overall mean T10SU is shown. The official results pages for the adaptive task show this data for every run.

Figures 4 and 5 show the utility and F-beta results for batch filtering. Figure 6 shows mean uninterpolated average precision for routing. Note that while there is still a range of performance, in general scores were quite high in batch filtering and routing.

3.3 Utility Scaling

In their TREC-9 paper, Ault and Yang proposed that the track use the F-beta measure instead of T9P, but did not suggest a replacement for the T9U utility measure [1]. This year, they did propose such a measure, called *normalized filtering utility*, which following our notation and filtering costs, is:

$$\begin{aligned} U_f &= \frac{2R^+ - N^+}{\text{MaxU}} \\ U_f' &= \frac{\max(U_f, U_{f,min}) - U_{f,min}}{1 - U_{f,min}} \quad (U_{f,min} = -0.5) \end{aligned}$$

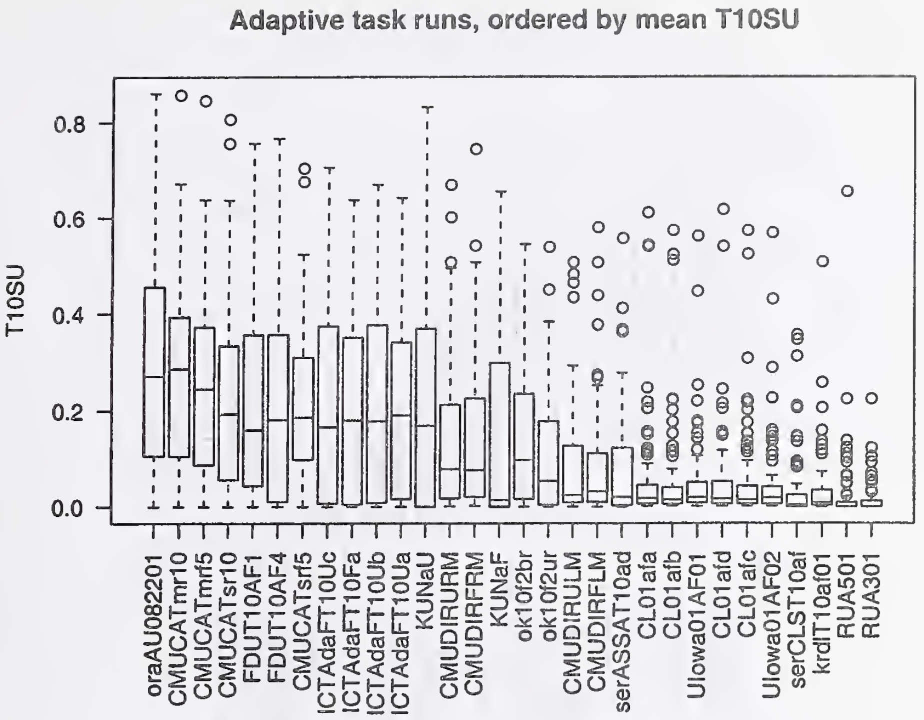


Figure 1: Adaptive filtering – T10SU

Note that this is not exactly Ault and Yang's notation; please see their paper elsewhere in this volume for details. The key difference is in how the measure is scaled between best possible and worst acceptable utility. To further discussion on utility metrics, we show the normalized filtering utility scores for adaptive filtering in Figure 7.

4 General Commentary

One major impression from the results of the TREC–10 filtering track is that the data set is significantly different in the way it behaves from those used in previous years. And the major single *a priori* difference, obvious from looking at the data set, is the number of relevant documents for each topic. There are topics which have in total tens of thousands of relevant documents, and most have hundreds or thousands. This is clearly a function of the way the topics were constructed, from Reuters categories, which tend to be fairly general (compared to typical/conventional TREC topics). Note that this is not simply a result of using predefined categories: in TREC–9, we used MeSH headings in a Medline document collection in roughly the same way. The topics defined from MeSH headings did not then appear to be very different from the more conventionally constructed ones. But MeSH headings are typically very much more specific than Reuters categories.

One effect of the use of these broad categories seems to have been as follows. In previous years, it has been very important for good performance to strictly limit the number of documents retrieved – it was all too easy to get into a non-recoverable negative utility realm by retrieving too many documents at the beginning. It is likely that some participants' thresholding methods were simply too restrictive for the conditions of the present test collection.

Nevertheless, other participants in adaptive filtering turned in very good performances. Even

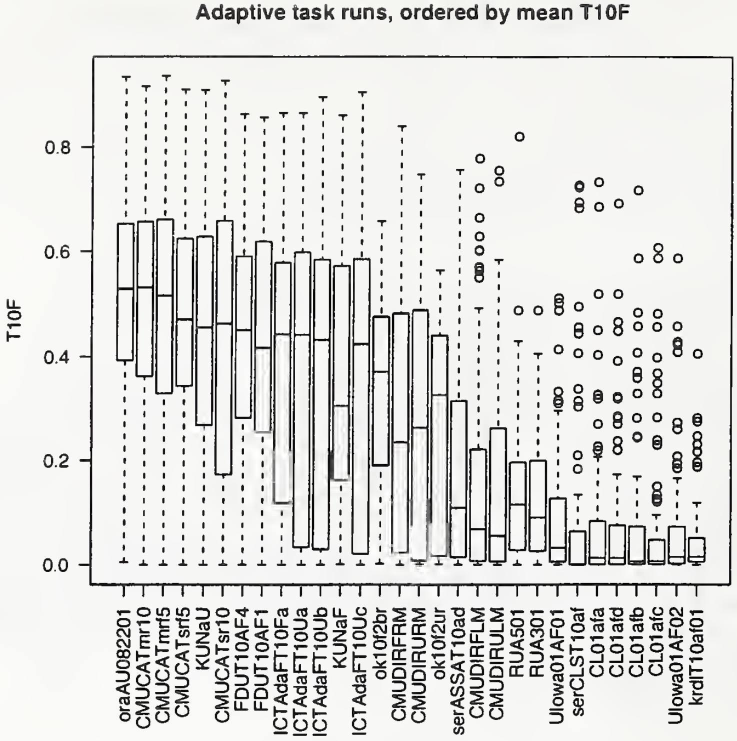


Figure 2: Adaptive filtering – T10F

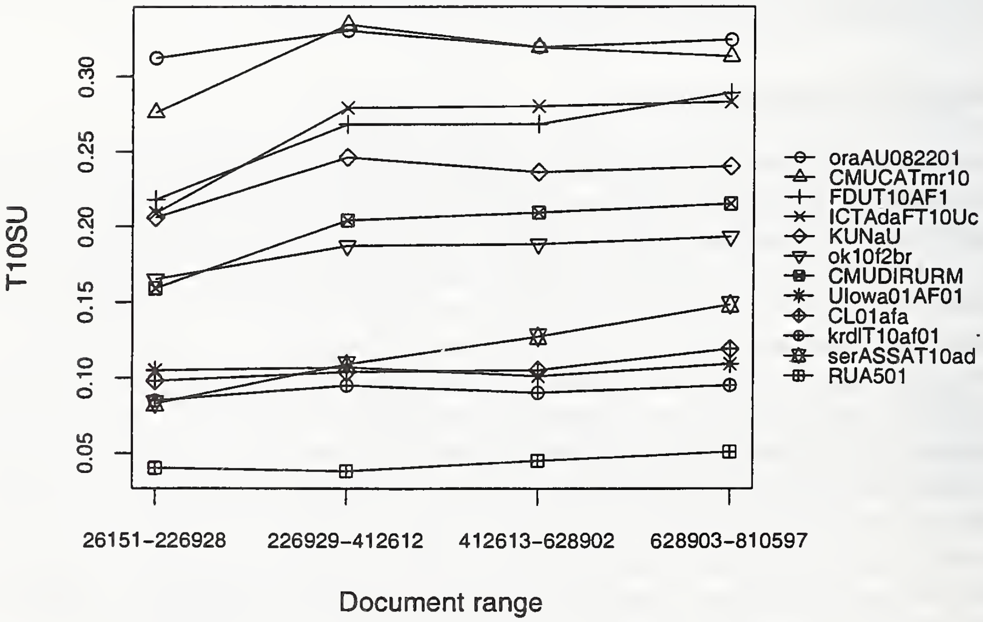


Figure 3: Adaptive filtering – T10SU within document period. The x axis labels show the document ID range in each period.

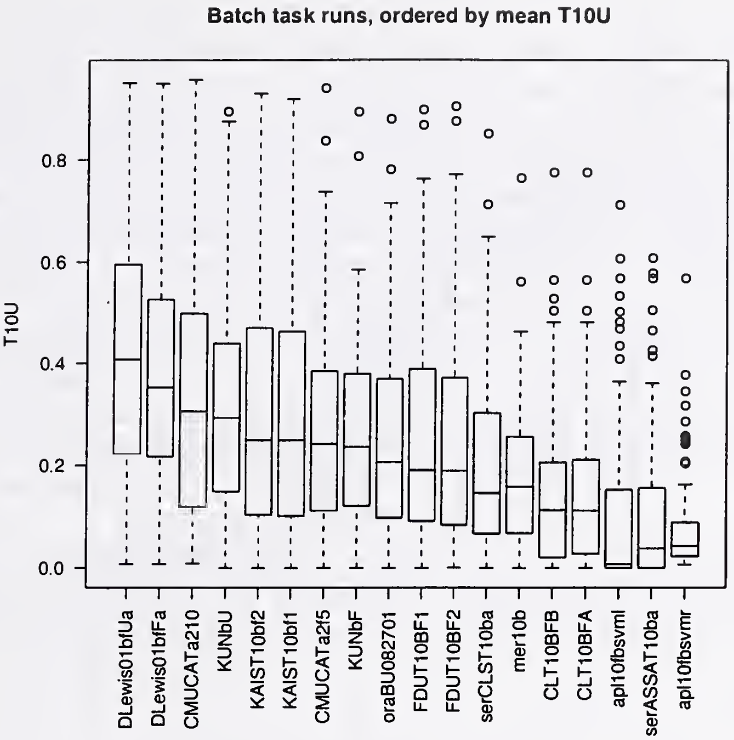


Figure 4: Batch filtering – T10SU

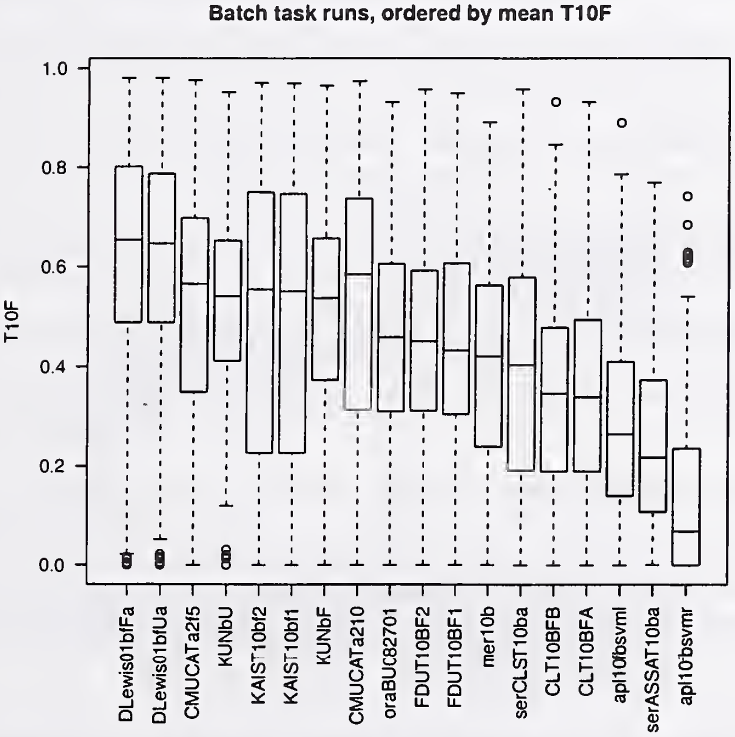


Figure 5: Batch filtering – T10F

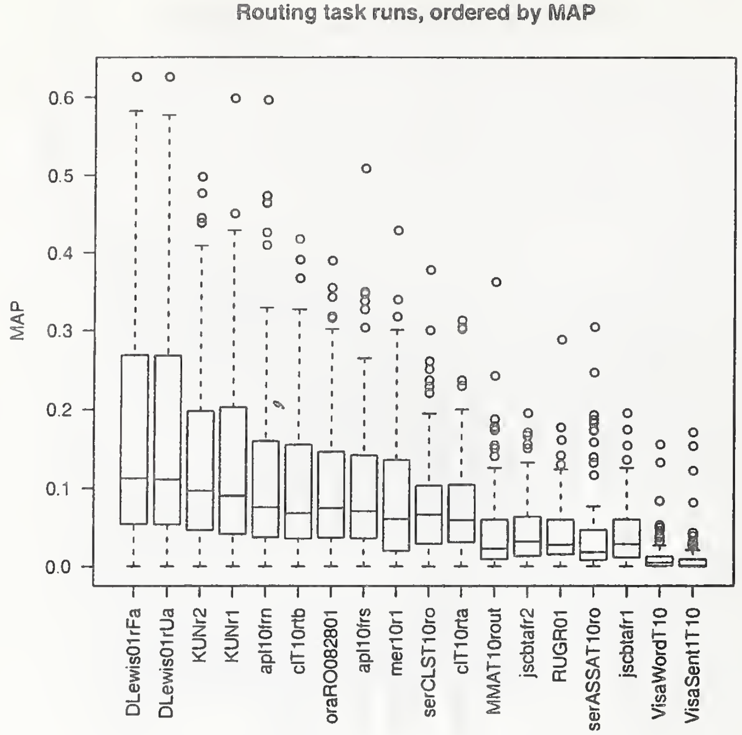


Figure 6: Routing – Mean Average Precision

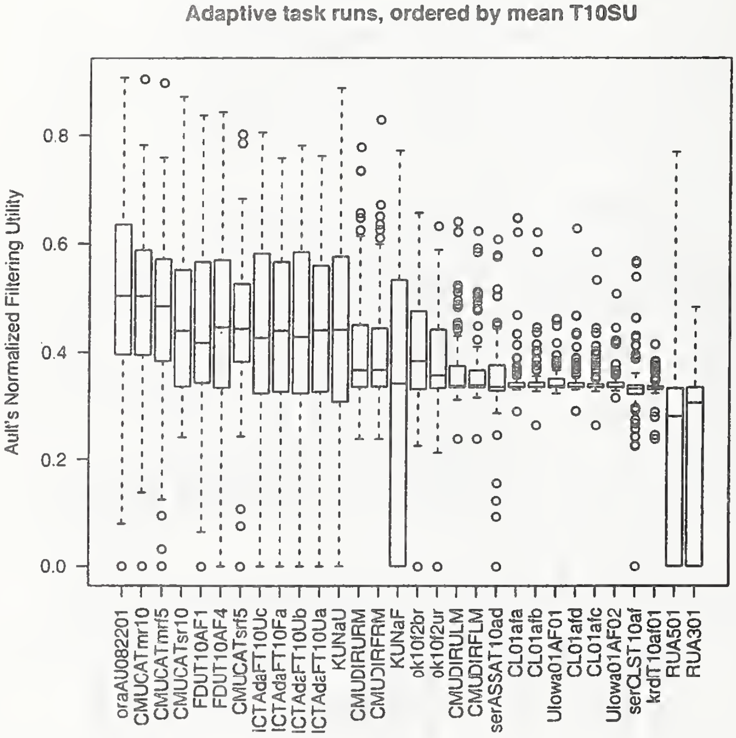


Figure 7: Adaptive filtering – Normalized Filtering Utility. The runs are ordered by mean T10SU, for comparison with Figure 1.

given the very limited starting point of 2 positive examples, it seems that a great deal of effective learning took place in the early part of the test period. The relatively flat learning curve over the whole learning period suggests not a lack of learning, but saturation. The best batch filtering and routing results also appear very good. A number of systems on a number of topics achieved over 90% precision at 1000 documents (this was the subject of some bets before TREC!).

This characteristic of the data set as defined for TREC–10 is seen as a disadvantage in a number of respects. One has to do with realism: while it may be possible to envisage situations in which a topic with 10,000 relevant documents over a year is plausible, it is well outside the sort of context we have typically imagined for a filtering system. Another is that it renders some measurements more questionable: if we can easily achieve 90% precision at 1000 documents, then that suggests that we should at a minimum evaluate much further down the ranking. This, however, would introduce logistical problems.

The likelihood is that the TREC–11 filtering track will use the Reuters corpus, but with a different set of topics. One aim would be to get back into a more reasonable range of numbers of relevant documents per topic.

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] Ault, Tom and Yang, Yiming. kNN at TREC-9. In *The 9th Text Retrieval Conference (TREC-9), NIST SP 500-249*, pages 127–134, 2001.

[2] Hull, David A. The TREC-6 Filtering Track: Description and Analysis. In *The 6th Text Retrieval Conference (TREC-6), NIST SP 500-240*, pages 45–68, 1998.

[3] Hull, David A. The TREC-7 Filtering Track: Description and Analysis. In *The 7th Text Retrieval Conference (TREC-7), NIST SP 500-242*, pages 33–56, 1999.

[4] Hull, David A. and Robertson, Stephen. The TREC-8 Filtering Track final report. In *The 8th Text Retrieval Conference (TREC-8), NIST SP 500-246*, pages 35–56, 2000.

[5] The Reuters Corpus Volume 1 – see http://about.reuters.com/researchandstandards/corpus/

[6] Lewis, David. The TREC-4 Filtering Track. In *The 4th Text Retrieval Conference (TREC-4), NIST SP 500-236*, pages 165–180, 1996.

[7] Lewis, David. The TREC-5 Filtering Track. In *The 5th Text Retrieval Conference (TREC-5), NIST SP 500-238*, pages 75–96, 1997.

[8] Robertson, Stephen and Hull, David A. The TREC-9 Filtering Track final report. In *The 9th Text Retrieval Conference (TREC-9), NIST SP 500-249*, pages 25–40, 2001.

TREC-2001 Interactive Track Report

William Hersh
(hersh@ohsu.edu)
Division of Medical Informatics and Outcomes Research
Oregon Health Sciences University
Portland, OR 97201, USA

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

Motivating principles

In the TREC 2001 Interactive Track six research teams carried out observational studies which increased the realism of the searching by allowing the use of data and search systems/tools publicly accessible via the Internet. To the extent possible, searchers were allowed to choose tasks and systems/tools for accomplishing those tasks.

At the same time, the studies for TREC 2001 were designed to maximize the likelihood that groups would find in their observations the germ of a hypothesis they could test for TREC 2002. This suggested that there be restrictions - some across all sites, some only within a given site - to make it more likely that patterns would emerge. The restrictions were formalized in two sorts of guidelines: one set for all sites and another set that applied only within a site.

Cross-site guidelines

Each site observed as many searchers as possible and appropriate. A target number of 24 was suggested.

Each searcher worked in one or more of the following domains provided by the track to all sites:

- finding consumer medical information on a given subject
- buying a given item
- planning travel to a given place
- collecting material for a project on a given subject

Each searcher carried out four searches - two from a list of fully specified tasks provided to all sites and two for which only the format was predetermined but which were otherwise up to the site/searcher to create.

Each site collected a minimal standard set of data defined roughly by the track and covering searcher characteristics and satisfaction, effectiveness, and efficiency.

Each site collected at least the urls of all pages visited during all searches.

The only real submission required was the notebook paper, which was to include among other things a testable hypothesis for TREC-2002.

Tasks

Here are the eight fully specified tasks:

- Medical
  - Tell me three categories of people who should or should not get a flu shot and why.
  - Find a website likely to contain reliable information on the effect of second-hand smoke.
- Buying
  - Get two price quotes for a new digital camera (3 or more megapixels and 2x or more zoom).
  - Find two websites that allow people to buy soy milk online.
- Travel
  - I want to visit Antarctica. Find a website with information on organized tours/trips there.
  - Identify three interesting things to do during a weekend in Kyoto, Japan.
- Project
  - Find three articles that a high school student could use in writing a report on the Titannic
  - Tell me the name of a website where I can find material on global warming.

Here are the eight partially specified tasks:

- Medical
  - List two of the generally recommended treatments for _____.
  - Identify two pros or cons of taking large doses of _____.
- Buying
  - Name three features to consider in buying a(n) _____.
  - Find two websites that will let me buy a(n) _____ online.
- Travel
  - Identify three interesting places to visit in _____.
  - I'd like to go on a sailing vacation in _____, but I don't know how to sail. Tell me where can I get some information about organized sailing cruises in that area.
- Project
  - Find three different information sources that may be useful to a high school student in writing a biography of ________.
  - Locate a site with lots of information for a high school report on the history of _____.

Within-site guidelines

Within the cross-site guidelines, each site could impose further restrictions of its own choice on ALL its searchers to define an area of interest for observation - to be reported to the track before the observations begin. Each site could define its own time limits for searches. For example, a site could have imposed inclusive or exclusive restrictions on any (combinations) of the following: the choice/assignment of domain from the 4 provided, the data to be searched, the search system/tools to be used (e.g., search systems, meta-search systems, directories,...), functionality within a given search system/tool, the characteristics of searchers, the time allowed, the pre- search training provided, etc. Sites were also encouraged to coordinate their plans with other sites, form small teams sharing guidelines, etc. Each site evaluated their searches using any criteria defined in the cross-site guidelines plus any site specific evaluations. As part of the data analysis for TREC 2001, each site was to attempt to formulate a testable hypothesis for TREC 2002 and report this as part of the results for TREC 2001.

Overview of results

A total of six groups participated in this year's Interactive Track and submitted reports for the proceedings. Even though there was no official correct "answer" for any of the tasks, most groups attempted to assess some aspect of user searching performance, usually comparing two or more groups and/or systems. See each group's report for information about the formulation of testable hypotheses.

- Toms et al. [1] had 48 subjects who were given a choice of initiating the search with a query or with a selection of a category from a pre-defined list. Participants were also asked to phrase a selected number of their search queries in the form of a complete statement or question. The results showed that there was little effect of the task domain (medical, buying, travel, report) on the search outcome. There was a preference for the use of queries over categories when the semantics of the search task did not map well to one of the available catetories.

- Bhavhani [2] compared the searching behaviors of expert vs. non-expert searchers, with medical librarians and those experienced with on-line shopping performing both the flu-shot and camera tasks. There were substantial differences in how each group, with expertise in one area but not the other, performed the tasks. When searching in an area of expertise, the searchers tended to use more efficient, domain-specific resources and procedures, e.g., a site devoted to selling items of type X. When searching in an area outside their expertise they used more general-pupose methods (e.g. a general search engine to find a site for buying an X)

- Belkin et al. [3] looked at the role of increasing query length to see if it had any impact in task performance and/or interaction. Thirty-four subjects searched in one of two conditions: a "box" query input mode and a "line" query input mode. One-half of the subjects were instructed to enter their queries as complete sentences or questions; the other half as lists of words or phrases. The results showed that queries entered as questions or statements were longer than those entered as words or phrases (twice as long), that there was no difference in query length between the box and line modes, and that longer queries led to better performance.

- Hersh et al. [4] carried out a pure observational study, with users having their choice of which search engine or other resources to use. They measured time taken for searching, the number of pages viewed, satisfaction of users, and what topics users selected for their partially-formed searches. Their results showed that all the tasks took between six to ten minutes, with the buying task taking longest, followed by the medical, project, and travel tasks. User satisfaction was generally high, and the Google search engine was by far the most common starting point.

- Craswell et al. [5] assessed whether there was any correlation between delivery (searching/presentation) mechanisms and searching tasks. Their experiment involved three user interfaces and two types of searching tasks. The interfaces included a ranked list interface, a clustering interface, and an integrated interface with ranked list, clustering structure, and Expert Links. The two searching tasks were searching for an individual document and for a set of documents. Their results showed that subjects usually used only one interface regardless of the searching task. No delivery mechanism was found to be superior to any other for any particular task. The only difference noted was the time used to complete a search, which was less for the ranked list interface.

- White et al. [6] examined whether implicit feedback (where the system attempts to estimate what the user may be interested in) could act as a substitute for explicit feedback (where searchers explicitly mark documents relevant). They hypothesized that implicit and explicit feedback were interchangeable as sources of relevance information for relevance feedback, comparing the two approaches in terms of search effectiveness. No significant difference between the two approaches was found.

References

1. E.G. Toms, R.W. Kopak, J. Bartlett, L. Freund, Selecting Versus Describing: A Preliminary Analysis of the Efficacy of Categories in Exploring the Web, Proceedings of the Tenth Text REtrieval Conference (TREC 2001), in press.
2. S. K. Bhavhani, Important Cognitive Components of Domain-Specific Search Knowledge, Proceedings of the Tenth Text REtrieval Conference (TREC 2001), in press.
3. N.J. Belkin, C. Cool, J. Jeng, A. Keller, D. Kelly, J. Kim, H-J Lee, M-C Tang, X-J Yuan, Rutgers' TREC 2001 Interactive Track Experience, Proceedings of the Tenth Text REtrieval Conference (TREC 2001), in press.
4. W. Hersh, L. Sacherek, D. Olson, Observations of Searchers: OHSU TREC 2001 Interactive Track, Proceedings of the Tenth Text REtrieval Conference (TREC 2001), in press.
5. N. Craswell, D. Hawking, R. Wilkinson, M. Wu, TREC10 Web and Interactive Tracks at CSIRO, Proceedings of the Tenth Text REtrieval Conference (TREC 2001), in press.
6. R.W. White, J.M. Jose, I. Ruthven, Comparing Explicit and Implicit Feedback Techniques for Web Retrieval: TREC-10 Interactive Track Report, Proceedings of the Tenth Text REtrieval Conference (TREC 2001), in press.

Overview of the TREC 2001 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. In its third year, the track continued to focus on retrieving small snippets of text that contain an answer to a question. However, several new conditions were added to increase the realism, and the difficulty, of the task. In the main task, questions were no longer guaranteed to have an answer in the collection; systems returned a response of 'NIL' to indicate their belief that no answer was present. In the new list task, systems assembled a set of instances as the response for a question, requiring the ability to distinguish among instances found in multiple documents. Another new task, the context task, required systems to track discourse objects through a series of questions.

The TREC 2001 question answering (QA) track was the third running of a QA track in TREC. The goal of the track has remained the same each year: to foster research on systems that retrieve answers rather than documents in response to a question, with a particular emphasis on systems that can function in unrestricted domains. Systems are given a large corpus of newspaper and newswire articles and a set of closed-class questions such as *Who invented the paper clip?*. They return a short ($\leq$ 50 bytes) text snippet and a document as a response to a question, where the snippet contains an answer to the question and the document supports that answer.

While the overall goal has remained the same in each running of the track, this year's track introduced new conditions to increase the realism of the task. The track included three separate tasks this year, the main task, the list task, and the context task. The main task was essentially the same as the task in previous years except questions were no longer guaranteed to have an answer in the collection. In the list task, systems assembled a set of instances as the response for a question, requiring the ability to distinguish among instances found in multiple documents. The context task required systems to track discourse objects through a series of questions.

This paper gives an overview of the TREC 2001 track. The next section provides the background that is common to all of this year's tasks. The following three sections then describe each of this year's tasks in turn. The final section discusses the future of the QA track.

1 Background

1.1 History of the TREC QA track

The QA track was started in 1999 (TREC-8) to support research on methods that would allow systems to move away from *document* retrieval toward *information* retrieval. An additional goal of the track was to define a task that would appeal to both the document retrieval and information extraction communities. Information extraction (IE) systems, such as those used in the Message Understanding Conferences (MUCs, see `http://www.itl.nist.gov/iad/894.02/related_projects/muc`), recognize particular kinds of entities and relationships among those entities in running text. Since answers to closed-class questions are generally entities of the types recognized by IE systems, the QA task was limited to answering closed-class questions. There were no restrictions on the domain the questions could be drawn from, however, and the data source was a large collection of free-text documents.

TREC QA track participants were given the document collection and a test set of questions. The questions were generally fact-based, short-answer 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. Participants returned a ranked list of five [*document-id*, *answer-string*] pairs per question such that each answer string was believed to contain an answer to the question. Answer strings were limited to either 50 bytes or 250 bytes depending on the run type, and could either be extracted from the corresponding document or automatically generated from information contained in the document. 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.

Not surprisingly, allowing 250 bytes in a response was an easier task than limiting responses to 50 bytes: for every organization that submitted runs of both lengths, the 250 byte limit run had a higher mean reciprocal rank. In the 50 byte limit case, the best performing systems were able to answer about 70 % of the questions in TREC-8 and about 65 % of the questions in TREC-9. While the 65 % score was a slightly worse result than the TREC-8 scores in absolute terms, it represented a very significant improvement in question answering systems. The TREC-9 task was considerably harder than the TREC-8 task because TREC-9 used actual users' questions while TREC-8 used questions constructed specifically for the track.

Most participants used a version of the following general approach to the question answering problem. The system first attempted to classify a question according to the type of its answer as suggested by its question word. For example, a question that begins with "who" implies a person or an organization is being sought, and a question beginning with "when" implies a time designation is needed. Next, the system retrieved a small portion of the document collection using standard text retrieval technology and the question as the query. The system performed a shallow parse of the returned documents to detect entities of the same type as the answer. If an entity of the required type was found sufficiently close to the question's words, the system returned that entity as the response. If no appropriate answer type was found, the system fell back to best-matching-passage techniques. Improvements in TREC-9 systems generally resulted from doing a better job of classifying questions as to the expected answer type, and using a wider variety of methods for finding the entailed answer types in retrieved passages.

In October 2000, the DARPA TIDES project released a white paper that included a roadmap for question answering research [3]. The paper described an ambitious program to increase the complexity of the types of questions that can be answered, the diversity of sources from which the answers can be drawn, and the means by which answers are displayed. It also included a five year plan for introducing aspects of these research areas into the TREC QA track, with the TREC 2001 track as the first year of the plan. The two new requirements suggested by the roadmap for TREC 2001 were including questions whose answers were scattered across multiple documents, and no longer guaranteeing an answer is present in the document collection. These new requirements were the motivation for the list task and removing the guarantee in the main task. The context task was added as a pilot study for TREC 2002 since the roadmap's new requirement for next year is question answering within a context.

1.2 Answer assessment

The TREC QA evaluations have been based on the assumption that different people will have different ideas as to what constitutes a correct answer. This assumption was demonstrated to be true during the TREC-8 evaluation. For TREC-8, each 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 had legitimate differences of opinion as to what constituted an acceptable answer even for the deliberately constrained questions used in the track. Two prime examples of where such differences arise are the completeness of names and the granularity of dates and locations.

Fortunately, as with document retrieval evaluation, the relative scores between QA systems remain stable despite differences in the judgments used to evaluate them [4]. The lack of a definitive answer key does mean that evaluation scores are only meaningful in relation to other scores on the same data set. Absolute scores *do* change if you use a different set of judges, or a different set of questions. However, this is an unavoidable characteristic of QA evaluation. Given that assessors' opinions of correctness differ, the eventual end users of the QA systems will have similar differences of opinion, and thus any evaluation of the technology must accommodate these differences.

A [*document-id*, *answer-string*] pair was judged correct if, in the opinion of the NIST 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" (except in TREC-8 where it was marked correct). 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 [5] for a very detailed discussion of responsiveness.

The basic unit of response for each of the tasks in this year's QA track was once again the [*document-id*, *answer-string*] pair, though all strings were limited to no more than 50 bytes. Response pairs were judged as described above. For the main task, each question was independently judged by two assessors despite the TREC-8 results that showed using multiple assessors per question is not necessary to get stable evaluation results. The TREC-9 judgments, which used only one assessor per question, contain a larger number of blunders (out-and-out mistakes, not differences of opinions) than anticipated. While comparative evaluation results are stable despite such errors, the judgments are also used as training data and the effect of the errors for training is not clear. To reduce the number of errors for the TREC 2001 main task, each question was judged by two assessors and differences between judges were flagged. NIST staff reviewed each flagged response to determine if the difference was a matter of opinion or a mistake. If the reviewer found the difference to be a matter of opinion, the opinion of the first judge prevailed. Otherwise, the reviewer corrected the mistake. On average, the two assessors disagreed on 5 % of the responses, and the initial judgment was changed in 30 % of the cases where there was a disagreement. As a check of the earlier TREC-8 result, we computed the correlations among the system rankings produced by evaluating the main task runs on the different judgment sets. Once again the correlations were very high (greater than 0.96), indicating that the results are stable regardless of which judgment set is used.

The QA roadmap called for another change for the TREC 2001 track that was not implemented: requiring systems to return an actual answer rather than a string that contains an answer. This change was not implemented because it is not clear how to operationalize "actual answer". Is a string wrong if it contains an answer and justification of that answer? Are titles required parts of names or extraneous information? Nonetheless, some move toward "actual answer" will be necessary for future tracks since allowing assessors to pick out the answer from a returned string is masking large differences between systems. For example, Figure 1 shows some of the answer strings that were judged correct for question 916, *What river in the US is known as the Big Muddy?*. Each of these strings should be marked correct according to the current assessment procedure, but some are much better responses than others.

1.3 The TREC 2001 track

The document set for all tasks was the set of news articles on the combined set of TIPSTER/TREC disks. In particular, this includes the AP newswire from disks 1–3, the *Wall Street Journal* from disks 1–2, the *San Jose Mercury News* from disk 3, the *Financial Times* from disk 4, the *Los Angeles Times* from disk 5, and the Foreign Broadcast Information Service (FBIS) from disk 5. This set contains approximately 979,000 articles in 3,033 megabytes of text, and covers a broad spectrum of subjects.

As a service to the track, 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

```
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?*

Table 1: Number of runs per task (Main, List, Context) submitted by TREC 2001 QA track participants.

Organization	M	L	C	Organization	M	L	C
Alicante University	2	–	–	National Taiwan University	2	2	1
Chinese Academy of Sciences	3	–	–	NTT Communication Sci. Labs	1	–	–
CL Research	2	2	1	Oracle	1	–	–
Conexor Oy	1	–	–	Pohang U. of Sci. & Tech.	1	–	–
EC Wise, Inc.	2	–	–	Queens College, CUNY	3	2	2
Fudan University	1	–	–	Sun Microsystems Labs	2	–	–
Harbin Institute of Technology	1	–	–	Syracuse University	2	2	–
IBM (Ittycheriah)	3	–	–	Tilburg University	2	–	–
IBM (Prager)	3	–	–	Universite de Montreal	3	2	–
InsightSoft-M	1	–	–	University of Alberta	1	–	–
ITC-irst	1	–	–	University of Amsterdam	3	2	–
KAIST	2	2	1	U. Illinois, Urbana/Champaign	1	–	–
KCSL	1	–	–	University of Iowa	2	–	–
Korea University	2	–	–	University of Pennsylvania	1	–	–
Language Computer Corp.	1	1	1	University of Pisa	3	–	–
LIMSI	3	–	–	U. of Southern California, ISI	2	1	–
Microsoft Research	2	–	–	University of Waterloo	3	2	1
MITRE	1	–	–	University of York	2	–	–

given from that same ranking). For the context task, the rankings were produced for the first question in a series only. 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-six groups submitted a total of 92 runs to the QA track. Table 1 lists each participating group and the number of runs that group submitted for each task.

2 The Main Task

The main QA task was very similar to previous years' tasks, providing continuity with previous years and giving newcomers to the track a stable task with which to begin. Participants received a set of closed-class questions and searched a large document set to extract (or construct) an answer to each question. Participants returned a ranked list of five [*document-id*, *answer-string*] pairs per question such that each answer string was believed to contain an answer to the question and the document supported that answer. Answer strings were limited to no more than 50 bytes.

Questions were not guaranteed to have an answer in the document collection. Recognizing that there is no answer is a challenging task, but it is an important ability for operational systems to possess since returning an incorrect answer is usually worse than not returning an answer at all. Systems indicated their belief that there was no answer in the document collection by returning 'NIL' rather than a [*document-id*, *answer-string*] pair as one of the responses to a question. The NIL response was scored the same as other responses; NIL was correct when no correct answer was known to exist in the collection for that question. A correct answer was known to exist if the assessor found a correct answer during question development, or if some system returned a correct, *supported* response to the question. Forty-nine questions had no known correct response in the document collection.

Recognizing that no answer exists in the document collection is a different task from having the system recognize that it does not know what the answer is. This latter task is also important, but is more difficult to evaluate because systems must then be scored using a combination of questions attempted and attempted questions correctly answered. As an initial foray into evaluating whether systems can determine if they know the answer, systems were required to report a single final answer for each question in addition to the ranked list of 5 responses. The final answer was either an integer from one to five that referred to a position in the ranked list of responses for that question, or the string 'UNSURE' that indicated the system did not know what the answer was. While the vast majority of systems returned the answer at rank one if they were confident of the answer, a few systems did return an alternate rank.

2.1 Test questions

The test set of questions continued a progression of using more realistic questions in each of the three runnings of the track. In TREC-8, the majority of the questions were created expressly for the track, and thus tended to be back-formulations of a statement in a document. In TREC-9, the questions were selected from an Encarta log that contained actual questions, and a raw Excite log. Since the raw Excite log did not contain many grammatically well-formed questions, NIST staff used the Excite log as a source of ideas for actual questions. All the questions were created without looking at any documents. The resulting test set of questions was much more difficult than the TREC-8 set, mainly because the TREC-9 set contained many more high-level questions such as *Who is Colin Powell?*. For this year's main task, the source of questions was a set of filtered MSNSearch logs and AskJeeves logs. Raw logs were automatically filtered (at Microsoft and AskJeeves) to select queries that contained a question word (e.g., what, when, where, which, etc.) anywhere in the query; that began with modals or the verb to be (e.g., are, can, could, define, describe, does, do, etc.); or that ended with a question mark. NIST did additional human filtering on these logs, removing queries that were not in fact questions; questions that asked for a list of items; procedural questions; questions that asked for the location of something on the web (e.g., pictures of someone); yes/no questions; and questions that were obviously too current for the document collection (e.g., questions about Britney Spears, etc.). The assessors then searched the collection looking for answers for the queries that remained.

The final question set consisted of 500 questions. NIST fixed the spelling, punctuation, and sometimes the grammar of the queries selected to be in the final question set, but except for a very few (less than 10) questions, the content of the question was precisely what was in the log. The few changes that were made were simple changes such as substituting one Greek god for another so that the question would have an answer in the collection.

NIST has made no attempt to control the relative number of different types of questions in the test set from year to year. Instead, the distribution of question types in the final test set has reflected the distribution

Table 2: Evaluation scores for a subset of the TREC 2001 main task runs. Scores are given for the best run from the top 15 groups. Scores include the mean reciprocal rank (MRR), and number (# qs) and percentage (%) of questions for which no correct response was returned for both strict (unsupported responses counted as wrong) and lenient (unsupported responses counted as correct) evaluation. Also included are the number of questions for which the run returned 'NIL' as a response (NIL Returned), the number of questions for which 'NIL' was correctly returned as a response (NIL Correct), the percentage of questions for which the system was sure of its final answer (Final Sure), and the percentage of questions for which the final answer was correct when the system was sure (Sure Correct).

	Strict Evaluation			Lenient Evaluation			# qs	# qs		
		No Correct			No Correct		NIL	NIL	Final	Sure
Run Tag	MRR	# qs	%	MRR	# qs	%	Returned	Correct	Sure	Correct
insight	0.68	152	30.9	0.69	147	29.9	120	38	75 %	77 %
LCC1	0.57	171	34.8	0.59	159	32.3	41	31	100 %	51 %
orcl1	0.48	193	39.2	0.49	184	37.4	82	35	100 %	40 %
isi1a50	0.43	205	41.7	0.45	196	39.8	407	33	80 %	38 %
uwmta1	0.43	212	43.1	0.46	200	40.7	492	49	100 %	35 %
mtsuna0	0.41	220	44.7	0.42	213	43.3	492	49	100 %	32 %
ibmsqa01a	0.39	218	44.3	0.40	212	43.1	192	28	100 %	30 %
IBMKS1M3	0.36	220	44.7	0.36	211	42.9	206	27	100 %	24 %
askmsr	0.35	242	49.2	0.43	197	40.0	491	49	100 %	27 %
pir1Qqa3	0.33	264	53.7	0.33	260	52.8	5	0	100 %	24 %
posqa10a	0.32	276	56.1	0.34	260	52.8	13	3	100 %	24 %
ALIC01M2	0.30	297	60.4	0.31	293	59.6	4	0	100 %	23 %
gazoo	0.30	304	61.8	0.31	300	61.0	11	0	100 %	24 %
kuqa1	0.29	298	60.6	0.30	295	60.0	6	0	100 %	23 %
prun001	0.27	333	67.7	0.27	332	67.5	201	38	100 %	24 %

in the source of questions. This year, the number of questions that asked for a definition was dramatically greater than in previous years. (Ken Litkowski of CL Research puts the count at 135/500 definition questions for TREC 2001 compared to 31/500 for TREC-9.) While a large fraction of definition questions is "real" in that the filtered MSNSearch and AskJeeves logs contain many definition questions, there are easier ways to find the definitions of terms than searching for a concise definition in a corpus of news articles. NIST will need to exert more control over the distribution of question types in future tracks.

Eight questions were removed from the evaluation, mostly due to spelling mistakes in the question. A ninth question, question 1070, also contains a typo, spelling 'Louvre' as 'Lourve'. However, that mistake was not noted until all results had been evaluated, so it remains in the test set.

2.2 Retrieval results

Table 2 gives evaluation results for the top fifteen groups. Only one run per group is included in the table. The table gives the mean reciprocal rank (MRR) scores, and the number and percentage of questions for which no correct response was returned for both strict (unsupported responses counted as wrong) and lenient (unsupported responses counted as correct) evaluation. The table also gives statistics regarding NIL and final answer processing.

Detecting whether or not an answer exists in the collection is feasible—the LCC1 run had an accuracy of 31/41 or 0.76—but apparently difficult—only five runs had an accuracy greater than 0.25 (see Table 3). (Accuracy is computed as the number of questions for which NIL was correctly returned divided by the total number of questions for which NIL was returned.) Since systems could return a ranked list of up to five responses per question, some systems returned NIL as one of the responses for every question. This resulted in an accuracy of only 0.1 (49/492), but tended to increase the overall MRR score of those systems somewhat since it is relatively rare to get the first correct response at large ranks when there is an answer

Table 3: Main task runs that had an accuracy greater than 0.25 in detecting when no answer was present in the collection. Returning NIL for all questions produced an accuracy of 0.1.

Run Tag	Returned	Correct	Accuracy
LCC1	41	31	0.76
orcl1	82	35	0.43
insight	120	38	0.37
ICTQA10a	35	10	0.29
ICTQA10b	55	15	0.27

in the collection. See the MultiText project's paper in this proceedings for an analysis of this effect [1].

In final answer processing the system was to indicate whether it was confident in the answer it produced or it recognized that it did not know what the answer was. The purpose of final answer processing in TREC 2001 was to gather data for investigating evaluation strategies for systems that can return "I don't know" as a response. Unfortunately, not enough data was collected to analyze new strategies since more than half of the runs were always confident in their response (see the last two columns of Table 2).

Almost all systems used variants of the strategy seen in earlier TRECs to perform the main task: determine the answer type from the form of the question; retrieve a small portion of the document set; and find the correct answer type in a document piece. Most systems used a lexicon, usually WordNet [2], to verify that a candidate response was of the correct type. Some systems also used the lexicon as a source of definitions.

While most systems used the same basic strategy, there was much less agreement on the best approaches to realizing that strategy. Many groups continued to build systems that attempt a full understanding of the question, but increasingly many groups took a more shallow, data-driven approach. The data-driven approaches rely on simpler pattern matching methods using very large corpora (frequently the web) rather than sophisticated language processing. The idea exploited in the massive data approach is the fact that in a large enough data source a correct answer will usually be repeated often enough to distinguish it from the noise that happens to occasionally match simple patterns.

A second area in which there is no consensus as to the best approach is classification schemes for answer types. Some systems use a few very broad classes of answer types, while others use many specialized classes. The difference is a standard trade-off between coverage and accuracy. With many specialized answer types, finding the actual answer once an answer type is correctly classified is much easier because of the specificity of the class. However, deciding which class is correct, and ensuring there is a class for all questions, is much more difficult with many specialized classes. Some systems use a hierarchical answer typology to exploit this trade-off.

3 The List Task

As mentioned above, one of the goals for the TREC 2001 QA track was to require 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.

The list task accomplished this goal. 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 2. Each instance was guaranteed to obey the same constraints as an individual answer in the main task and was judged in the same manner as a response in the main task: each true instance was no more than 50 characters long; some document was explicit that it was an instance of the desired type; each answer string had to have an associated document that supported the answer; answer strings had to be responsive; etc. The document collection was guaranteed to contain at least the target number of instances. 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 and NIST staff since

- Name 4 U.S. cities that have a "Shubert" theater.
- Name 30 individuals who served as a cabinet officer under Ronald Reagan.
- Who are 6 actors who have played Tevye in "Fiddler on the Roof"?
- Name 4 countries that can produce synthetic diamonds.
- What are 9 novels written by John Updike?

Figure 2: Example list task questions.

Table 4: Average accuracy of the TREC 2001 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
LCC2	0.76	UdeMlistP	0.15
isi1l50	0.45	qntual2	0.14
pir1Qli1	0.34	UAmsT10qaL2	0.13
SUT10PARLT	0.33	clr01l1	0.13
SUT10DOCLT	0.25	UAmsT10qaL1	0.12
uwmtal1	0.25	clr01l2	0.12
uwmtal0	0.23	KAISTQALIST1	0.08
pir1Qli2	0.20	KAISTQALIST2	0.07
qntual1	0.18	UdeMlistB	0.07

there were not enough appropriate questions in the logs. 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 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 correct, incorrect, or not supported were made individually for each [*document-id*, *answer-string*] pair. 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. Incorrect responses were always marked as not distinct (the assessment software enforced this), but unsupported responses could be marked distinct.

Since answer strings could be up to fifty bytes long, a single string might contain more than one instance. The track guidelines specified that the left-most instance in a string was always counted as the instance of record for that string. For example for the question *Name 9 countries that import Cuban sugar.*, the string `China and Russia imported Cuban sugar` was counted as an instance of China only. If another answer string in the list was `China imports`, one of the two responses would be marked as distinct for China, and Russia still would not be counted as a retrieved instance.

List results were evaluated using accuracy, the number of distinct responses divided by the target number of instances. Note that since unsupported responses could be marked distinct, the reported accuracy is a lenient evaluation. Table 4 gives the average accuracy scores for all of the list task submissions.

Given the way the questions were constructed for the list task, the list task questions were intrinsically easier than the questions in the main task. Most systems found at at least one instance for most questions. Each system returned some duplicate responses, but duplication was not a major source of error for any of the runs. (Each run contained many more wrong responses than duplicate responses.) With just 18 runs, there is not enough data to know if the lack of duplication is because the systems are good at recognizing and eliminating duplicate responses, or if there simply wasn't all that much duplication in the document set.

4 The Context Task

The context task was intended to test the systems' ability to track discourse objects (context) through a series of questions. Eventual users of QA systems will likely interact with the system on a regular basis, and the user will expect the system to have some basic understanding of previous interactions. The TREC 2001 context task was designed to represent the kind of dialog processing that a system would require to support an interactive user session.

The type of questions that were used in the context task was the same as in the main task. However, the questions were grouped into different series, and the QA system was expected to track the discourse objects across the individual questions of a series. That is, the interpretation of a question later in the series could depend on the meaning or answer of an earlier question in the series. Correct interpretation of a question often involved resolving referential links within and across questions. Figure 3 gives three examples of series used in the context task.

NIST staff created ten question series for the context task. Most series contained three or four questions, though one series contained nine questions. There were 42 questions across all series, and each question was guaranteed to have an answer in the document collection. A context task run consisted of a ranked list of up to five [*document-id*, *answer-string*] pairs per question as in the main task. The context task questions were judged and evaluated as in the main task as well, except that only lenient evaluation scores (unsupported as correct) were computed. All questions were judged by the same assessor.

Seven runs were submitted to the context task, with unexpected results. The ability to correctly answer questions later in a series was uncorrelated with the ability to correctly answer questions earlier in the series. The first question in a series defined a small enough subset of documents that results were dominated by whether the system could answer the particular type of the current question, rather than by the systems' ability to track context. Thus, this task is not a suitable methodology for evaluating context-sensitive processing for the current state-of-the-art in question answering.

5 Future

The TREC QA track will continue, with the selection of tasks included in future tracks influenced by both the QA roadmap and the ARDA AQUAINT program (see `http://www.ic-arda.org/InfoExploit/aquaint/index.html`). The goal of future tracks is to increase the kinds and difficulty of the questions that systems can answer.

The main task in TREC 2002 will focus on having systems retrieve the *exact* answer as opposed to text snippets that contain the answer. While this will entail marking as incorrect "good" responses such as an answer plus justification, we believe that forcing systems to be precise will ultimately produce better QA technology. Systems will be allowed to return only one response per question, another change aimed at forcing systems to be more precise. NIST will exert more control over the relative proportions of different kinds of questions in the test set. In particular, definitional questions will be a very small percentage of the total question set.

The list task will be repeated in essentially the same form as TREC 2001. NIST will attempt to find naturally occurring list questions in logs, but appropriate questions are rare, so some constructed questions may also be used. We hope also to have a new context task, though the exact nature of that task is still undefined.

The main focus of the ARDA AQUAINT program is to move beyond the simple factoid questions that have been the focus of the TREC tracks. Of particular concern for evaluation is how to score responses that cannot be marked simply correct/incorrect, but instead need to incorporate a fine-grained measure of the quality of the response. We expect that NIST and the AQUAINT contractors will run pilot studies to

CTX1a Which museum in Florence was damaged by a major bomb explosion in 1993?

CTX1b On what day did this happen?

CTX1c Which galleries were involved?

CTX1d How many people were killed?

CTX1e Where were these people located?

CTX1f How much explosive was used?

CTX3a What grape variety is used in Chateau Petrus Bordeaux?

CTX3b How much did the futures cost for the 1989 vintage?

CTX3c Where did the winery's owner go to college?

CTX3d What California winery does he own?

CTX10a How many species of spiders are there?

CTX10b How many are poisonous to humans?

CTX10c What percentage of spider bites in the U.S. are fatal?

Figure 3: Example question series for the context task.

experiment with different measures in the first year of AQUAINT (2002). Promising measures will then be put to a broader test by being incorporated into later TREC tracks.

References

[1] C.L.A. Clarke, G.V. Cormack, T.R. Lynam, C.M. Li, and G.L. McLearn. Web reinforced question answering (MultText experiments for TREC 2001). In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Tenth Text REtreival Conference (TREC 2001)*, 2002.

[2] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database.* The MIT Press, 1998.

[3] Sanda Harabagiu, John Burger, Claire Cardie, Vinay Chaudhri, Robert Gaizauskas, David Israel, Christian Jacquemin, Chin-Yew Lin, Steve Maiorano, George Miller, Dan Moldovan, Bill Ogden, John Prager, Ellen Riloff, Amit Singhal, Rohini Shrihari, Tomek Strzalkowski, Ellen Voorhees, and Ralph Weishedel. Issues, tasks, and program structures to roadmap research in question & answering (Q&A), October 2000. `http://www-nlpir.nist.gov/projects/duc/roadmapping.html`.

[4] 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.

[5] 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`.

The TREC-2001 Video Track Report

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

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

April 18, 2002

1 Introduction

New in TREC-2001 was 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 built on publicly available video provided by the Open Video Project of the University of North Carolina at Chapel Hill under Gary Marchionini (Marchionini, 2001), the NIST Digital Video Library (Over, 2001), and stock shot video provided for TREC-2001 by the British Broadcasting Corporation (Richard Wright et al). The track used very nice work on shot boundary evaluation done as part of the ISIS Coordinated Research Project (AIM, 2001).

This paper is an introduction to the track framework — the tasks, data, and measures. For information about results, see the tables associated with the conference proceedings.

TREC research has remained true to its late twentieth century origins, concentrating on retrieval of text documents with only occasional excursions into other media: spoken documents and images of documents. Using TREC as an incubator, the Video Track has pushed into true multimedia territory with respect to formulation of search requests, analysis of multimedia material to be searched (video, audio, transcripts, text in video, music, natural sound, etc), combination of search strategies, and in some cases presentation of results to a human searcher.

The TREC video track had 12 participating groups, 5 from US, 2 from Asia and 5 from Europe. 11 hours of MPEG-1 data was collected and distributed as well as 74 topics or queries. What made these queries particularly interesting and challenging was that they were true multimedia queries as they all had video clips, images, or audio clips as part of the query, in addition to a text description. Participating groups used a variety of techniques to match these multimedia queries against the video dataset, some running fully automated techniques and others involving users in interactive search experiments.

As might be expected for the first running of such a track, the framework was a bit unorthodox by the standards of mature TREC tracks. Participating groups contributed significant amounts of work toward the creation of the track infrastructure. Search systems were called upon to handle a very wide variety of topic types. We hoped exploring more of the possible territory, though it decreased the likelihood of definitive outcomes in any one area this year, would still generate some interesting results and more importantly provide a good foundation for a more focused track in TREC-2002.

In TREC-2001, participating groups were invited to test their systems one or more of the following three tasks/evaluations.

- Shot boundary detection
- Search (fully automatic or interactive)
  - Using known-item topics or queries
  - Using general topics or queries

See the "Approaches" section for a list of the 12 participating groups and information on their systems. Details about each task follow here.

2 Shot boundary detection

Movies on film stock are composed of a series of still pictures (frames) which, when projected, the human brain smears together so we see 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. Software has been developed and evaluations of various methods against the same test collection have been published e.g., using 33 minutes total from 5 feature films (Aigrain & Joly, 1994); 3.8 hrs 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 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 8 television news, advertising, and series videos (Ruiloba, Joly, Marchand-Maillet, & Quénot, 1999).

2.1 Data

The shot boundary test collection for this year's TREC task comprises about half the videos in the overall collection so that each series is represented. The videos are mostly of a documentary nature but vary in their age, production style, and quality. There are 42 videos encoded in MPEG-1 with a total runtime of about 5.8 hours and a total size of 3.34 gigabytes.

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 first of next shot, no fade or combination

dissolve - the first shot fades out *while* the second fades in

fadeout/in - the first shot fades out, *then* the second fades in

other - everything not in the previous categories

The VirtualDub software (Lee, 2001) was used in the Microsoft Windows environment to view the videos and frame numbers. The VirtualDub website contains information about VirtualDub and the MPEG decoder it uses. Twenty of the videos (from the BBC stock shot collection) had no internal transitions and thus no shot boundaries. The collection used for evaluation of shot boundary determination contains 594179 frames and 3176 transitions with the following breakdown as to type (using the post-conference corrected reference data):

- 2066 — hard cuts (65%)
- 975 — dissolves (30.7%)
- 54 — fades to black and back (1.7%)
- 81 — other (2.6%)

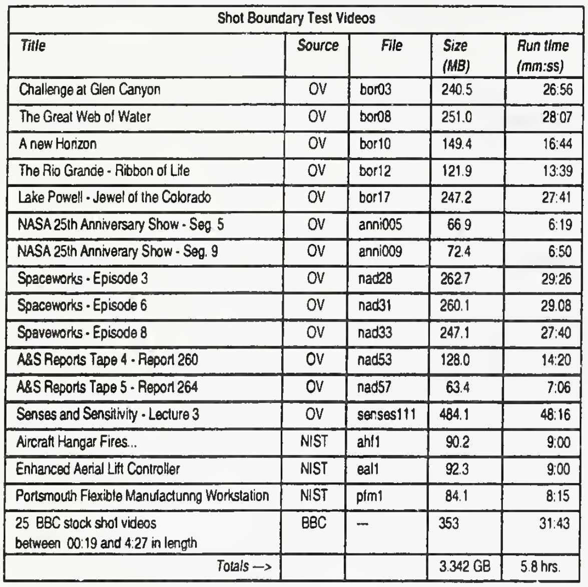
The proportion of gradual transitions is about twice that reported by Boreczky and Rowe (1996) and Ford (1999). Gradual transitions are generally harder to recognize than abrupt ones. Table 1 lists the videos with title, source collection, file name, size in megabytes, and run time (mm:ss). Note that the reference data for the video "A new Horizon" (bor10) turned out to have been inadvertently truncated. Consequently, no results for it were ready until immediately after the TREC-2001 workshop.

2.2 Evaluation

Submissions were compared to the shot boundary reference data using a modified version of the protocol proposed for 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. The version used in TREC has the following features:

- A short gradual transition (less than 6 frames) was treated as a cut

Table 1: Shot Boundary Determination Test Collection



Shot Boundary Test Videos				
Title	*Source*	*File*	*Size (MB)*	*Run time (mm:ss)*
Challenge at Glen Canyon	OV	bor03	240.5	26:56
The Great Web of Water	OV	bor08	251.0	28:07
A new Horizon	OV	bor10	149.4	16:44
The Rio Grande - Ribbon of Life	OV	bor12	121.9	13:39
Lake Powell - Jewel of the Colorado	OV	bor17	247.2	27:41
NASA 25th Anniversary Show - Seg. 5	OV	anni005	66.9	6:19
NASA 25th Anniverary Show - Seg. 9	OV	anni009	72.4	6:50
Spaceworks - Episode 3	OV	nad28	262.7	29:26
Spaceworks - Episode 6	OV	nad31	260.1	29.08
Spaveworks - Episode 8	OV	nad33	247.1	27:40
A&S Reports Tape 4 - Report 260	OV	nad53	128.0	14:20
A&S Reports Tape 5 - Report 264	OV	nad57	63.4	7:06
Senses and Sensitivity - Lecture 3	OV	senses111	484.1	48:16
Aircraft Hangar Fires...	NIST	ahf1	90.2	9:00
Enhanced Aerial Lift Controller	NIST	eal1	92.3	9:00
Portsmouth Flexible Manufactunng Workstation	NIST	pfm1	84.1	8:15
25 BBC stock shot videos between 00:19 and 4:27 in length	BBC	---	353	31:43
Totals —>			3.342 GB	5.8 hrs.

- A submitted cut matched a reference cut if the latter fell entirely within the boundaries of the former after the former has been extended 5 frames on each end.
- Gradual transitions matched if the intersection was at least 0.333 of the longer and 0.499 of the shorter transition — the default values from the earlier ISIS evaluation scheme.

For the purposes of evaluation, the categories were divided into two:

- cuts - cuts
- graduals - dissolves, fades to black and back, and other

2.3 Measures

For continuity with earlier work, the following measures were calculated by NIST: inserted transition count, deleted transition count, correction rate, deletion rate, insertion rate, error rate, quality index, correction probability, recall, and precision. See Ruiloba et al. (1999) for details on the definitions of these measures.

2.4 Issues/Lessons

There were several unexpected issues that cropped up during the running and subsequent evaluation of the shot boundary determination task.

Varying frame numbering

Different MPEG-1 decoders produced slightly different frame numbering from the same video source file. This caused problems for evaluation of cuts since, initially, exact matches were required. A fixed shift of plus or minus 2 and then plus or minus 5 for an entire file was used until evidence was found that in some cases the shift of frame numbers varied within a file. The solution to this problem was eventually the algorithm described above, immediately under "Evaluation". The TREC video mailing list was quite active on this point and contributed to addressing the problem. The applicability of the 11-frame window to new data, is unknown and as an alternative for the future, a standard decoder or set of decoders could be mandated for determining frame numbers in the submission. Workshop participants generally felt this would be impractical for them.

Test collection available in advance

Although they did not know specifcially which files would be used, the shot boundary test collection was available to the participating groups long before the test began. Groups were reminded that systems to be tested could not have been trained on any of the test collection files — standard research practice anyway. It would however be preferable in future to use test video not generally available before the test.

Single reference

A second reference set was started but could not be completed in time. Finishing it would allow one to gauge the variability in system evaluation due to inter-annotator disagreements. For the final results we did check the shot boundary reference in cases where more than a couple systems told us there was a transition we did not have. This resulted in the addition of 20 transitions. We also completed the reference for the bor10.mpg file which had been inadvertently truncated.

3 The Search Tasks

The search tasks in the Video Track were extensions of their text-only analogues. The systems, some 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 shots from the videos in the test collection which met the need.

In the case of the Video Track, the topics contained not only text but possibly examples (including video, audio, images) of what is needed. The topics expressed a very wide variety of needs for video clips: of a particular object or class of objects, of an activity/event or class of activities/events, of a particular person, of a kind of landscape, on a particular subject, using a particular camera technique, answering a factual question, etc. See Table 3 for an overview of the topics and their makeup.

The boundaries for units of retrieval to be identified - shots - were not predefined for all systems and each system made its own independent judgment of what frame sequences constituted a relevant shot. This had important consequences for evaluation.

The evaluation of video retrieval, whether for known-items or general searching, presents a larger, if not harder, set of problems than evaluations of text-only retrieval and we are not aware of any other large, open evaluation of content-based retrieval from digital video. Wide-spread use of video data, when it exists, is often limited by cost and intellectual property rights. Details about each of the tasks follow.

Although the track decided early on that it should work with more than text from audio, systems were allowed to use transcripts created by automatic speech recognition (ASR). Any group which did this had to submit a run without the ASR or one using only ASR — as a baseline. At least two groups used ASR.

3.1 Data to be searched

The test collection for the search task consisted of the collection used for the shot boundary determination task plus another six or so hours of similar video as listed in Table 2. The only manually created information that search systems were allowed to use was that which was already as part of the test collection, namely: the existing transcripts associated with the NIST files and the existing descriptions associated with the BBC material.

3.2 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 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.

Table 2: Additional video to be searched

Additional test videos				
Title	*Source*	*File*	*Size (MB)*	*Run time (mm:ss)*
The Coloredo	OV	bor02	178.3	19.58
The Story ot Hoover Dam	OV	bor07	246.1	27:24
Wetlands Regeined	OV	bor09	126.5	14:01
Gient on the Bighorn	OV	bor11	125.4	14:03
Take Pride in America	OV	bor14	103.0	11:32
How Weter Won the West	OV	bor19	100.8	11.17
NASA 25th Anniverery Show - Seg. 6	OV	enni006	97.6	9:13
NASA 25th Anniverary Show - Seg. 10	OV	enni010	184.8	17:27
Spaceworks - Episode 5	OV	nad30	266.1	29:48
Spaceworks - Episode 7e	OV	nad32	259.3	29.03
A&S Reporte Tepe 4 - Report 259	OV	nad52	129.7	14:31
A&S Reports Tape 4 - Report 262	OV	nad55	131.2	14:41
A&S Reporte Tape 5 - Report 265	OV	nad58	68.8	7:42
Senses end Sensitivity - Lecture 4	OV	senses114	486.4	48:30
Telepresence Miscoscopy	NIST	dbe1	94.3	12:30
NIST in 5 Minutes end 41 Seconds	NIST	n5m1	65.9	5:41
A Decade of Bueiness Excellence for America	NIST	ure1	85.1	8.50
A Uniquely Rewarding Experience	NIST	ydh1	128.1	12:23
25 BBC stock shot videos between 00:11 end 3:40 in length	BBC	---	301.8	27:08
Totals --->			2.96 GB	5.4 hrs.

For a number of practical reasons, the topics were created by the participants. This was not an easy or quick process. Each group was asked to formulate five or more topics they could imagine being used by someone searching a large video archive. Twelve sets of topics were submitted. NIST submitted topics as well, did some selection, and negotiated revisions. All the topics were pooled and all systems were expected to run on the union, if at all possible. The worst-case scenario in which each group found it's topics too easy and everyone else's topics too hard to learn something did not occur. Several groups found their own topics quite challenging and most groups had some success with topics other than their own.

All topics contained a text description of the user information need. Examples in other media were optional. There were indicators of the appropriate processing. And finally, if the need was conceived as a hunt for one or more known-items, then the list of known-items was included. Here is a summary of the topic layout:

- Text description of the information need
- Examples of what is needed
  - video clip illustrating what is needed

Table 3: Overview of topics

Topic #	Inter-active	Auto-metic	Text deecription of needed Information/ehot	Number of exemplee			Known Iteme
				Video	Imege	Audio	
1	Y		number of spikas on Statua of Liberty's crown		1		10
2		Y	liftoff of the Space Shuttle	4			
3		Y	vehicla traveling on the moon	1			2
4		Y	mountaina es prominent scenery	1			8
5		Y	water skiing	1			5
6		Y	acanas with e yellow boet	1			4
7		Y	pink flower		1		1
8	Y	Y	the plenet Jupitar		2		6
9		Y	people who ere water skiing	1			
10		Y	swimming pools	1			
11		Y	paople on the beech	1			
12		Y	surfece of Mers	1		1	4
13		Y	spaaker talking in front of the US flag	2		2	2
14	Y	Y	astroneut driving luner rover over lunar surfece	2			5
15	Y	Y	corn on tha cob		1		4
16	Y	Y	deer with its entlers	1	1		2
17	Y	Y	eirlinar lending		1		3
18	Y	Y	John Deera tractor		2		2
19	Y	Y	luner rovar from Apollo missions		2		5
20	Y	Y	picturas of Ron Vaughn, Preaidant of Vaughncraft				1
21	Y	Y	picturaa of Roneld Reegen apaeking		3	3	1
22	Y	Y	picturas of Herry Hertz		2		5
23	Y	Y	imegea of Lou Gossett, Jr.		3		2
24	Y	Y	ell other picturas of R. Lynn Bondarant	1			
25	Y	Y	scena from Ster-Wers with R2D2 end 3CPO		2		1
26	Y	Y	givan summary, find the full scene sequence	1			1
27	Y	Y	biplane flying over a field	1	1		4
28	Y	Y	selling boat on e beech		1		2
29	Y	Y	hot eir belloon in the sky		1		5
30	Y	Y	governmentel buildings looking like Cepitol	1			4
31	Y	Y	weterskier behind e speed boat		2		7
32	Y	Y	chopper lending			3	1
33	Y	Y	additional shots of white fort	1			1
34	Y	Y	Ronald Raegen raeding speech about Spaca Shuttla		1		1
35	Y	Y	Where else doaa this peraon appeer?	1			11
36	Y	Y	Where elsa doas thia person appear?	1			7
37	Y		other exemplas of rockat and shuttle launchas	7		7	
38	Y		othar axamples of fires	4			
39	Y		other exemples of eirplanaa taking off	3		3	
40		Y	ell monologua shota	2			
41		Y	ell ahota with et leest 8 paopla	2			
42		Y	ell shots with Devid J. Neah	1			
43		Y	ell shots with e specific landacape: graaaland	1			
44		Y	all shots with specific camere technique: pen & tilt	1			
45		Y	othar shota of cityscepes	1			
46		Y	othar shota of selling boets	1			
47		Y	clips thet deel with floods	1			
48		Y	ovarheed zooming-in views of cenyons...	8			
49		Y	other clips from tha lacture ahowing/expleining axampla grephic	9			
50		Y	other axemples of neturel outdoors scenes with birds	8		10	
51		Y	other examplea of aplaahing water in neturel outdoors environmant	7		10	
52	Y	Y	spece shuttle on leunch ped	6	2		
53	Y	Y	picturas of the Parsaus high altitude piena		3		
54	Y	Y	clips showing Glan Canyon dam	1			
55	Y	Y	picturas of Hoovar Dam	1			
56	Y	Y	clips of rockets taking off	2			
57	Y	Y	footaga of axplosiona, blasting of hillsidaa	1			
58	Y	Y	edditional shota of Lynn Bonderent	1			
59	Y	Y	launch of the Spece Shuttle	3	1		
60	Y	Y	axplosions in progress		1		60
61	Y	Y	environmentel degredetion	3	1	1	
62	Y	Y	how long hes Beldrige Awerd existad				3
63		Y	clips of different intervlawees	7			
64		Y	clips of different mele interviewees	4		3	
65		Y	greduel shot chenges	1			
66	Y	Y	clips telking ebout watar projects	1			
67	Y	Y	segments of eircraft X-29	2	5		10
68	Y	Y	segment with e(n expert) person showing the X-29	2	5		1
69	Y	Y	logo of Northwest Airlinas		5		2
70	Y	Y	identify the producer of each item				3
71	Y	Y	ecenee with etroet traffic (care, trucke, maybe people)		1		18
72	Y	Y	other similar clips conteining a rocket launch	2			
73		Y	all shots with a specific landscepe: leke	2			
74		Y	ell shota with specific camara technique: zoom	1			

- still image illustrating what is needed
    - audio illustrating what is needed
- Processing recommendations
    - indication of whether topic is for interactive processing
    - indication of whether topic is for automatic processing
- list of known-items, if any defined

If examples to illustrate the information need were included then these were to come from outside the test data. They could be taken from NIST or Open-Video material not part of the test collection or from other public domain sources. If the example came from the test collection, the topic's text description was to be such that using a video quotation from the test collection is plausible, e.g., "I want to find all the OTHER shots dealing with X." A search for a single shot could not be described with example video or images from the target shot.

3.3 Evaluation of known-item searches

The known-item search submissions were evaluated by NIST using a variation of the algorithm used in the shot boundary determination task. Matching a submitted item to a known-item defined with the topic was a function of the length of the known-item, the length of the submitted item, the length of the intersection, and two variables:

- KI coverage: minimum value for the ratio of the length of the intersection to the length of the known-item, i.e., how much of the known-item was captured by the submitted item
- RI coverage: minimum value for the ratio of the length of the intersection to length of the submitted result item, i.e., how much of the submitted result item was on target

The evaluation was run with four different settings of the two variables — as examples. In the absence of an application, a choice of particular settings would be arbitrary. The four settings reported to participants were the four combinations of 0.333 and 0.666. The pages at the back of the TREC-2001 proceedings report results where the length of the intersection must be at least 0.666 of the length of the known-item and at least 0.333 of the submitted item.

The performance of systems/runs can't be compared directly since they attempt different subsets of topics and may or may not include a human in the loop though we are dealing with rather small differences. It may be worth noting that the ranking of the systems/runs based on these values appear to be fairly stable across different match parameter settings as measured by Kendall's tau (see Table 4).

Table 4: Stability of known-item search system rankings as match parameter settings vary

Kendall's tau for recall-ranked systems by matching-parameter settings

KI,RI settings	0.333, 0.333	0.333, 0.666	0.666, 0.333	0.666, 0.666
0.333, 0.333		0.923	0.881	0.814
0.333, 0.666			0.838	0.876
0.666, 0.333				0.890
0.666, 0.666				

Kendall's tau for precision-ranked systems by matching-parameter settings

KI,RI settings	0.333, 0.333	0.333, 0.666	0.666, 0.333	0.666, 0.666
0.333, 0.333		0.957	0.914	0.876
0.333, 0.666			0.900	0.900
0.666, 0.333				0.942
0.666, 0.666				

3.4 Known-item measures

The measures calculated for the evaluation of known-item searching were precision and recall. It should be noted that a result set item could match more than one known-item and a known-item could match more than one result set item. In calculating precision, credit was given if a result set item matched at least one known-item. In calculating recall, credit was given for all known-items that a result item matched. The number of known-items varied from 1 to 60 with a mean of 5.63, so the upper bound on precision in a result set of 100 items was quite low.

3.5 Known-item issues/lessons

Evaluation of the known-item searches turned out to be more difficult than we anticipated. Because neither the known-items nor the result items were chosen from a predefined set of shot bounds or other video segments, a parameterized matching procedure was defined as described above. It is not yet clear if/how system performance across a range of parameter settings is most usefully reported and depicted. If retrieval and evaluation could be done in terms of a reasonable set of predefined segments, the matching problem might be avoided.

Table 5: Raw counts of video assessment (dis)agreements

Counts of assessor (dis)agreements by type		
	B: Relevant	B: Not relevant
A: Relevant	1524	587
A: Not relevant	553	4729

3.6 Evaluation of general searches

Submissions for the general search topics were evaluated by retired information analysts at NIST. They were instructed to familiarize themselves with the topic material and then judge each submitted clip relevant if it contained material which met the need expressed in the topic as they understood it, even if there was non-relevant material present. Otherwise they were told to judge the clip as not relevant. They used web-based software developed at NIST to allow them to (re)play the video, audio, and image examples included in the topic as well as the submitted clips.

We had time to get a second set of judgments of the submitted materials. The raw counts of the ways in which the pairs of assessments (dis)agree are as shown in Table 5.

There were 7393 pairs of judgments. Overall, the two assessors agreed 84.6% of the time. On average, if either one of the assessors said the item was relevant, the other agreed 72.8% of the time. On average, if either one of the assessors said the item was not relevant, the other agreed 89.2% of the time. This is as good or better than the agreement among assessors judging text documents as measured in TREC-2 and TREC-4.

3.7 General Search Measures

The measure calculated for the evaluation general searching was precision.

We also made an effort to calculate a partial recall score. Each result item that was judged relevant and came from a file covered by the shot boundary reference was compared to the shots defined by the shot boundary reference. A reference shot was marked as relevant if at least one relevant result item matched it. A result item matched if it overlapped with the reference shot and the overlap was at least one third of the result item and at least two thirds of the reference shot. A result item could match more than one reference shot.

Table 6: Raw counts of intra-assessor assessment (dis)agreements

Intra-assessor (dis)agreements					
Result item types by times judged	Total items of each type	Number of total agreements		Number of disagreements	Disagreements as percent of total items
		Rel	Not Rel		
1	3849	-----	-----	-----	-----
2	1633	564	1054	15	1%
3	91	29	59	3	3%
4	1	1	0	0	0%

Once the relevant reference shots for each topic has been identified, each submission was evaluated against this partial list of relevant shots. The same matching criteria as above were applied in deciding which result items matched relevant reference shots. The table at the back of these proceedings shows the results of this procedure.

3.8 General Search Issues/Lessons

No pooling

Some groups submitted runs from multiple related systems which returned identical shots. No attempt was made to remove these since, lacking predefined retrieval units, we did not expect to be able to pool results and so did not try. This means some shots were assessed more than once by the same assessor. This set could be looked at as a sort of "natural experiment" for information on within-assessor consistency.

Interpretation of topics

Questions from the assessors about how to interpret the topics raised important issues in multimedia topic formulation. Basically the problems had to do with the relationship between the text and non-textual parts of the topic. Often it was not clear that all of the example was exemplary, but there was no way to indicate, even to a human, what aspects of the example to emphasize or ignore.

4 Approaches in brief

The following are very short descriptions of the approaches taken by each participating research group. For detailed information the reader should consult the relevant system- specific paper in these proceedings.

- Carnegie Mellon University

  Search: with, and without, the Sphinx speech recognition system, both automatic and interactive searches; Minor changes to the Informedia system; Used colour histogram matching, texture, video OCR, face detection and speech recognition;

- CLIPS IMAG Grenoble (Fr)

  Shot boundary detection (SBD): where there is significant motion between adjacent frames, uses motion compensation based on optical flow, and a photo flash detector, and a dissolve detector;

- Dublin City University (Irl)

  SBD: some work on macroblock patterns but only on partial dataset;

  Search: interactive, to evaluate the effectiveness of 3 different keyframe browsers (timeline, slideshow, hierarchical), used 30 real users, each doing 12 topics using 3 browsers each;

- Fudan University (China)

  SBD: used frame differences based on luminance and colour histograms;

  Search: for 17 topics, calculated camera motion, face detection and recognition, video text and OCR, speaker recognition and clustering, speech recognition and speaker gender detection;

- Glasgow University (UK)

  SBD: Examining the frequency of occurrence of macroblock types on compressed files, technique not tuned to gradual transitions;

- IBM Groups Almaden and T.J. Watson (US)

  SBD: used the IBM CueVideo toolkit;

  Search: with, and without, speech recognition, automatic and interactive searching tasks; based on the semi-automatic construction of models for different kinds of scenes, events and objects - extensive experiments;

- Imperial College (UK)

  SBD: used colour histograms but by comparisons across a range of frame distances, instead of the usual adjacent frames;

- Johns Hopkins University (US)

  SBD: based on colour histogram and luminance;

  Search: treated video as a sequence of still images and used colour histograms and texture to match query images and topic video keyframes vs. video data keyframes; no processing of text or audio; no previous video experience;

- Lowlands Group (NL)

  Search: both automatic and interactive searching, used output from CMU speech processing plus recognition of video text via OCR, detector for the number of faces on-screen, camera motion (pan, tilt, zoom), scene detectors, and models of lazy, interactive users;

- Microsoft Research Asia (China)

  SBD: working on uncompressed video, 2 techniques for hard and for gradual shots, integrated together; very elaborate SBD technique;

- University of Maryland (US)

  SBD: based on examining macroblock and DCT coefficients;

  Search: temporal colour correlogram (a colour histogram with the spatio-temporal arrangement of colours considered) is used to automatically retrieve from video topic examples;

- University of North Texas (US)

  Search: did 13 of the general search topics; used a keyframe extractor and an image retrieval tool to match topics which had exemplar video or images;

5 Summing up and moving on

The track revealed that there are still a lot of issues to be addressed successfully when it comes to evaluating the performance of retrieval on digital video information and it was encouraging to see so much interest from the community who specialise in evaluation of interactive retrieval, in what was achieved in the video track.

Overall, the track was a great success with more participants than expected and the promise of even more groups next year. However the real impact of the track was not in the measurement of the effectiveness of one approach to retrieval from digital video archives over another approach but was in the fact that we have now shown that there are several groups working in this area worldwide who have the capability and the systems to support real information retrieval on large volumes of digital video content. This year's TREC video track was a wonderful advertisement for what some current content-based video retrieval systems are capable of and of the potential we have for future development.

For next year it is hoped that we will be able to use a new dataset which will be greater in size, and more challenging in nature - perhaps as much as 100 hours if we can get such data. It is expected that we will repeat the searching task with a more focussed set of topics, though we will still use multimedia topic descriptions. We are also likely to have a variety of detection tasks such as the occurrence of faces, text, camera motion, speech and dialogue properties, etc. to be included in addition to the automatic detection of shot boundaries as was done this year. Finally, some participants may use MPEG-7 as an interchange format. All of the decisions on these, and other, topics will be made over the TREC Video mailing list in the coming months.

6 Authors' note

More information about the track is available from the track website at www-nlpir.nist.gov/projects/trecvid. The interaction (e.g., topics, submissions, and evaluation output) was based on XML for which DTDs are available on the website.

Finally, we would like to thank all the track participants and other contributors on the mailing list whose combined efforts made the first running of the track possible. The spirit of the track was been very positive. Special thanks to everyone who early on did the tedious work of watching the videos and making up candidate topics and more recently to Jan Baan et al at TNO for help in better addressing the varying frame numbering problem as deadlines loomed.

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.

AIM. (2001). *AIM home page in French.* URL: www-asim.lip6.fr/AIM.

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.

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.

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.

Over, P. (2001). *NIST Digital Video Collection home page.* URL: www-nlpir.nist.gov/projects/dv.

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

Overview of the TREC-2001 Web Track

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

May 9, 2002

Abstract

TREC-2001 saw the falling into abeyance of the Large Web Task but a strengthening and broadening of activities based on the 1.69 million page WT10g corpus. There were two tasks. The topic relevance task was like traditional TREC ad hoc but used queries taken from real web search logs from which description and narrative fields of a topic description were inferred by the topic developers. There were 50 topics. In the homepage finding task queries corresponded to the name of an entity whose home page (site entry page) was included in WT10g. The challenge in this task was to return all of the homepages at the very top of the ranking.

Cursory analysis suggests that once again, exploitation of link information did not help on the topic relevance task. By contrast, in the homepage finding task, the best performing run which did not make use of either link information or properties of the document's URL achieved only half of the mean reciprocal rank of the best run.

1 Introduction

The TREC-9 Web Track activities centred on two tasks: A Topic Relevance Task and a HomePage Finding Task. Both made use of a 10 gigabyte, 1.69 million document subset of the VLC2, distributed on five CD-ROMs as the WT10g collection. [Bailey et al. 2001].

2 Guidelines

2.1 This Year's Aims

1. To extend the utility of the WT10g Web test collection by obtaining "sufficiently complete" relevance judgements for 50 additional (correctly spelled) ad hoc (topic relevance) topics.
2. To explore a different type of retrieval task (homepage finding) for which it is known that link-based methods can be beneficial.
3. To investigate the benefit (or harm) of correctly implemented link methods on topic relevance.

Participants are welcome to explore specific Web retrieval issues, such as:

1. Can Distributed Information Retrieval techniques be used to improve retrieval effectiveness and/or efficiency?
2. How well can systems accommodate to misspelled queries. Note that the intention is that the standard query set will be correctly spelled so that we maximise the chance of finding all the relevant answers. However, if participants are sufficiently interested, we could issue a set of misspelled variants of the judged queries.

There are obviously many other interesting questions to ask about the Web data.

2.2 Dataset

The data for the TREC-9 Main Web Task is the 10 gigabyte WT10g [CSIRO 2001] collection, distributed by CSIRO. Note that this is entirely Web data. Documents include the information returned by the http daemon (enclosed in DOCHDR tags) as well as the page content. A draft paper [Bailey et al. 2001] describing the WT10g collection is available.

2.3 Web Ad Hoc Task

TREC-2001 ad hoc topics (topics 501-550) were created by NIST. They are available from the main TREC website [National Institute of Standards and Technology 1997]. They take a similar form to previous TREC Ad Hoc topics, but the topic title is a real Web query taken from search engine logs and the other fields are reverse engineered by NIST assessors. The additional fields are intended to define what the searcher wanted (but didn't fully specify) when they typed their query.

Systems are officially compared only on the basis of title-only queries, processed completely automatically. Queries using additional fields have no Web reality! However, despite this, participants were encouraged to submit additional interactive, manual and full topic statement runs to increase the discovery rate of relevant documents in the collection. As part of the automated submission process, participants were required to identify the type of each run.

Official training data (distributed by NIST) consisted of the TREC-9 topics and qrels (topics 451-500). These were directly comparable with the TREC-2001 task.

2.4 Home Page Finding Task

NIST devised a set of 145 homepage finding queries. The process involved finding a homepage within WT10g and then composing a query designed to locate it. This is a known-item search task in which each known item is the entry page to a Website. As an example, the query "Text Retrieval Conference" might be generated for the `http://trec.nist.gov/` homepage. A minimal amount of judging was required to determine if the URLs of documents returned by participants were in fact equivalent to the answer originally chosen. For example, `http://allen.rad.nd.edu:80/` and `http://rad.nd.edu/` both refer to the home page for the Notre Dame Radiation Laboratory.

Systems are compared on the basis of the rank of the first correct answer. Measures include mean reciprocal rank of first correct answer and success rate (percentage of cases in which the correct answer or equivalent URL) occurred in the first N documents.

A set of 100 queries and correct answers generated by Nick Craswell using a similar method were made available [CSIRO 2001] for training purposes.

No manual or interactive query modification was permitted in this task. There was a blanket prohibition on tuning, tweaking or altering of systems based on examining the test queries.

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

1. All submissions were due at NIST on or before 2 August 2001.
2. An automated submission process was used which collected a small amount of information about each run.
3. No. of runs submitted/judged.
4. All judging was performed by NIST (not CSIRO) assessors.

5. Judgments in the Web Ad Hoc task (not Homepage Finding) were TERNARY (nonrelevant, relevant, highly relevant) as they were last year.

6. Judgments were made on the basis of the text within the document (only)

7. Judges were not able to follow links.

In the Topic Relevance task, 70400 documents were judged and 3363 were judged either relevant (2573) or highly relevant (790).

In the Homepage Finding task, there were a total of 252 right answers over the 145 queries, an average of 1.74 right answers per query. However, the distribution of number of right answers per query was very skewed. For 132 queries there was only one right answer but for three queries there were more than 10 right answers: query EP33 (Best Internet) - 25, query EP122 (Society for Technical Communications) - 22, and query EP139 (The Leader OnLine) - 17).

Best Internet seems to be (have been) an internet hosting company which controls a whole lot of internet domain names and presents all of them with its own homepage (prior to selling them to customers I presume). The URLs by which this page was accessible included: www.voici.com, www.avantisoft.com, www.panint.com, www.samoyed.org, www.cookiefactory.com, www.prost.org, www.bayberry.com, www.voici.com, www.biloxi-ms.com, www.globeprint.com, www.buoymedia.com, www.nm-solutions.com, www.growing.com, www.caber.com, apogee.best.com, 204.156.149.14, www.weblab.com, www.anymtnltd.com, www.romenet.com, www.spottedantelope.com, www.straw.com, www.jjsblues.com, www.jointventure.org, 204.156.144.1, www.mochinet.com, www.flick.com.

By contrast, the multiple results for the Society for Technical Communications, seem to include some spurious answers. The real home page appears to be at www.stc-va.org/display.html but lots of the others judged equivalent are subsidiary pages or homepages of individual chapters or regions of STC.

Finally, the multiple answers for the Online Leader, correspond to separate issues of an online publication. Each issue looks like a homepage but each has a specific date, eg. www.olympus.net/leader/leaderonlineoctober23961023.htm. The page which you might expect to be a homepage (www.olympus.net/leader/index.html) also has a date.

We considering URL depth to be the number of slashes in the URL after eliminating trailing slashes, we computed a histogram of the shallowest right answer for each of the queries. It turns out that 95 of the 145 shallowest answers are actually at the very top level eg. africa.cis.co.za:81, amelia.experiment.db.erau.edu, dbcl13.cs.ust.hk01. Only 11 of the shallowest right answers are at a depth greater than 2.

3 Results

3.1 Topic Relevance Task

Table 1 gives details of the 77 official submissions in the title-only, automatic category of the Topic Relevance task. The best performing run fub01be2 (FUB) did not make use of links, document structure, or URL text. Features listed for that run were: no-stemming, single-word indexing, novel probabilistic term weighting model, automatic query expansion.

The second best run JuruFull (IBM-Haifa) used document structure and referring anchortext. Features listed for that run were: Vector space model, using lexical affinites, Porter stemming, slight stopword filtering.

The best run from the third-ranked group (Ricoh) used only document content. Features listed for that run were: Probabilistic model, Query expansion, Automatic parameter value estimation

The best run from the fourth ranked group (JustSystem) made use of link information but at this stage it is unclear how. Features listed for that run were: vector space search, reference DB, pseudo-relevance feedback

In summary, it was possible to achieve top performance using document content only. Automatic query expansion was used by most of the top ranked runs. There was no clear advantage to either probabilistic or vector space approaches.

Table 2 gives details for the 20 other runs, including two manual runs. The best full-topic automatic run performed 27% better than the best title-only run. Interestingly, it made use of URL text as well as page content.

3.2 Home Page Finding Task

Table 3 gives details of all 43 official runs in the Home Page Finding task. Interestingly, the top 23 runs in this table all made use of either URLtext or links (or both). The best run which did not (IBMHOMENR) achieved an MRR score only half as high as that of the top ranked run. It made use of document structure. The highest ranked run which used content only achieved an MRR score only 30% of the best and found a right answer in the top 10 only half as often.

The performance of the top ranked run (tnout10epCAU) is quite impressive. It found a right answer in the top 10 in nearly 90% of cases. The features of this run were listed as follows: Unigram language model URL text priors (based on depth of URL-path) content run merged with seperate anchor-text run. Interestingly, a companion run which did not use anchor text scored almost as well, reflecting the importance of URL depth as a feature on this task - at least for this set of queries on this collection.

Acknowledgements

With assistance from her colleagues at NIST, Ellen Voorhees played a major role in organising the Web track, through topic formulation, assessment, evaluation and analysis. Much of the Main Web data and many of the analyses reported here are the result of her work.

The pivotal contribution of Peter Bailey in engineering the WT10g is gratefully acknowledged.

We are very much indebted to Brewster Kahle of the Internet Archive for making available the spidered data from which the VLC2 and WT10g collections.

Finally, thanks are due to the NIST assessors for topic development and assessments.

Bibliography

Bailey, P., Craswell, N., and Hawking, D. 2001. Engineering a multi-purpose test collection for web retrieval experiments. *Information Processing and Management*. In press. www.ted.cmis.csiro.au/~dave/cwc.ps.gz.

CSIRO. 2001. TREC Web Tracks home page. www.ted.cmis.csiro.au/TRECWeb/.

National Institute of Standards and Technology. 1997. TREC home page. trec.nist.gov/.

Table 1: All official submissions in the title-only, automatic topic relevance task.

Runid	Group	Fields	Struct.	URLtext	Links	AveP	ret(100)	ret(1000)
fub01be2	FUB	T	-	-	-	0.2226	17.38	46.1
JuruFull	IBM-Haifa	T	Y	Y	-	0.2105	17.22	45.6
JuruFullQE	IBM-Haifa	T	Y	Y	-	0.2091	16.92	42.6
ricMM	ricoh	T	-	-	-	0.2084	16.84	47.4
ricAP	ricoh	T	-	-	-	0.2077	17.62	49
ricMS	ricoh	T	-	-	-	0.2068	16.84	47.4
JuruPruned01	IBM-Haifa	T	Y	Y	-	0.2066	17.48	43.1
JuruPrune006	IBM-Haifa	T	Y	Y	-	0.2065	17.3	44.1
jscbtawtl4	Justsystem	T	Y	-	Y	0.2060	16.88	46.9
jscbtawtl3	Justsystem	T	Y	-	Y	0.2003	16.9	45.9
Lemur	cmu-lti	T	-	-	-	0.1985	17.68	48
fub01ne2	FUB	T	-	-	-	0.1962	16.42	42.9
jscbtawtl2	Justsystem	T	Y	-	Y	0.1954	16.3	45.2
ok10wt3	microsoft	T	-	Y	-	0.1952	16.86	47.2
hum01tlx	hummingbird	T	Y	-	-	0.1949	16.48	45.8
ricST	ricoh	T	-	-	-	0.1933	16.2	46.3
msrcn1	microsoft-china	T	Y	Y	-	0.1913	16.9	45.1
ok10wt1	microsoft	T	-	Y	-	0.1908	16.78	46.7
fub01idf	FUB	T	-	-	-	0.1900	16.44	42.4
tnout10t2	tno/utwente	T	-	-	-	0.1891	16.66	47.1
iit01tfc	IIT	T	-	-	-	0.1890	16.92	47.6
jscbtawtl1	Justsystem	T	Y	-	Y	0.1890	15.84	44.6
msrcn4	microsoft-china	T	Y	Y	-	0.1880	14.12	43.6
msrcn2	microsoft-china	T	Y	Y	Y	0.1864	14.14	43.6
fub01ne	FUB	T	-	-	-	0.1790	14.98	41.2
hum01tl	hummingbird	T	Y	-	-	0.1784	15.1	43.7
msrcn3	microsoft-china	T	Y	Y	-	0.1779	14.3	39.9
posnir01rpt	postech	T	Y	-	-	0.1771	14.7	42.1
pir1Wt2	cuny	T	-	-	-	0.1742	15.2	45.5
flabxt	Fujitsu	T	-	-	-	0.1719	16.4	43.1
UniNEtdL	Neuchatel	T	-	-	-	0.1715	14.58	43.2
flabxtl	Fujitsu	T	-	-	Y	0.1705	16.06	43
UniNEt7dL	Neuchatel	T	-	-	-	0.1699	14.66	44
fdut10wtc01	Fudan	T	-	-	-	0.1661	15.1	34.7
pir1Wt1	cuny	T	-	-	-	0.1660	14.68	45.3
UniNEtd	Neuchatel	T	-	-	-	0.1659	14.34	43.3
tnout10t1	tno/utwente	T	-	-	-	0.1652	14.62	43.6
hum01t	hummingbird	T	Y	-	-	0.1582	14.42	40.8
apl10wc	apl-jhu	T	-	-	-	0.1567	14.12	42.1
fdut10wtl01	Fudan	T	-	-	Y	0.1544	14.56	34.7
posnir01st	postech	T	Y	-	-	0.1636	13.7	42
posnir01pt	postech	T	Y	-	-	0.1521	13.96	42.2
iit01t	IIT	T	-	-	-	0.1509	13.92	40.1
ARCJ0	ibm-web	T	Y	-	-	0.1497	11.94	31.4
ARCJ6	ibm-web	T	Y	-	Y	0.1439	11.88	31.4
Merxt	IRIT	T	-	-	-	0.1438	13.76	39.9
uwmtaw2	waterloo	T	-	-	-	0.1420	13.88	39.9
uwmtaw1	waterloo	T	-	-	-	0.1416	12.84	39
PDWTAHDR	padova	T	-	-	-	0.1332	12.74	37.8
Ntvenx2	nextrieve	T	Y	-	-	0.1313	11.94	33.3
yeahtb01	Yonsei	T	Y	-	-	0.1287	12.84	26.8
yeaht01	Yonsei	T	Y	-	Y	0.1286	12.82	26.7
Ntvenx1	nextrieve	T	Y	-	-	0.1273	11.76	35.5
PDWTAHWL	padova	T	-	-	Y	0.1209	11.56	37.8
Ntvfnx3	nextrieve	T	Y	-	-	0.1128	11.94	30.1
ajousi0103	ajou	T	-	-	Y	0.1116	10.72	37.1
ajousi0101	ajou	T	-	-	-	0.1114	10.74	37.1
csiro0awa1	CSIRO	T	Y	Y	Y	0.1085	10.68	34.3
uncvsms	uncYang	T	-	-	-	0.1069	12.26	33.4
Ntvfnx4	nextrieve	T	Y	-	-	0.0978	10.08	25.8
uwmtaw0	waterloo	T	-	-	-	0.0951	11.3	27.2
csiro0awa3	CSIRO	T	Y	Y	Y	0.0946	10.76	29.8
icadhoc3	imperial	T	-	-	-	0.0883	9.88	26.9
ictweb10n	chinese_academy	T	-	-	-	0.0860	9.42	28.6
ictweb10nl	chinese_academy	T	-	-	Y	0.0860	9.54	28.5
PDWTAHPR	padova	T	-	-	-	0.0842	10.14	36.7
apl10wa	apl-jhu	T	-	-	-	0.0805	9.72	34
csiro0awa2	CSIRO	T	Y	Y	Y	0.0789	9.48	27.9
apl10wb	apl-jhu	T	-	-	-	0.0671	6.96	12
uncfsls	uncYang	T	-	-	Y	0.0663	11.62	33.3
PDWTAHTL	padova	T	-	-	Y	0.0601	6.82	37.8
icadhoc1	imperial	T	-	-	Y	0.0537	7.96	24.5
ictweb10nf	chinese_academy	T	-	-	-	0.0464	6.68	28.4
ictweb10nfl	chinese_academy	T	-	-	Y	0.0463	5.68	28.4
icadhoc2	imperial	T	-	-	Y	0.0458	8.28	23
irtLnut	uncNewby	T	Y	-	-	0.0221	3.36	16.8
irtLnua	uncNewby	T	Y	-	-	0.0002	0.06	0.6

Table 2: All other (manual and long automatic) official submissions in the topic relevance task. Manual runs are marked with an asterisk.

Runid	Group	Fields	Struct.	URLtext	Links	AveP	ret(100)	ret(1000)
iit01m*	IIT		-	-	-	0.3324	20.8	43.2
ok10wtnd1	microsoft	TDN	-	Y	-	0.2831	22.36	53.8
csiro0mwa1*	CSIRO		Y	Y	Y	0.2817	19.68	42
ok10wtnd0	microsoft	TDN	-	Y	-	0.2512	20.42	51.7
flabxtd	Fujitsu	TD	-	-	-	0.2332	19.88	49
UniNEn7d	Neuchatel	TDN	-	-	-	0.2242	17.52	48.8
hum01tdlx	hummingbird	TD	Y	-	-	0.2201	18.62	49.4
kuadhoc2001	kasetsart	TDN	-	-	-	0.2088	17.7	44.9
apl10wd	apl-jhu	TDN	-	-	-	0.2035	19.56	50.5
posnir01ptd	postech	TD	Y	-	-	0.1877	17.62	44.5
flabxtdn	Fujitsu	TDN	-	-	-	0.1843	17.32	43.4
iit01tde	IIT	TD	-	-	-	0.1791	16.9	45
Merxtd	IRIT	TD	-	-	-	0.1729	15.58	42.4
pir1Wa	cuny	TDN	-	-	-	0.1715	14.88	45.7
fdut10wac01	Fudan	TDN	-	-	-	0.1661	15.1	34.7
uncvsmm	uncYang	TD	-	-	-	0.1269	14.4	35.9
fdut10wal01	Fudan	TDN	-	-	Y	0.1248	12.72	34.7
yeahdb01	Yonsei	TD	Y	-	-	0.1094	11.52	23.5
yeahtd01	Yonsei	TD	Y	-	Y	0.1092	11.48	23.5
uncfslm	uncYang	TD	-	-	Y	0.0781	13.46	35.8

Table 3: All official submissions in the homepage finding task. MRR is the mean reciprocal rank of the first correct answer. %top10 is the proportion of queries for which a right answer was found in the top 10 results. %fail is the proportion of queries in which no right answer was found in the top 100 results.

Runid	Baseline	Group	Struct.	URLtext	Links	MRR	%top10	%fail
tnout10epCAU	tnout10epCU	tno/utwente	-	Y	Y	0.774	88.3	4.8
tnout10epCU		tno/utwente	-	Y	-	0.772	87.6	4.8
jscbtawep2		Justsystem	Y	Y	Y	0.769	83.4	9.0
jscbtawep1		Justsystem	Y	Y	Y	0.754	83.4	9.0
jscbtawep4		Justsystem	Y	Y	Y	0.752	83.4	8.3
jscbtawep3		Justsystem	Y	Y	Y	0.746	83.4	9.0
yehp01	yehpb01	Yonsei	Y	Y	Y	0.669	76.6	22.1
yehpb01		Yonsei	Y	Y	-	0.659	75.9	22.8
UniNEep1		Neuchatel	-	Y	-	0.637	69.0	8.3
UniNEep2		Neuchatel	-	Y	-	0.637	69.0	7.6
IBMHOMER	IBMHOMENR	ibm-web	Y	-	Y	0.611	77.9	10.3
flabxeall		Fujitsu	-	-	Y	0.599	80.7	9.7
csiro0awh2		CSIRO	-	-	Y	0.593	71.7	21.4
iit01stb	iit01st	IIT	Y	Y	Y	0.578	66.9	24.8
iit01st		IIT	Y	Y	-	0.559	62.8	29.7
UniNEep3		Neuchatel	-	Y	-	0.530	68.3	6.9
VTEP	VTBASE	VT	-	Y	Y	0.506	68.3	15.9
msrcnp2	msrcnp1	microsoft-china	Y	Y	Y	0.505	69.0	15.2
csiro0awh1	csiro0awh3	CSIRO	Y	Y	Y	0.498	72.4	11.0
UniNEep4		Neuchatel	-	Y	-	0.477	68.3	11.0
msrcnp1		microsoft-china	Y	Y	-	0.424	65.5	13.1
flabxe75a		Fujitsu	Y	Y	Y	0.399	55.9	37.9
ok10wahd1	ok10whd1	microsoft	-	Y	Y	0.387	64.1	13.1
IBMHOMENR		ibm-web	Y	-	-	0.382	62.1	11.7
flabxemerge		Fujitsu	Y	Y	Y	0.365	51.0	33.8
flabxet256		Fujitsu	Y	-	Y	0.363	50.3	33.8
ok10wahd0	ok10whd0	microsoft	-	Y	Y	0.362	62.1	13.1
ok10whd1		microsoft	-	Y	-	0.340	60.7	15.9
tnout10epC		tno/utwente	-	-	-	0.338	58.6	18.6
tnout10epA		tno/utwente	-	-	Y	0.331	48.3	35.9
ok10whd0		microsoft	-	Y	-	0.312	58.6	15.2
apl10ha		apl-jhu	-	-	-	0.238	44.8	22.1
ichp2		imperial	-	-	-	0.237	44.8	29.7
apl10hb		apl-jhu	-	-	-	0.220	42.8	21.4
ichp1	ichp2	imperial	-	-	Y	0.208	33.8	37.2
kuhpf2001		kasetsart	-	-	-	0.191	36.6	42.1
PDWTEPDR		padova	-	-	-	0.189	33.8	42.8
PDWTEPWL	PDWTEPDR	padova	-	-	Y	0.178	30.3	42.8
VTBASE		VT	-	-	-	0.126	24.1	45.5
ajouai0102		ajou	-	-	-	0.101	23.4	49.7
ajouai0104		ajou	-	-	Y	0.100	23.4	49.7
PDWTEPTL	PDWTEPDR	padova	-	-	Y	0.099	20.0	42.8
PDWTEPPR		padova	-	-	-	0.054	13.1	44.8

TREC 2001 Cross-lingual Retrieval at BBN

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

1 INTRODUCTION

BBN only participated in the cross-lingual track in TREC 2001. Arabic, the language of the TREC 2001 corpus, presents a number of challenges to both monolingual and cross-lingual IR. First, many inflected Arabic words can correspond to multiple uninflected words, requiring context to disambiguate them. Second, orthographic variations are prevalent; certain glyphs are sometimes written as different, but similar looking glyphs. Third, broken plurals, analogous to irregular nouns in English, are very common. Such nouns cannot be easily reduced to their singular forms using a rule-based approach. Fourth, Arabic words are highly ambiguous due to the tri-literal root system and the omission of short vowels in written Arabic. The focus of this report is to explore the impact of these issues on Arabic monolingual and cross-lingual retrieval.

2 ISSUES IN ARABIC RETRIEVAL

2.1 Stemming

We used a modified version of Buckwalter's 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 re-implemented the stemmer to make it faster and compatible with UTF8 encoding. Also, we modified it so that if no valid combination of prefix-stem-suffix is found, the word itself is returned as the stem.

Ambiguities arise when a word has several stems. We used two techniques to deal with this problem. With the *sure-stem* technique, we only stem a word if it has exactly one stem. Otherwise, the word is left alone. With the *all-stems* technique, we probabilistically map a word to all possible stems. Since our retrieval system is based on a probabilistic generative model, such ambiguities can be easily accommodated. In the absence of training data, we assume that all possible stems are equally probable. That is, if a word has n possible stems, each stem gets $1/n$ probability. The advantage of sure-stem is that it does not introduce additional ambiguity, while the advantage of all-stems is that it always finds a stem for a word when one exists.

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

2.2 Orthographic Variation

Arabic orthography is highly variable. For instance, changing the letter YEH (ي) to the letter ALEF MAKSURA (ى) at the end of a word is very common. (Not surprisingly, the shapes of the two letters are very similar.) 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.

A much trickier type of orthographic variation is when certain diacritical ALEFs (e.g. أ , إ and آ) are written as the plain ALEF (ا). Often, both the intended word and what is actually written are valid words. This is much like confusing "résumé" with "resume" in English. We explored two techniques to address the problem. With the *normalization* technique, we replace all occurrences of the diacritical ALEFs by the plain ALEF. With the *mapping* technique, we map a word with the plain ALEF to a set of words that can potentially be written as that word by changing diacritical ALEFs to the plain ALEF. In this absence of training data, we will assume that all the words in the set are equally probable. Both techniques have pros and cons. The normalization technique is simple, but it increases ambiguity. The mapping technique, on the other hand, does not introduce additional ambiguity, but it is more complex. Another problem is that the uniform probability assignment may deviate from the true probability distributions.

2.3 Broken Plurals

Broken plurals, analogous to irregular nouns in English (e.g. "woman/women"), are very common in Arabic. It is hard if not impossible to write a rule-based algorithm to reduce them to singulars. As such, broken plurals are not dealt with by the Buckwalter stemmer.

The problem is primarily a concern for monolingual retrieval. For CLIR, it is not a major problem because plurals and singulars can be translated separately. For monolingual IR, we use a statistical thesaurus, derived from the UN parallel corpus, to address the problem of broken plurals. The basic idea is that the singular and the plural forms of the same Arabic word should have the same stemmed translations in English. The problem can be formalized as the problem of estimating the probability that a user uses one Arabic word *b* to describe another Arabic word *a*. That is achieved by translating *a* to an English word *x* and then translating *x* to *b*. Translation probabilities from *a* to *x* and *x* to *b* are estimated by applying a statistical machine translation tool-kit, GIZA++ (to be described later), on the UN parallel corpus. It is easy to verify that

$$P_{thesaurus}(b \mid a) = \sum_{English\ words\ x} p(x \mid a)p(b \mid x)$$

A mixture model was used to emphasize the original words in the translation:

$$p(b \mid a) = 0.4 p_{diag}(b \mid a) + 0.6 p_{thesaurus}(b \mid a)$$

where $p_{diag}(b|a)$=1 if a=b and 0 otherwise. The mixture weights were chosen based on experiments using the TREC-8 English monolingual test queries.

2.4 Tri-literal root system and omission of vowels

Most Arabic words can be derived from a small number (e.g. a few thousands) of roots. Most roots consist of only three consonants. Making things worse, short vowels are normally omitted in written Arabic. As a result, Arabic words tend to have a high level of ambiguity. If not addressed, this problem will hurt cross-lingual retrieval, because an Arabic word would have many translations.

Instead of explicit disambiguation, which weeds translations out based on context, we use a probabilistic solution that differentiates likely and unlikely translations. Although an Arabic word may have many translations, certain translations are more likely than others; hence, probability estimates limit the impact of ambiguity. In our CLIR experiments, we estimate translation probabilities from a large parallel corpus (the UN corpus) in addition to a manual bilingual lexicon.

3 BILINGUAL RESOURCES

We used a manual lexicon and a parallel corpus for estimating term translation probabilities. The manual lexicon consists of word pairs from three sources:

- A bilingual term list from Buckwalter (Buckwalter, 2001), with 86,000 word pairs.
- 20,000 word pairs, derived by applying the Sakhr machine translation system (http://www.sakhr.com/) on a list of frequent English words
- 10,000 word pairs gleaned from NMSU's named entity lexicon (http://crl.nmsu.edu/~ahmed/downloads.html).

Uniform translation probabilities are assumed for the English translations in the lexicon. That is, if an Arabic word has *n* English translations, each translation gets probability *1/n*.

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). Model 1 in Brown's work was used in this work for its efficiency.

The translation probabilities for the two sources were linearly combined to produce a single probability estimate for each word pair:

$$p(e \mid a) = 0.8 p_{un}(e \mid a) + 0.2 p_{lexicon}(e \mid a)$$

where *e* is an English word, *a* is an Arabic word, p_{un} and $p_{lexicon}$ are probabilities from the UN corpus and the manual lexicon respectively. We gave a higher weight to the UN corpus because it appears to be of higher quality.

4 OUR RETRIEVAL SYSTEM

Our retrieval system was documented in (Xu and Weischedel 2000; Xu et al, 2001). Our system ranks documents based on the probability that a query is generated from a document:

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

Where Q is a query, D is a document, t_q's are query terms, t_d's terms in the document. GL is a background corpus of the query language. The mixture weight α is fixed to 0.3. $p(t_q|t_d)$ is the translation probability from t_d to t_q. We estimate $p(t_q|GL)$ and $\mathrm{p}(t_d|D)$ as:

$$p(t_q \mid GL) = \frac{\textit{frequency of } t_q \textit{ in } GL}{\textit{size of } GL}$$

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

In our cross-lingual experiments, the general English corpus (i.e. *GL* in the formulas) consists of newspaper articles in the TREC English disks 1-5 and more recent articles from FBIS. Translation probabilities were estimated as described in the previous section.

Because monolingual retrieval is a special case of cross-lingual IR, where document terms and query terms happen to be of the same language, the same system was used for both cross-lingual and monolingual IR. For simple monolingual IR, the translation matrix is an identity matrix (a diagonal matrix with 1's on the diagonal). In that case, the retrieval model is the same as the one proposed by (Miller, Leek and Schwartz, 1999). For thesaurus-based retrieval, the translation matrix is the thesaurus.

Our system can easily accommodate the all-stems technique for stemming and the mapping technique for orthographic resolution, since both are simple probabilistic translations. In CLIR, these translations are applied before the translations to English terms. In other words, the translation from a document term to a query term consists of a number of intermediate translations. It is easy to verify that the translation matrix from document terms to query terms is the product of the intermediate translation matrixes.

5 OFFICIAL RESULTS

In all submitted runs, the document terms are unstemmed Arabic words. Words with apparently incorrect endings such as substitution of ALEF MAKSURA (ى) for YEH (ي) were handled automatically as described in Section 2.2. We submitted one official monolingual run and four official cross-lingual runs as follows:

- BBN10MON. Our monolingual run. Only the title and description fields were used for query formulation. Queries consist of Arabic stems. In query processing, each Arabic word is replaced by its stem(s). The statistical thesaurus described before was used for translations between Arabic stems.

Stop words were removed. Our stop word list was obtained from Yaser Al-Onaizan at ISI (http://www.isi.usc.edu). That list was augmented with a few manually selected high frequency words from the AFP corpus.

The mapping technique was used for orthographic resolution. The all-stems technique was used for stemming. Both were applied before the thesaurus translations of Arabic stems.

Automatic query expansion was used to add additional terms to the queries. An initial retrieval was performed on an Arabic corpus consisting of AFP (i.e. the TREC 2001 corpus) and additional articles from newspaper sources Al-Hayat and An-Nahar. For each query, 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 re-weighted:

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

where D_i's are the top retrieved documents.

- BBN10XLC. Cross-lingual run without query expansion. Only the title and description fields of the English topics were used for query formulation. Term translation used both the manual bilingual dictionary and the statistical bilingual dictionary described in the previous section.
- BBN10XLB. Cross-lingual run with Arabic expansion. In addition to BBN10XLC, Arabic query expansion terms were used. The same query expansion procedure in BBN10MON was used here.
- BBN10XLA. Cross-lingual run with Arabic and English expansions. In addition to BBN10XLB, English expansion terms were used. English documents were retrieved from a newspaper corpus with 1.2 million articles from sources AP, Reuters and FBIS.
- BBN10XLD. Cross-lingual run with long queries. Same as BBN10XLA, except the narrative field was also used in query formulation. Arabic and English expansions were used.

The mapping technique was used for orthographic resolution and the all-stems technique was used for stemming in BBN10MONO. In contrast, in the cross-lingual runs, normalization and sure-stem were used in deriving term translations from the UN corpus.

Table 1 shows the TREC average precision for each run. In addition, it shows the number of queries in each run that achieved the best monolingual or cross-lingual performance among all submitted runs and the number of queries above the median. Overall, all our runs achieved very good performance.

Table 1 Retrieval results for official runs

	Average Precision	=best(out of 25)	>median(out of 25)
BBN10MON	0.4537	14	21
BBN10XLA	0.4382	6	23
BBN10XLB	0.4639	8	24
BBN10XLC	0.3604	0	22
BBN10XLD	0.4453	3	22

6 EXPERIMENTS USING SHORT QUERIES

The TREC 2001 topics are very long. Excluding stop words, the full topics have 26 English words per topic. Without the narrative field, the average query length is 12 words per query, still too long for typical ad hoc retrieval. The title field, which has an average of 6.6 words per topic, is more realistic. Table 2 shows the scores our official runs would have achieved h`ad we used only the title field in query formulation. The degradation due to the shortened queries is modest, except for BBN10XLA, for which the degradation is very large.

Table 2 Title and description vs title-only for query formulation

	BBN10MON	BBN10XLA	BBN10XLB	BBN10XLC
Title+Desc words (official runs)	0.4537	0.4382	0.4639	0.3604
Title words	0.4222	0.3699	0.4475	0.3441

7 MONOLINGUAL EXPERIMENTS

The goal of the following experiments is to demonstrate the impact of a number of issues on monolingual retrieval. In all experiments, query formulation used the title and description fields of the topics.

a. No text processing except for the removal of stop words in query and indexing. The translation matrix is an identity matrix.

b. All-stems stemming was used in addition to the removal of stop words. Elements in the translation matrix are "translation" probabilities from unstemmed words to stems.

c. The difference from b is that sure-stem stemming was used.

d. In addition to b, the mapping technique was used for orthographic resolution.

e. The only difference from d is that normalization instead of mapping was used for orthographic resolution.

f. Same as d, except that the statistical thesaurus was used for term translation

g. In addition to f, query expansion was used, based on AFP, Al-Hayat, and An-Nahar. This is our official monolingual run, BBN10MON.

h. Same as g, except that query expansion used only the AFP corpus.

Table 3 Monolingual results

a	b	c	d	e	f	g	h
0.1873	0.2388	0.2492	0.3145	0.3131	0.3682	0.4537	0.4571

Retrieval scores in Table 3 show that:

- Stemming is very useful for Arabic retrieval (a->b). The absolute change in performance is 0.05. The value of stemming seems to be even greater for Arabic than for English monolingual retrieval. This is not surprising given the fact that Arabic has more complex morphologies.

- There is a small difference between all-stems and sure-stem (b->c), the latter being slightly better. The difference is not statistically significant.

- Orthographic resolution is very important (b->d). The change in performance is 0.075. This suggests that word spellings in the documents are very different from those in the queries.

- There is little difference between the mapping and the normalization techniques for orthographic resolution (d->e). More research is needed to determine whether a better probability estimation procedure will improve the mapping technique.

- The automatically derived thesaurus is very useful (d->f). The performance change is 0.05. We believe that most of the improvement is due to the broken plurals successfully resolved by the thesaurus. The rest of the improvement is probably due to general synonyms captured by the thesaurus.

- Query expansion is very useful for TREC 2001 queries (f->g). The performance change is 0.085. This is not very surprising given the success of query expansion techniques in earlier TRECs.

- Query expansion using only AFP is as effective as using the combined corpus of AFP, Al-Hayat and An-Nahar (g->h). The advantage of using a larger corpus for query expansion suggested by earlier studies (e.g. Kwok and Chan, 1998) is not observed. The probable reason is that the AFP corpus already has enough relevant documents for the queries (165 relevant documents per query on average). The additional relevant documents in Al-Hayat and An-Nahar did not improve the worthiness of the top documents for the purpose of query expansion.

8 CROSS-LINGUAL EXPERIMENTS

8.1 Impact of Orthographic Variations

We compared BBN10XLC with an unofficial run where orthographic variations were not handled. Other conditions are the same for both runs. We found that there is little difference between the two runs (0.3604 vs 0.3584). This is very different from monolingual retrieval, where orthographic resolution is critical. However, the result is not surprising given the fact that different variants of the same word can be translated individually. Indeed, a casual inspection of the Buckwalter lexicon indicates that it often has separate entries for different spellings of the same word. It appears that the UN corpus also contains such spelling variations.

8.2 Effect of Arabic Stemming in Inducing a Bilingual Lexicon from a Parallel Corpus

We have compared three modes of learning term translations from the UN corpus. The first did not use stemming. The second used sure-stem. The third used all-stems. All three have pros and cons. The first keeps the maximum amount of word distinction, but requires more training data. The third requires less training data due to the reduced dimensionality, but increases word ambiguity, and the probability estimates are affected due to the one-to-many mapping from words to stems. The second is a compromise.

The retrieval scores in Table 4 show that no-stem is slightly better than sure-stem, which is slightly better than all-stems. While the differences are too small to make firm conclusions, they suggest that Arabic stemming is not an important issue in CLIR.

Table 4 Three modes of GIZA++ training: no-stem, sure-stem and all-stems

No-stem	Sure-stem	All-stems
0.3106	0.2994	0.2895

8.3 Impact of Resource Combination

Table 5 shows the retrieval scores when:

- The Buckwalter lexicon was used for term translation.
- The augmented Buckwalter lexicon (with additional word pairs from Sakhr and NMSU) was used.
- The UN corpus was used.
- All resources were combined.

Table 5 Impact of resource combination

Buckwater only	Augmented Buckwalter	UN only	ALL (BBN10XLA)
0.2695	0.2697	0.2994	0.3604

The scores indicate that the additional translation pairs from Sakhr and NMSU are not helpful. The combination of the UN and the manual lexicon significantly outperforms either resource alone, suggesting that the word ambiguity problem in Arabic is satisfactorily handled by complementing a manual lexicon with a parallel corpus. The result is consistent with our TREC9 Chinese CLIR work (Xu and Weischedel 2000).

8.4 Query Expansion

Table 6 shows that both English and Arabic expansion terms improve retrieval scores. The Arabic expansion terms are more effective than English expansion terms. This is expected because we know that the particular English corpus we used for query expansion is not a very good match for the Arabic test corpus. It is disappointing that using both sources of expansion terms does not improve retrieval further. In fact, it is worse than using Arabic expansion alone. One possible reason is that the weights for English expansion terms are larger than they should be. That suggests that reducing the weight on English expansion terms may result in better retrieval.

Table 6 Effect of query expansion on CLIR retrieval

No expansion (BBN10XLC)	English expansion	Arabic expansion (BBN10XLB)	English & Arabic expansions (BBN10XLA)
0.3604	0.4060	0.4639	0.4382

9 CONCLUSIONS

Concerning monolingual Arabic retrieval, the following proved true empirically:

- As in other languages, stemming is very important.
- Proper handling of orthographic variations is critical; the probabilistic model handled this type of ambiguity.
- A statistically derived thesaurus from a parallel corpus can effectively cope with the broken plural problem.
- Automatic query expansion by unsupervised relevance feedback proved very helpful, just as it has in other languages.

Concerning cross-lingual IR, the following was demonstrated empirically:

- Combining manual lexicons and parallel corpora in a probabilistic model gave much better performance than either alone.
- Stemming and handling of orthographic variations proved less critical for CLIR than for monolingual IR.
- Query expansion significantly improved retrieval performance, though query expansion in Arabic alone proved most effective.
- Cross-lingual retrieval outperformed monolingual retrieval, as it had in our Chinese experiments in TREC-9.

Acknowledgement: We would like to thank Ghada Osman, Mohamed Noamany and John Makhoul for their invaluable help.

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. L. Kwok and M. Chan, 1998. "Improving Two-Stage Ad-Hoc Retrieval for Short Queries." In proceedings of SIGIR 1998.

D. Miller, T. Leek, and R. Schwartz, 1999. "A Hidden Markov Model Information Retrieval System." In Proceedings of ACM SIGIR 1999.

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

J. Xu and R. Weischedel, 2000. "TREC9 Crosslingual Retrieval at BBN", *TREC9 Proceedings.*

J. Xu, R. Weischedel, and C. Nguyen, 2001. "Evaluating a Probabilistic Model for Cross-lingual Retrieval." In proceedings of ACM SIGIR 2001, pp. 105-110.

The Bias Problem and Language Models in Adaptive Filtering

Yi Zhang Jamie Callan

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15232, USA

{yiz, callan}@cs.cmu.edu

ABSTRACT

We used the YFILTER filtering system for experiments on updating profiles and setting thresholds. We developed a new method of using language models for updating profiles that is more focused on picking informative/discriminative words for query. The new method was compared with the well-known Rocchio algorithm. Dissemination thresholds were set based on maximum likelihood estimation that models and compensates for the sampling bias inherent in adaptive filtering. Our experimental results suggest that using what kind of distribution to model the scores of relevant and non- relevant documents is corpus dependant. The experimental results also show the sampling bias problem of training data while filtering makes the final profile learned biased.

1. INTRODUCTION

Given an initial description of a profile, a filtering system must sift through a stream of information and deliver the most relevant documents to the profile. Filtering is more like an online classification problem than a traditional search problem, because it is a binary decision process. Text classification algorithms, such as SVM, Rocchio, Boosting and Naive Bayes, can be applied to filtering, especially for batch filtering and routing. A common approach to learning profiles is to use an incremental version of the Rocchio algorithm [7]

$$Q' = \alpha Q + \beta \left(\frac{\sum_{D_i \in R} D_i}{|R|} \right) - \gamma \left(\frac{\sum_{D_i \in NR} D_i}{|NR|} \right)$$

Where Q is the initial profile vector, Q' is the new profile vector, R is the set of relevant documents, NR is the set of non-relevant documents, D_i is a document vector, and α, β, and γ are constants indicating the relative value of each type of evidence.

Language modeling has been applied to filtering track in TREC8 [4]. EM algorithm is used to find optimal parameters to maximize the likelihood of joint probability of relevant document and query. Our work introduces another way of using language modeling for the profile learning, which is also usin g EM but maximizing the likelihood of training data. We compares it with Rocchio in TREC10. Our result also shows the sampling bias problem (user only provide feedback for documents delivered) on learned profile terms and term weights/

2. SYSTEM DESCRIPTION

The YFILTER information filtering system we used is architecturally similar to InRoute [3]. It supports both structured Boolean and natural language descriptions of initial profiles. For natural language profiles, it can automatically update the profile according to user relevance feedback. YFILTER provides an option to use the INQUERY stopwords list and the Porter word-stemming algorithm [6].

3. PROFILE LEARNING

3.1 Initial Profile and Scoring Method

Each profile begins with topic words (usually 1-4 words) given by NIST, together with the training documents (usually 2). We used the BM25 tf.idf formula for scoring documents. Idf is initialized based on training data and updated over time as documents are filtered.

3.2 Profile Updating

At the very beginning when the number of training data is small, YFILTER has profile-specific anytime updating. That is, it updates a profile (threshold and scoring function) immediately whenever feedback, positive or negative, is available for that profile (Figure 1). After getting enough positive training examples (30 by default), the system begins to decrease the update frequency, updating the threshold only if:

- Current number of relevant documents delivered is 2*(number of relevant documents delivered at the last update), or
- Current number of non-relevant documents is 2*(number of non-relevant documents delivered at the last update), or
- Recent dangerous performance, which we define as 9 non-relevant documents delivered in a row.

3.2.1 Threshold Updating

We used the algorithm described in our SIGIR01 paper for threshold updating [9]. We provided a solution to optimize for F-beta measure based on our model. In case the system failed to find the optimal model, some heuristic were used to set the threshold.

3.2.2 Updating Terms and Term Weights

We tried the following two-term selection and term weight updating algorithms and compared their performance.

3.2.2.1 Language Model

Probabilistic language models, which are used widely in speech recognition and have shown promise for ad-hoc information retrieval. The strong theoretical foundation of language models enables us to build a variety of new capabilities. Current research on using language models for information retrieval tasks is focused on developing techniques similar to those used in speech recognition. However the differing requirements of speech recognition and information retrieval suggest that major adaptation of traditional approaches to language modeling is necessary to develop algorithms that will be highly accurate in the real world.

One research work in this direction is [4]. In their work, EM algorithm is used to find optimal relevance weights of each word that maximize the likelihood of joint probability of relevant document and query:

$$\prod_{j} P(d_j, q) = \prod_{j} P(d_j) P(q \mid d_j) \qquad (1)$$

In our work, we tried a different way of using language model. We propose a mixture of generative language models, which is more appropriate for the task of query expansion in information filtering and traditional ad-hoc retrieval task. As shown in Figure 1, we assume

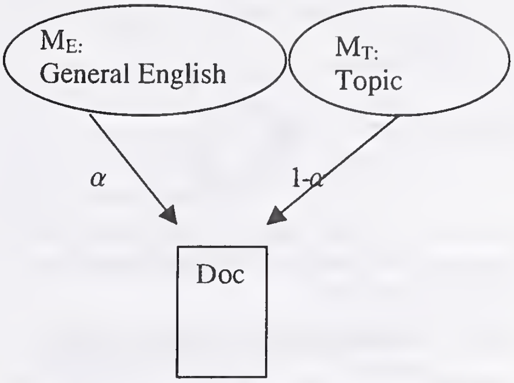


Figure 1 A mixture model to generate on topic documents

each relevant document is generated by a combination of two language models: A General English Model M_E, and a user-specific Topic Model M_T. For example, if the user is interested in "Star Wars", in a relevant document we expect words such as "is" and "the" will come from the general English model, and words such as "star" and "wars" will come from the topic model.

When doing user profile updating for query expansion, the filtering system will focus on learning M_T to find words that are very informative for deciding whether the document is on topic or not. Given a fixed value of α (usually a very high value, such as 0.95), we can train M_E to maximize the likelihood of all documents processed, and train M_T using the EM algorithm to maximize the likelihood of all the relevant documents processed. A sketch of the training algorithm was given in Figure 2.

This new mixture model will pick words where P(w| relevant) / P(w | general English) is very high. Because the task of profile updating is to provide a profile that can separate relevant documents from non-relevant documents, we believe such words are more discriminative, and thus are good candidates for being added to user profiles. Similar techniques are developed for ad-hoc retrieval independently [10].

Although both algorithms are called "Language Model" approach, our work and [4] are very different in that 1) Our optimization goal is to maximize the likelihood of training data (which is a widely used method when using MLE), while they are maximizing something else (Equation 1) 2) The parameter our algorithm estimates is the Topic model, which is a multinomial distribution, while the parameters [4] are estimating is relevance weights.

(1) Calculate the General English language model:

For each word w_i

$$P(w_i \mid M_E) = \frac{tf\_i}{\sum_{j\text{:all words in corpus}} tf\_j}$$

(2) Calculate the Topic language model using the EM algorithm to maximize the likelihood of all relevant documents

a. Initialize Topic language model as uniform

$$P(w_i \mid M_T) = 1/n$$

where n is the number of unique words that appear in the relevant docuements

b. EM step:

Iterate the following steps until changes on $P(w_i \mid M_T)$ are small enough for that iteration:

For each words in relevant documents:

$$T\_tf_i = rtf_i * \frac{(1-\alpha)*P(w_i \mid M_T)}{(1-\alpha)*P(w_i \mid M_T)+\alpha*P(w_i \mid M_E)}$$

$$P(w_i \mid M_T) = \frac{T\_tf_i}{\sum_{i\text{:all words in relevant documents}} T\_tf_i}$$

Figure2 Training Algorithm for Language Model

3.2.2.2 Rocchio

The Rocchio algorithm used this year is very similar to the one we used last year in TREC9. Each time a positive relevance feedback arrives (including those in the training data), all words in that document are added to the profile's candidate list of terms. Then the weight of each word in the candidate list is calculated according to the incremental Rocchio formula:

$$Rocchio = \alpha \cdot w_q + \beta \cdot w_{rel} - \gamma \cdot w_{non-rel} \qquad (1)$$

where

$w_q(t)$: max(term frequency of word t in original topics, 0.5)

$$w_{rel}(t) = \frac{1}{|rel\_set(t)|+1} \sum_{d \in rel\_set(t)} tf\_bel_{t,d} * idf_{t,b} \qquad (2)$$

$$idf_{t,d} = \log((C_d + 0.5)/df_{t,d})/\log(C_d + 1.0) \quad (3)$$

$$tf\_bel_{t,d} = tf_{t,d}/(tf_{t,d} + 0.5 + 1.5 \cdot (dl_d / avg\_dl_d)) \qquad (4)$$

The meanings of the above parameters are:

$tf_{t,d}$: Number of times term t occurs in document d

dl_d : Length of document d

C_d: Number of documents that arrived before document d

adv_dl_d : Average length of documents that arrived before document d

$rel\_set(t)$: Relevant documents after word t is added to the candidate list of the profile

α: 1; β: 3.5; γ: 2

In order to learn faster, β is set bigger than usual in the relevance feedback formula to emphasize the importance of relevant documents.

3.3 Hierarchical Category Structure

The filtering profiles correspond to Reuters categories, which are organized hierarchically. We assume that if a document belongs to a child category, it should also belong to the parent category. We used the following rules to take advantage of the hierarchical relationship between profiles when making the decision whether to deliver a document: If C_i is a child category of Cj, then:

- When a document d_t comes, we first consider whether it should be delivered to Cj, and if so, *then* consider whether it should be delivered to Ci.
- If the document d_t was delivered to Cj and the system received negative feedback, do not deliver d_t to Ci
- If d_t was not delivered to Cj previously, but was delivered to Ci and the system received a positive feedback, deliver d_t to Cj

By doing this, we get more training data for some of the profiles. For example, profiles 17, 34, and 45 get 8 instead of 2 relevant training documents to begin with, which is very helpful at the early stage.

The Reuters category assignments (i.e., the training data) are not consistent. Some documents are judged as relevant to a child category but non-relevant to the parent category. For example, documents 135639, 24269 26015 belong to R18 (DOMESTIC MARKETS) but do not belong to R17 (MARKETS/MARKETING). After reading those documents, we believe that they are in fact relevant to R17. It is well-known that human category assignments are not perfectly consistent, and any algorithm that uses them must compensate for noise in the training data. Using hierarchical structure of the categories helps to solve this problem to some extend.

4. ANALYSIS OF RESULTS

	Run 1	Run2	Run3	Run4
Profile Updating	Roc.	Roc.	LM	LM
Threshold Updating	ML	ML	ML	ML
Optimized for	T10SU	T10F	T10SU	T10F
T10SU	0.144	0.143	0.081	0.080
T10F	0.273	0.275	0.158	0.163
Profile>=Mean	55	41	32	21

Table 1: Submitted runs in TREC-9

We submitted 4 runs for the adaptive filtering task using Rocchio ("Roc") or language models ("LM") for profile updating, and our Maximum Likelihood Estimation for setting dissemination thresholds. The results are reported in Table 1. Compared with other groups, the results are not satisfying. Implementation errors were one cause, but we defer a discussion of their impact.

Figures 2 and 3 show the system performance over time for run 1. Precision and recall improve over time (Figure 3), but Utility decreases (Figure 4). This means the profiles (terms and term weighting) were improving as we got more training data while filtering, but unfortunately the threshold was set too high and got worse over time.

Despite the bugs and problems with the threshold, we can still analyse the performance of the Rocchio and language model methods of adding terms to profiles. According to Table 1, the simple Rocchio method works much better, which was a surprise. Our hypothesis was that the language model would work well; because the new language model approach does not need a stopwords list. However, the idf weights in the BM25 scoring method penalized words with high idf, thus allowing Rocchio to work well. Possible reason is BM25 scoring method used in Yfilter is good for Rocchio, but not good for language model. We probably should replace BM25 with KL divergence, which is a more natural scoring mechanism to measure distributional similarities.

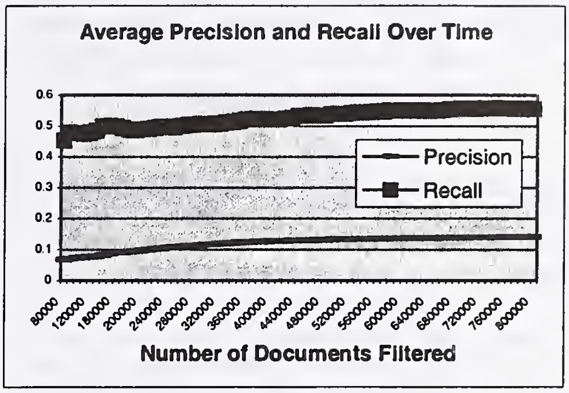


Figure 2 Filtering performances at different stages: Average Precision and Recall. (Run 1)

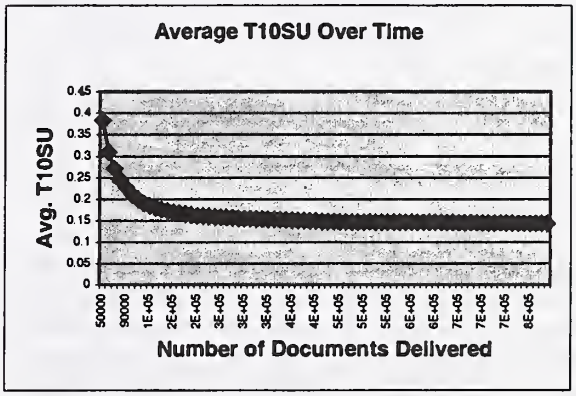


Figure 3 Filtering performances at different stages: T10SU Metric. (Run 1)

4.1 Score Density Distribution

We examined profile 77 on which our thresholding method did a very bad job. We used the learned final profile to score all of the documents in the corpus. Figure 4 shows the score density distribution of the relevant documents, which can be approximated by a normal distribution. Figure 5 shows the score density distribution of non-relevant documents that contain at least one profile term.

We found that using the exponential distribution to approximate the probability density function of non-relevant documents is problematic in this profile. A Beta distribution seems more appropriate. Since the exponential distribution is a special case of the beta distribution, using a beta distribution will also cover cases where the exponential distribution is right. Considering the maximum likelihood estimation method proposed in [9] does not require any specific distribution, we can plug the beta distribution into the general framework and find the optimal parameters. We have not yet implemented the algorithm to find the optimal beta distribution parameters. We simply observe that it may be a better approximation function than the exponential distribution proposed by [1] and used in our previous experiments [9]

4.2 Biased Training Problem for Profile Updating

We looked at the score distribution of profile 71 on which most of the systems did poorly. We fixed the profile terms and term weights (using the profile learned by the end) and scored all the documents in the corpus. We are surprised that the score density distribution of relevant documents looked more like an exponential distribution (Figure 6). Redrawing the score density distribution of the top scoring relevant documents for this profile (Figure 7) shows that the real distribution is actually like a mixture model of exponential and normal.

One possible explanation of this is the biased sampling problem [9]. In adaptive filtering, the user only provides feedback about documents delivered, so the training data is not sampled randomly, and the profile learned by the system is biased. The score density distribution of Profile 71 provides an extreme real case that illustrates this problem. Although we have proposed an algorithm to solve the sampling bias problem for threshold setting [9], we didn't develop a solution to solve the sampling bias problem when terms, term weights, and thresholds are all being adjusted simultaneously. One possible way is to deliver interesting "near miss" documents, so that the learning software gets a broader view of the surrounding information landscape. Theories in other research area, such as active learning (also known as experimental design) and reinforcement learning, are potentially useful considering the similarity of the tasks. Also, the bias problem for threshold learning and profile term updating are correlated and should be solved together. Another solution is to explicit modeling the sampling bias while profile term weights and threshold are changing, and more advanced analysis is needed.

4.3 Defects and explanation

Our results are disappointing on TREC10. There are several problems with the runs we submitted:

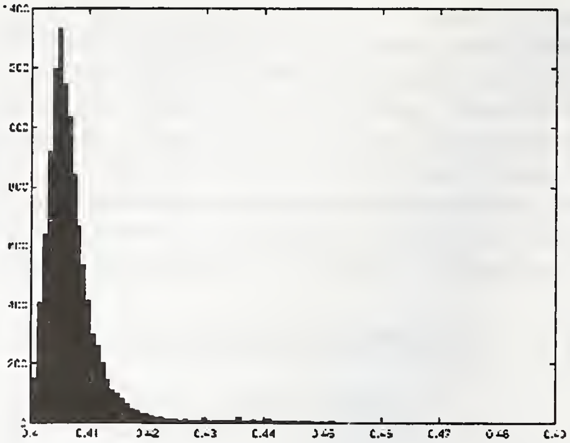


Figure 4 Score density distribution of relevant documents for profile 77.

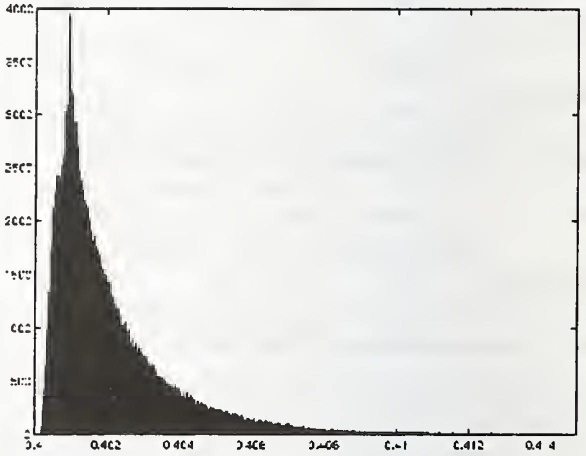


Figure 5 Score density distributions of non-relevant documents for profile 77.

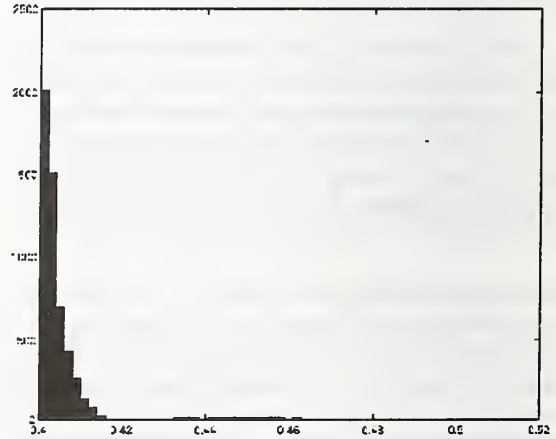


Figure 6 Score density distribution of relevant documents for profile 71.

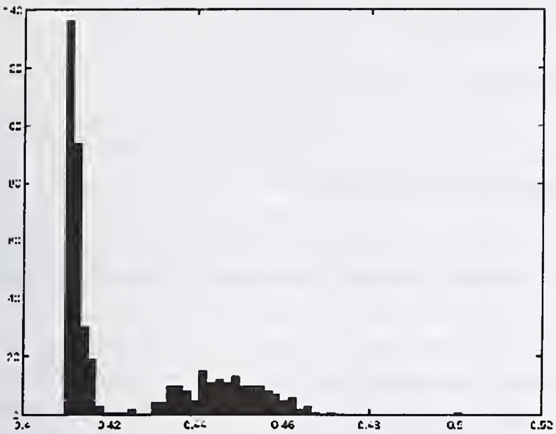


Figure 7 Score density distribution of top scoring relevant documents for profile 71.

(1) The methods deciding when to adjust terms and term weights was not integrated with threshold learning. Thus the system could change profiles without also adjusting thresholds to compensate for the changes in document scores.

(2) We used a heuristic rule to set thresholds when there is no solution based on maximum likelihood estimation. Unfortunately due to a programming error, we set the threshold too high (usually 1, which corresponded to delivering no document). Recovery speed was too slow.

(3) The words in the original profiles were stemmed before case conversion, but words in documents were stemmed after case conversion. The Porter stemmer is sensitive to case, so this difference produces inconsistent stemming in profiles and documents.

5. CONCLUSIONS

We tried a new profile-updating algorithm based on a mixture language model that we believe is more appropriate for the task of query expansion, and compared it with Rocchio. Compared with traditional language models, our new approach does select very discriminative words and requires no stop word list. The performance is encouraging. But what surprised us is how efficient (in terms of running time) and effective (in terms of performance) the old method of Rocchio is.

We noticed that the Beta distribution might be more appropriate for modeling the non-relevant document scores, although our previous experiments shows on some other dataset exponential distribution works well. We hypothesis that what kind of distribution to use is corpus/system dependant, although the Maximum Likelihood estimation we proposed does not require what kind of corpus to use, a real filtering should chose the right approximation function when applying our algorithm. We also noticed the effect of the sampling bias problem not only on profile threshold setting, but also on profile term weighting. Active learning and explicit modeling of the sampling bias while profile is changing are possible solutions for this problem.

6. ACKNOWLEDGEMENTS

This material is based on work supported by Air Force Research Laboratory contract F30602-98-C-0110. Any opinions, findings, conclusions or recommendations expressed in this paper are the authors', and do not necessarily reflect those of the sponsors.

7. REFERENCE

[1] Avi Arampatzis, A. Hameren. The Score-Distribution Threshold Optimization for Adaptive Binary Classification Task. *SIGIR01*

[2] J. Allan. 1996. Incremental Relevance Feedback for Information Filtering. *SIGIR96*

[3] J. Callan. 1998. Learning while Filtering. *SIGIR98*

[4] W. Kraaij, R. Pohlmann, D. Hiemstra. Twenty-One at TREC-8: using Language Technology for Information Retrieval. In *Proceeding of Eighth Text Retrieval Conference (TREC-8),* NIST.

[5] J. Lafferty, C. Zhai. 2001. Document Language Model, Query Models and Risk Minimization for Information Retrieval. *SIGIR01*

[6] M. F. Porter, 1980. An algorithm for suffix stripping.

[7] J. J. Rocchio. 1971. Relevance feedback in information retrieval in The SMART Retrieval System- Exp eriments in Automatic Document Processing, pages 313-323. Prentice Hall Inc.

[8] V. Rijsbergen. 1979 *Information Retrieval*

[9] Yi Zhang, J. Callan. Maximum Likelihood Estimation for Filtering Thresholds *ACM SIGIR01*

[10] Chengxiang Zhai, John Lafferty. Model-based Feedback in the Language Modeling Approach to Information Retrieval. In *Proceedings of Tenth International Conference on Information and Knowledge Management*

kNN, Rocchio and Metrics for Information Filtering at TREC-10[1]

Tom Ault TOMAULT@CS.CMU.EDU
Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Yiming Yang YIMING.YANG@CS.CMU.EDU
Language Technologies Institute & Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract

We compared a multi-class k-nearest neighbor (kNN) approach and a standard Rocchio method in the filtering tasks of TREC-10. Empirically, we found kNN more effective in batch filtering, and Rocchio better in adaptive filtering. For threshold adjustment based on relevance feedback, we developed a new strategy that updates a local regression over time based on a sliding window over positive examples and a sliding window over negative examples in the history. Applying this strategy to Rocchio and comparing its results to those by the same method with a fixed threshold, the recall was improved by 37-39% while the precision was improved by as much as 9%. Motivated by the extremely low performance of all systems on the $T10S$ metric, we also analyzed this metric, and found that it favors more frequently occurring categories over rare ones and is somewhat inconsistent with its most straightforward interpretation. We propose a change to this metric which fixes these problems and brings it closer to the C_{trk} metric used to evaluate the TDT tracking task.

1. Introduction

We participated in the TREC-10 information filtering track, submitting results of a standard Rocchio method and one variant of our k-nearest neighbor (kNN) algorithms[13] for the batch and adaptive filtering tasks. Our goals for this year's TREC were twofold:

1. To establish a performance baseline for text categorization on the new Reuters corpus.
2. To develop an effective adaptive thresholding technique for the adaptive filtering subtask.
3. To investigate the properties of the $T10S$ metric using the isocurve analysis in precision-recall space we developed for the TREC-9 metrics[1].

Item (1) was motivated by the success of kNN on previous versions of the Reuters corpus[10, 9] and is addressed by our batch filtering results, for which we applied the Avg2 variant of kNN. Item (2) was motivated by our difficulties in developing an effective adaptive thresholding method for TREC-9, and is addressed by our new *margin-based local regression* technique of adaptive thresholding. We were successful in both endeavors, ranking third of eighteen runs in the batch-filtering task and second of thirty runs in the adaptive filtering task for both the F_β and $T10S$ metrics[2]

Item (3) was motivated by the over 50% drop in performance for the $T10S$ metric when moving from the validation to the test set, compared to a drop of only 27% for the F_β metric. This observation combined with an analysis of the $T10S$ metric in terms of its isocurves in precision-recall space lead to the discovery of some inconsistencies between the user behavior that the $T10S$ metric appears to model and what it actually models. We propose a fix for $T10S$, and compare the modified metric to the weighted tracking cost (C_{trk}) metric used for the TDT tracking task.

This paper has five sections past the introduction. Section 2 reports the experiments with our kNN and Rocchio systems in batch filtering. Section 3 compares kNN and Rocchio in adaptive filtering, and introduces our novel approach for adaptive thresholding. Section 4 analyzes the potential problems inherent in the $T10S$ metric, suggests a minor alteration that resolves these problems, and discusses the relationships between the modified and unmodified $T10S$ and the F_β and C_{trk} metrics. Section 5 presents our conclusions and future research goals for information filtering.

2. Batch Filtering

We applied our kNN system and our implementation[13] of a standard Rocchio method to this task to compare these two methods.

[1] Authors' names are in alphabetical order.

[2] Rankings were computed by the authors across all runs submitted to the TREC-10 filtering tasks from official per-category, per-run performance data supplied by the filtering track coordinators; these rankings are not official.

2.1 K-Nearest Neighbor (kNN)

kNN, an instance-based classification method, has been an effective approach to a broad range of pattern recognition and text classification problems [2, 8, 10, 13]. In contrast to "eager learning" algorithms (including Rocchio, Naive Bayes, Decision Trees, etc.) which have an explicit training phase before seeing any test document, kNN uses the training documents "local" to each test document to make its classification decision on that document. Our kNN uses the conventional vector space model, which represents each document as a vector of term weights, and the distance between two documents is measured using the cosine value of the angle between the corresponding vectors. We compute the weight vectors for each document using one of the conventional TF-IDF schemes [4], defined as:

$$w_d(t) = (1 + \log_2 n(t,d)) \times \log_2(|\mathcal{D}|/n(t)) \qquad (1)$$

where $n(t,d)$ is the within-document frequency of term t in document d, $n(t)$ is the total document frequency of term t in document set D.

Given an arbitrary test document d, the kNN classifier assigns a *relevance* score to each candidate category (c_j) using the following formula:

$$s(c_j, d) = \sum_{d' \in R_k(d) \cap \mathcal{D}_j} \cos(w_d, w'_d) \qquad (2)$$

where set $R_k(d)$ are the k nearest neighbors (training documents) of document d. By sorting the scores of all candidate categories, we obtain a ranked list of categories for each test document; by further thresholding on the ranks or the scores, we obtain binary decisions, i.e. the categories above the threshold will be assigned to the document. There are advantages and disadvantages to different thresholding strategies [12].

2.2 Rocchio

Rocchio is an effective method using relevance judgments for query expansion in information retrieval[3, 5], and the most common (and simplest) approach to the filtering tasks in TREC[14].

The standard Rocchio formula computes a vector as the *prototype* or *centroid* of a class of a documents. Given a training set of documents with class labels, the *prototype* vector is defined to be:

$$\vec{c}(\gamma, k) = \frac{1}{|R_c|} \sum_{u_i \in R_c} \vec{u}_i - \gamma \frac{1}{\bar{R_{c,k}}} \sum_{v_i \in \bar{R_{c,k}}} \vec{v}_i \qquad (3)$$

where R_c is the set of positive training examples, $\bar{R_{c,k}}$ is the "query zone"[6], that is, the k top-ranking documents retrieved from the negative training examples when using the centroid of the positive training examples as the query. To increase the efficiency of computation, we retain only the top p_{max} components of the prototype vector. The values of γ, k (the size of the local zone) and p_{max} are the prespecified parameters for the Rocchio method.

In the filtering process, Rocchio computes the cosine similarity between each test document and the prototype of every category, where the prototype is updated over time using the past documents whose category labels are available through relevance feedback. Thresholding on these scores yields binary decisions on each document with respect to every category.

2.3 Batch Filtering Results

In our experiments with Rocchio and kNN, we defined a token to be the longest possible sequence of letters and digits, followed by an optional "'s" or "'nt". Tokens which were purely numbers were discarded, as were common English words found on a stop word list. Tokens were stemmed with the Porter stemmer and assigned weights according to equation 1 above. Per-category thresholds for binary decision making were set by five-fold cross-validation on the training data.

We submitted two sets of results, labelled "CMUCATa2f5" and "CMUCATa210", for batch filtering; the former is optimized for the F_β metric and the latter the $T10S$ metric. Both runs used the kNN.avg2 method with $kp = 200$ (number of nearest neighbors which are positive examples of the category) and $kn = 500$ (number of nearest neighbors which are negative examples of the category), since this method and parameter settings had the best performance during cross-validation for both metrics. Table 1 summarizes the results.

Based on previous experience with the Reuters-21578 and OHSUMED corpora[9, 11], we applied a variety of feature selection methods, including document frequency, mutual information, information gain, and chi-square. None of them produced any significant improvement in the performance of our system on the Reuters 2001 corpus. Why we should see no improvement on this corpus while we see considrable improvement on the other corpora requires further investigation.

3. Adaptive Filtering

Our research strategy consists of two parts:

- analyzing the scores generated by kNN and Rocchio over time, to see which method produces more discriminatory scores for separating positive and negative examples of a category; and

- using *margin-based local regression* (our new approach) to track the potential shift of the optimal threshold for each category over time.

RUN ID	RECALL	PREC	$T10S$	F_β	RANK-$T10S$	RANK-F_β
CMUCATa2f5	0.322	0.719	0.287	0.511	7/18	3/18
CMUCATa210	0.358	0.618	0.324	0.489	3/18	7 or 8/18

Table 1. Results by CMU-CAT for Batch filtering

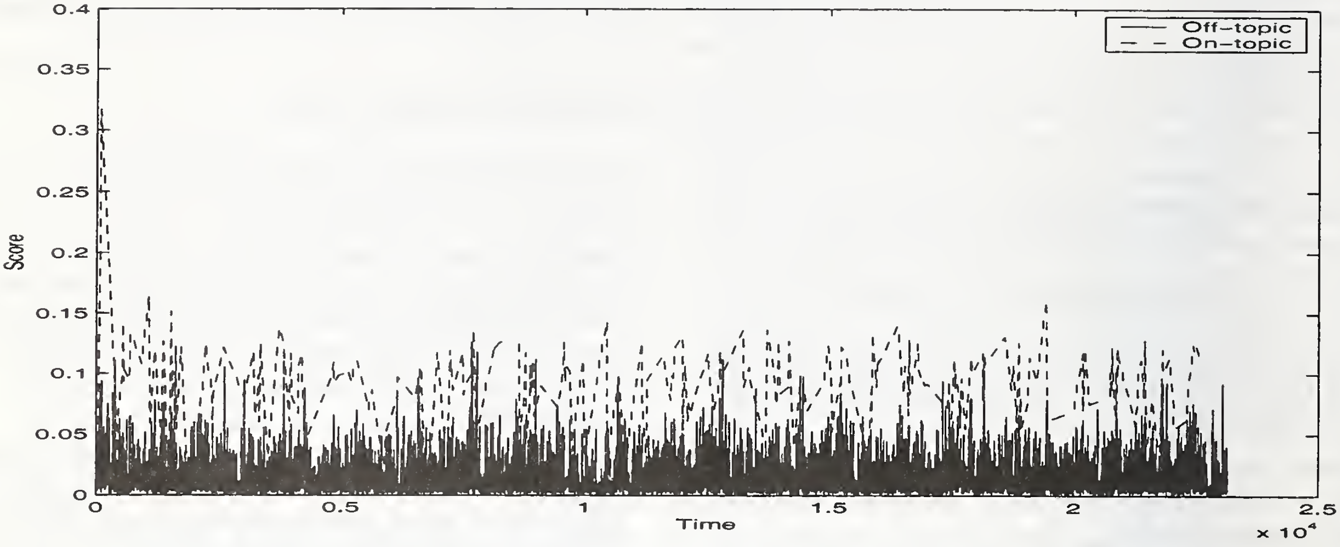


Figure 1. Scores vs. time for on- and off-topic documents by Rocchio for category R83

3.1 Score analysis for Rocchio and kNN

Figures 1 and 2 show the scores for category R83 ("METALS TRADING") generated by kNN and Rocchio during the adaptive filtering process where the relevance judgment for each test document was made available to the system after that document is scored, regardless what decision (Yes or No) the system made for that document; the category prototype was updated accordingly per test document. We used the TREC-10 training corpus for this experiment by splitting the data in to the training and test halves and then running the systems on these data. The scores generated in such a process, obviously, are better than those kNN and Rocchio would generate under the condition required in TREC for relevance feedback, i.e. relevance judgments are available for a system only for the documents which the system make a Yes decision upon. Nevertheless, those figures allow us to get a rough idea about a major difference between the scores generated by our two systems. For Rocchio, there is clear separation on average between the scores for the two classes (Yes and No) over time, while for kNN, the scores for the two classes are will blended. This means that finding the optimal thresholding function over time using the scores generated by Rocchio would be a much easier task than thresholding over the scores generated by kNN. Also, the average scores for each class by Rocchio seem to be constant in different time intervals; however, there is a visible trend (increasing in value over time) in the average scores by kNN, at least for the particular category being observed. In fact, we compared pairwise figures for kNN and Rocchio over all the 84 categories selected from Reuters 2001 for TREC-10, and observed similar patterns with most categories in the TREC-10 filtering training corpus.

These empirical findings were rather suppressing to us, because we have found kNN to perform better than Rocchio in batch filtering and conventional text categorization[9, 10]. On the other hand, we have also found Rocchio works surprising well (comparable or just slightly worse than kNN) for the event tracking task in the domain of Topic Detection and Tracking (TDT)[12], which is similar (but not identical) to adaptive filtering, in the sense that both processes start with a small number of training examples per class. Why does Rocchio perform worse than kNN in batch filtering but better in adaptive filtering? We do not have a satisfactory interpretation for this question at this point; deeper understanding about this invites future research.

3.2 Margin-based local regression

Here we propose a novel approach, namely *margin-based local regression*, for predicting optimal thresholds over time. The intuition is rather simple: if we have two streams of scores (one for previously-classified positive examples and the other for previously-classified negative examples for a particular class), and if the two streams are separable in value (Figure 1) in any particular time interval, then we would choose some values inside of the *margin* between the two streams as the thresholds, where by margin we mean the difference between the minimal score for positive examples

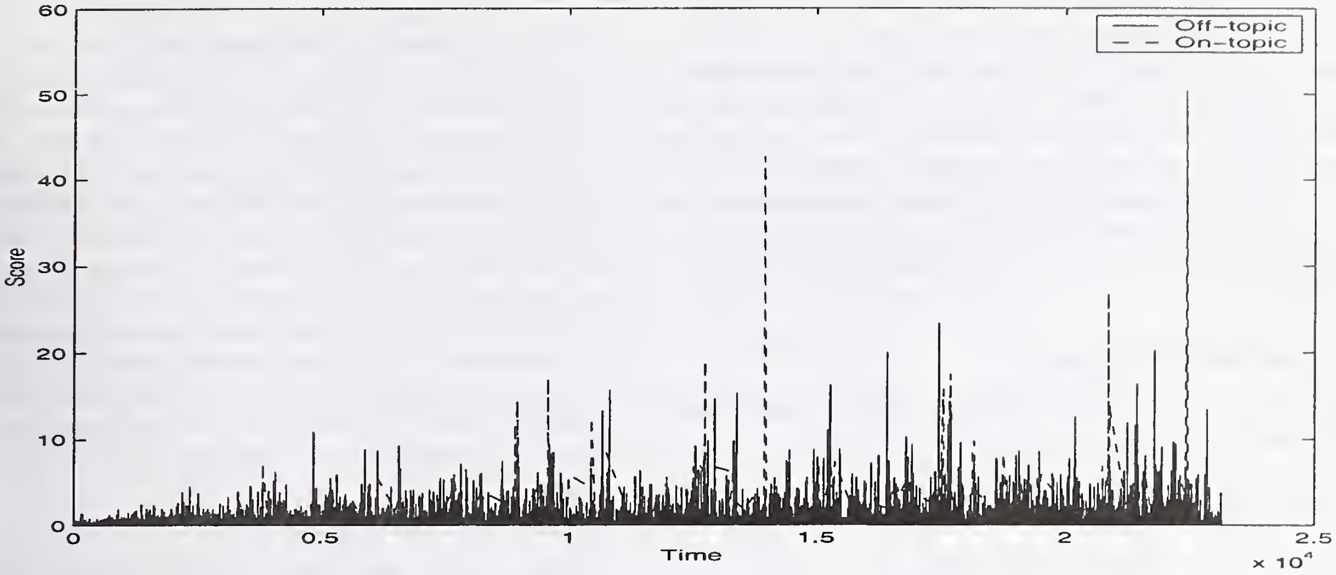


Figure 2. Scores vs. time for on- and off-topic documents by unnormalized kNN for category R83

and the maximal score for negative examples in a particular time interval. One can think of a waving band with both its center location and width changing over time in a two dimensional space of magnitude verses time. We want to use local regression to track the shift of the "optimal" threshold over time inside of the band.

This simple description is sufficient to gain intuition into the method, but is not sufficient for an accurate definition. First, we do not assume that the positive and negative streams are always separable, but artificially make those streams separable by excluding outliers. That is, for the positive stream, our system only includes the truly positive examples for which the system predicted YES. All other examples, including the false-alarms, misses and documents for which the system correctly predicted NO, form the negative stream; our system unfortunately cannot exclude the misses because the relevance judgments for those examples are not available during the relevance feedback. Second, we need to relax the definition of margin, to avoid making our method overly sensitive to outliers. Third, we need to be precise about what is "optimal", discussing the concern about risk minimizing. Fourth, we need specify the local regression for tracking the shift of optimal thresholds over time.

The precise definition of our approach uses the following notation:

- $X = \{x_1, x_2 \dots x_{k_+}\}$ is the sliding window of scores for the k_+ (at most) most recent positive examples for which the system predicted YES;
- $Y = \{y_1, y_2 \dots y_{k_-}\}$ is the sliding window of scores for the k_- (at most) false-alarms, misses, and documents for which the system correctly predicted NO;
- $\mu_x(t) = a_1 t + b_1$ is the local regression obtained by fitting a line over the data points in sample X, where a_1 and b_1 are the regression coefficients;
- $\mu_y(t) = a_2 t + b_2$ is the local regression obtained by fitting a line over the data points in sample Y, where a_2 and b_2 are the regression coefficients;
- $\delta(t) = \mu_x(t) - \mu_y(t)$ is the local *margin* (the Mean-Mean version; see the next section);
- $\theta(t) = \mu_y(t) + \eta\delta(t) = at + b$ is the local regression for optimal thresholding, where

$$a = \eta a_1 + (1 - \eta) a_2$$

$$b = \eta b_1 + (1 - \eta) b_2.$$

This method has five parameters, k_+, k_-, η, Δ_+ and Δ_- which are empirically chosen (through cross validation). We purposely allow k_+ and k_- to take different values (instead of a single parameter k) so that we can empirically tune the window sizes to be sufficient sensitivity to the local trends for both positive examples and negative examples. As for parameter η, it allows us to adjust the position (instead a fixed position, such as the middle) of the thresholding function between the margin, in order to overcome the inductive bias of the system (if any) and to optimize the performance with respect to different evaluation metrics ($T10S$, F_β, or the like) through cross validation. Δ_+ and Δ_- are the number of documents the window must slide through before the positive and negative margins are updated; in our TREC-10 results, we updated both margins with every document ($\Delta_+ = \Delta_- = 1$).

For initialization, we set $a_0 = a_1 = 0$ and set b_0 and b_1 to return the top 1% of documents in the validation set for each category. Early on, when there is not sufficient data to reliably compute the margins (defined in terms of number of documents within the window and parameterized by min_- and min_+ for the positive and negative windows respectively), we apply the following hueristic to set the threshold to a reasonable value without drawing too many false- alarms or misses: if both the positive window and the negative window have less than min_- and min_+ documents, the threshold is set to just above the score of the last false-alarm observed.

3.3 Variants of margins

We propose several versions of margin as the variants of our approach:

1. *Min-Max* margin:

$$x = arg\min\{x_1, x_2 \dots x_{k+}\}$$

$$y = arg\max\{y_1, y_2 \dots y_{k-}\}$$

$$\delta(t) = x - y$$

$$\theta(t) = \eta x + (1-\eta)y.$$

2. *Mean-Mean* margin:

$$\mu_x(t) = a_1 t + b_1$$

$$\mu_y(t) = a_2 t + b_1$$

$$\delta(t) = \mu_x(t) - \mu_y(t)$$

$$\theta(t) = \mu_y(t) + \eta\delta(t)$$

3. *MinK-MaxK* margin:

   X' is the bottom n_+ data points in $X = \{x_1 \dots x_{k+}\}$,
   Y' is the top n_- data points in $Y = \{y_1 \dots y_{k-}\}$,
   $\mu'_x(t) = a_1 t + b_1$ is the linear fit to X';
   $\mu'_y(t) = a_2 t + b_2$ is the linear fit to Y';

$$\delta'(t) = \mu'_x(t) - \mu'_y(t)$$

$$\theta(t) = \mu'_y(t) + \eta\delta'(t)$$

   where n_+ and n_- are pre-specified parameters.

4. *MeanVar-MeanVar* margin:

$$\delta(t) = (\mu_x(t) + \alpha\sigma_x) - (\mu_y(t) + \alpha\sigma_y)))$$

$$\theta(t) = \mu_y(t) + \eta\delta(t)$$

   where σ_x is the standard deviation of X, σ_y is the standard deviation of Y, and α is a pre-specified parameter.

5. Other combinations, e.g., *Mean-MaxK*:

$$\delta''(t) = \mu_x(t) - \mu'_y(t)$$

$$\theta(t) = \mu'_y(t) + \eta\delta''(t)$$

The Min-Max margin is the simplest, but likely to be over-sensitive to outliers and under-sensitive to the trend of the margin within a window. The Mean-Mean margin is the one we introduced in the previous section, which is less sensitive to extreme values than Min-Max. The MinK-MaxK has an sensitivity between the previous two, with additional (ad-hoc) parameters; in fact, Min-Max is just a specific case of MinK-MaxK in which $n_+ = n_- = 1$. MeanVar-MeanVar take the densities of data points on both sides (the positive side and the negative side) in to consideration, which would be more powerful than Mean-Mean but assumes normal distribution of the scores for the positive and negative examples and requires more data for the estimation of the variances. There are other possible variants along this line; we do not intend to give an exhaustive list.

3.4 Adaptive filtering results

We chose Rocchio over kNN for adaptive filtering. Tables 2 and 3 describe our submitted runs and the results, including one run (CMUCATmrf5) using Mean-Mean margin (as our primary submission), one run (CMUCATmr10) using the Mean-MaxK margin, and two runs for the baseline Rocchio (CMUCATsrf5 and CMUCATsr10) in which the the prototypes were adaptive but the thresholds were fixed. The last two runs were generated as baselines for comparisons with the margin-based adaptive filtering methods.

In addition to the submitted runs, we also tested other versions of the margins (MinK-MaxK, for example). We found the Mean-Mean method with the best results in cross validation over the TREC-10 training corpus. The Mean-MaxK, however, performed better on the evaluation data, suggesting that that variant tends to have a large performance variance.

We were surprisingly pleased by the improvements by the margin-based regression over the baseline Rocchio with a fixed threshold. Under the same condition of optimizing F_β, the Mean-Mean method improved the performance over the baseline by 37.5% (from 24.8% to 34.1%) in recall and 0.5% (from 65.8% to 66.1%) in precision. Under the condition of optimizing $T10S$, the Mean-MaxK improved the performance over the baseline by 38.7% (from 24.8% to 34.4%) in recall and 9.2% (from 60.3% to 65.7%) in precision. We are also surprised that the Rocchio baseline with a fixed threshold worked very well, being ranked at the top four among 30 submissions in both $T10S$ and F_β measure.

It is worth mentioning that the margin-based local regression approach is not a part of the Rocchio method. Instead, it can be applied to the output of any system as long as the average of scores for positive examples by that system are higher than the average of the scores for negative examples, and as long as there is some continuous trends over time in the margins. An interesting point is, when we designed this method and until our submission to TREC-10, we only tested Rocchio under the condition of complete relevance feedback (and did not have the time to run it under more re-

RUN ID	DESCRIPTION
CMUCATsrf5	Adaptive filtering, Rocchio ($\gamma = -1.5$,$k = 200$,$p_{max} = 500$), using fixed threshold, optimized for F_β
CMUCATsr10	Adaptive filtering, Rocchio ($\gamma = -1.5$,$k = 200$,$p_{max} = 500$), using fixed threshold, optimized for $T10S$
CMUCATmrf5	Adaptive filtering, Rocchio ($\gamma = -1.5$,$k = 200$,$p_{max} = 500$), using margin-based thresholding with means for both margins, optimized for F_β
CMUCATmr10	Adaptive filtering, Rocchio ($\gamma = -1.5$,$k = 200$,$p_{max} = 500$), using margin-based threshold with lower margin computed from median of top 20 negative examples and higher margin computed from mean of positive margin, optimized for $T10S$.

Table 2. Official submissions by CMU-CAT for the Adaptive filtering task

RUN ID	RECALL	PREC	$T10S$	F_β	RANK by $T10S$	RANK by F_β
CMUCATsrf5	0.248	0.658	0.211	0.467	7/30	4/30
CMUCATsr10	0.248	0.603	0.228	0.415	4/30	6/30
CMUCATmrf5	0.341	0.661	0.251	0.489	3/30	3/30
CMUCATmr10	0.344	0.657	0.263	0.499	2/30	2/30

Table 3. Results by CMU-CAT for Adaptive Filtering

alistic settings of relevance feedback); under that condition we did not found dynamic trends in the margins among the scores by Rocchio. We took this approach anyway because it was rational. The strong results, 37.5-38.7% improvement in recall while precision improved in the same direction over Rocchio baseline, suggest that, perhaps, there were indeed dynamic trends in the margins that worth tracking.

4. Metrics

	$T10S$	F_β
Batch filtering		
Minimum	0.081	0.154
Mean	0.239	0.429
Maximum	0.414	0.606
Adaptive filtering		
Minimum	0.015	0.046
Mean	0.134	0.266
Maximum	0.291	0.519

Table 4. Macro-average performance summary for the TREC-10 filtering task

The adaptive and batch filtering tasks for this year used two metrics: $T10S$ and van Rijsbergen's F_β[7], which are defined as with respect to a category C as:

$$T10S(C) = \frac{max(2A - B, minU) - minU}{2 * N_+ - minU} \tag{4}$$

$$F_\beta(C) = \frac{(\beta^2 + 1)A}{A + B + \beta^2 N_+} \tag{5}$$

where A is the number of documents correctly assigned to C, B is the number of documents incorrectly assigned to C (aka false-alarms), N_+ is the number of documents relevant to C, $minU$ is the lower bound on the unscaled utility $(2A - B)$, and β is a constant that specifies the relative weight between recall and precision for F_β. Both $T10S$ and F_β are scaled to fall between 0 and 1, and for TREC-10, $minU$ was fixed at -100 and β at 0.5 for all categories. The overall performance of the system for the task was obtained by computing the unweighted average across all categories (called the *macro-average* in the information retrieval literature).

The most straightforward interpretation of the $T10S$ metric is that it computes the return the user receives in terms of information gained vs. effort expended in reading the documents assigned by the filtering system to a particular category, scaled relative to the range of possible returns, where 1.0 represents maximum information gain with minimum effort, and 0.0 represents the point at which the effort required in reading the documents for the category in question so exceeds the information gained that the user regards any information contained in those documents as worthless. The F_β metric does not have such a straighforward interpretation in terms of the preferences of a particular user, but is instead the weighted harmonic average of recall [3] and precision[4] over the set of documents assigned to a category.

An examination of table 4 shows that systems participating in the batch and adaptive filtering tasks performed much worse on $T10S$ than on F_β. Does the $T10S$ measure, in fact, describe a harder task than the F_β measure or are there

[3] *Recall is defined for a category as the ratio of documents correctly assigned to that category to the total number of documents relevant to that category, e.g.* $r = \frac{A}{N_+}$.

[4] *Precision is defined for a category as the ratio of documents correctly assigned to the category to the total number of documents assigned to that category, e.g.* $p = \frac{A}{A+B}$.

other factors at work which would cause this performance gap? In the following section, we analyze the properties of the $T10S$ metric and find that, while $T10S$ is a definite improvement over $T9U$, it still has an undesirable characteristic that biases it against frequently-occuring categories. We propose a minor modification to $T10S$ which fixes the undesirable properties and brings it closer to the tracking cost (C_{trk}) metric used in the TDT evaluations.

4.1 T10S

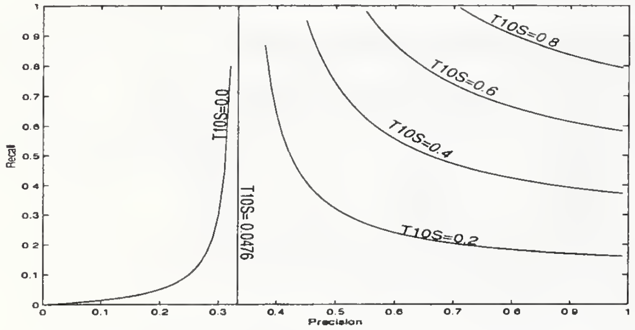


Figure 3. Isocurves of T10S for $\alpha = -0.1$

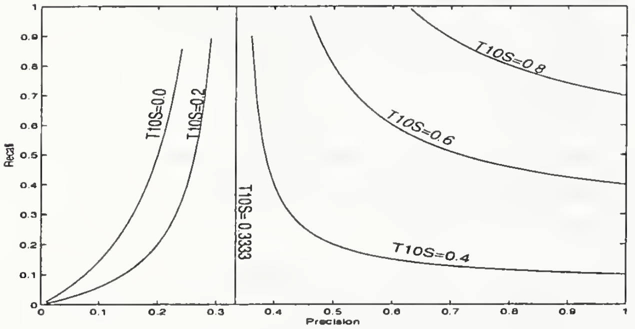


Figure 4. Isocurves of T10S for $\alpha = -1$

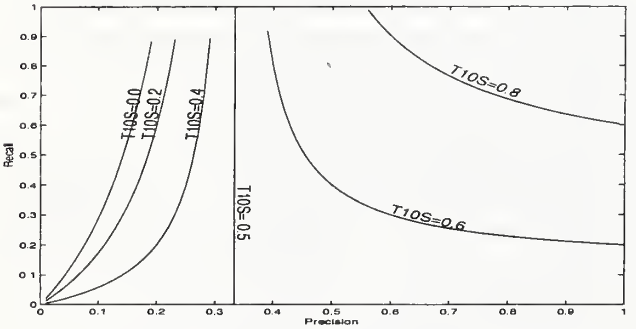


Figure 5. Isocurves of T10S for $\alpha = -2$

$T10S$ is a scaled version of the linear utility metric used in TREC-9 ($T9U$). This scaling addressed some of the problems of $T9U$ (which are discussed in detail in [1]), specifically, the wide variation in the maximum value of $T9U$ with the number of relevant documents for the category, which makes the macro-average of $T9U$ difficult to interpret and causes performance on common categories to dominate the average. However, the scaling introduced another problem, which can be seen if $T10S$ is written in terms of recall(r) and precision(p):

$$T10S = \begin{cases} \frac{max(r * \frac{3p-1}{p}, \alpha) - \alpha}{2-\alpha} & \text{if } p > 0, \alpha < 2 \\ \text{Some value in } [0, \frac{\alpha}{2-\alpha}] & \text{if } p = 0, \alpha < 2 \end{cases} \qquad (6)$$
$$\alpha = \frac{minU}{N_+}$$

The isocurves of $T10S$ for $\alpha = -0.1$, $\alpha = -1$ and $\alpha = -2$ are shown in figures 3 through 5. The biggest problem with $T10S$ is that the locations of its isocurves are dependent on the number of examples of the category in question, giving two different categories with the same relative performance different $T10S$ values. In particular, the larger the number of relevant documents for a category, the lower it's $T10S$ score for the same relative performance compared to a category with fewer documents. Given the large number of categories with 5,000 or more relevant documents in the test set used for TREC-10, it's not surprising that the scores for this metric were significantly lower than those for F_β.

Another problem with $T10S$ is that it's lower bound of $minU$ is applied before the metric is scaled. This means that the user's tolerance for poor-performing profiles (as modeled by $T10S$) also varies with the number of relevant documents for the category. In particular, the user is far more tolerant of poor performance (in relative terms) for categories with fewer documents. Furthermore, the value of $minU$ controls not only the user's tolerance for poor performance but also how sensitive the locations of the isocurves are to changes in the value of N_+; the larger the magnitude of $minU$, the less sensitive $T10S$ is to variations in N_+. This linkage between the minimum return the user is willing to accept on his or her investment in reading documents before he or she gives up and the preference of the user for categories with fewer relevant documents is counterintuitive and not obvious from the form of the $T10S$ metric itself.

One could dismiss these objections to $T10S$ by claiming that it is not necessary for $T10S$ to have consistent properties across categories with respect to recall and precision, since $T10S$ is not based on those metrics. However, having inconsistent properties with respect to recall and precision leads to inconsistencies within the $T10S$ metric itself if we take the most straightforward interpretation described in section 4. Under this interpretation, each document on average requires equal effort to read and provides the same amount of information, which are reasonable assumptions given that the TREC filtering tasks only supply binary relevance judgements and no judgements about the effort required to read a document. Furthermore, the TREC filtering tasks make no distinctions among categories as to which might be more or less important to the user. These conditions imply that a reasonable user should expect to spend more effort reading documents for categories that occur more frequently than for those that occur less frequently, and that the metric used to model the user should take this into account. Moreover, two sets of documents assigned to two different categories with different occurence frequencies which have the same relative amount of information (e.g. same recall) and require the same relative effort to extract that information (e.g. same precision), should be regarded as equally useful to the user, since all documents require the same effort, all relevant documents have the same information content, and the user expects to spend more effort on the more frequently-occuring category. In violating this latter principle that equal recall and precision should yield equal utility, $T10S$ implicitly assumes that the user favors rare categories over more common ones, that the utility of relevant documents decreases as more of them are found while effort to read them remains the same, or that the effort required to read a document decreases in proportion to the number of relevant documents for a category while the utility of relevant documents remains the same. None of these latter assumptions are consistent with the straightforward interpretation of $T10S$ described above, or with the fact that the TREC filtering tasks make no explicit distinctions between categories.

4.2 Normalized Filtering Utility

Given that outside of its variable properties in precision-recall space, $T10S$ is otherwise a good metric with an understandable user model, one wonders if it might be possible to correct for these problems while still preserving it's understandable user model and hyperbolic isocurves in precision-recall space. We can gain insight in how to do so if we consider another linear metric: unnormalized tracking cost (C_{trk}) which is used for the TDT tracking task (which is similar in many ways to the TREC filtering tasks). The value of C_{trk} is defined for category C as:

$$\begin{aligned} C_{trk}(C) &= C_{miss} * P_{on} * P_{miss} + C_{fa} * P_{off} * P_{fa} \\ &= C_{miss} * P_{on} * (\frac{N_{+} - A}{N_{+}}) + C_{fa} * P_{off} * (\frac{B}{N - N_{+}}) \end{aligned} \tag{7}$$

where:

- C_{miss} and C_{fa} are the relative costs of a miss (relevant document not assigned to C) and a false-alarm respectively
- P_{miss} and P_{fa} are the conditional probabilities of a miss or a false-alarm occuring, given that the document is relevant or not relevant to C respectively
- P_{on} and P_{off} are the prior probabilities that a document is relevant or not-relevant to C. $P_{on} + P_{off} = 1$ naturally.
- A, B, and N_{+} are the number of correct assignments, false-alarms, and documents relevant to C respectively.
- N is the total number of documents in the corpus.

In TDT, the values of P_{on} and P_{off} are fixed to their prior probabilities of 0.02 and 0.98 respectively for all categories in the tracking task. However, if we replace these values with their posterior probabilities (e.g. $P_{on} = N_{+}/N$ and $P_{off} = (N - N+)/N$, then equation 7 becomes:

$$C_{trk}(C) = C_{miss} * \frac{N_{+} - A}{N} + C_{fa} * \frac{B}{N} \tag{8}$$

Written in terms of recall and precision, this form of C_{trk} is

$$C_{trk}(C) = \frac{N_{+}}{N}(C_{miss} * (1 - r) + C_{fa} * r(\frac{1-p}{p})), p \neq 0 \tag{9}$$

This immediately suggests that by normalizing C_{trk} by P_{on}, we can obtain a version of C_{trk}, designated C'_{trk} which is stable in precision-recall space. Written in terms of A, B, and N_{+}, C'_{trk} is:

$$\begin{aligned} C'_{trk}(C) &= \frac{N}{N_{+}}(C_{miss} * \frac{N_{+} - A}{N} + C_{fa} * \frac{B}{N}) \\ &= C_{miss} + \frac{C_{fa} * B}{N_{+}} - \frac{C_{miss} * A}{N_{+}} \end{aligned} \tag{10}$$

If we subtract C'_{trk} from C_{miss} (which is equivalent to flipping the scale and moving the zero point), scale by $1/C_{miss}$ so that the upper bound becomes 1, and rename C_{miss} to C_{corr}, we get the following normalized linear utility metric, which we call *normalized filtering utility* and designate U_f:

$$\begin{aligned} U_f(C) &= \frac{C_{corr} * A - C_{fa} * B}{C_{corr} * N_{+}} \\ &= \frac{C_{corr} * P_{on} * P_{corr} - C_{fa} * P_{off} * P_{fa}}{C_{corr} * P_{on}} \\ &\doteq r(1 - (\frac{C_{fa}}{C_{corr}})(\frac{1-p}{p})) \end{aligned} \tag{11}$$

U_f is essentially an unbounded $T10S$. We derrive it in this fashion to emphasize both it's connection to the C_{trk} metric used in TDT and its theoretical justification in terms of being a weighted combination of the conditional probabilities of correctly and incorrectly identifying relevant documents. Unlike $T10S$, U_f has consistent isocurves in precision-recall space, and thus it's straighforward interpretation as measuring the trade-off between effort expended and information gained is consistent with what it actually measures.

As an unbounded metric, U_f suffers from the weakness that poor-performing categories can dominate the macro-average. We address this by limiting the limiting the lower value of U_f to $U_{f,min}$. Like $minU$ for $T10S$, $U_{f,min}$ represents the lowest return on reading the set of documents assigned a category the user is willing to accept before he or she regards that set as worthless, but since $U_{f,min}$ is applied after normalization, the tolerance of the user for poor performance by the filtering system remains consistent from category to category. We can now scale U_f to fall between 0 and 1 by:

$$U'_f = \frac{max(U_f, U_{f,min}) - U_{f,min}}{1 - U_{f,min}} \tag{12}$$

where 1 represents maximum information gain with minimum effort, and 0 represents the point where the documents become worthless.

4.3 Comparison of Metrics

As an example, figure 6 plots F_β and $T10S$ vs. precision across all runs submitted to the adaptive and batch filtering subtasks for category R15. As we expect from our analysis in section 4.1, F_β and $T10S$ are correlated when precision is greater than or equal to 1/3, since the isocurves of $T10S$ in this region have a similar shape to the isocurves for F_β and thus a strategy that maximizes $T10s$ is also likely to maximize F_β and vice-versa, but are uncorrelated when precision is less than 1/3. Note that because most runs have a precision above 1/3 for most categories, the macro-average F_β and $T10S$ for each run will appear to track each other, even though the metrics are not necessarily correlated.

	Batch		Adaptive	
Metric	Validation	Test	Validation	Test
$T10S$	0.681	0.324	0.387	0.263
$F\beta$	0.703	0.511	0.343	0.499
U'_f	0.671	0.548	0.362	0.463

Table 5. Performance of our systems on for $T10S$, F_β and U'_f metrics

Table 5 shows the performance of our batch and adaptive filtering systems on both the validation and test sets for all three metrics. For U'_f, we set $C_{corr} = 2$, $C_{fa} = 1$ and $U_f, min = -0.5$, which corresponds to $T10S$ with an α of -1.0. Note that F_β and U'_f have much more stable performance when going from validation to evaluation conditions, than $T10S$ for which performance decreases by more than half for the batch filtering tasks. In the adaptive filtering task, the performance drop experienced by the increase of N_+ in going from validation to test data hides an important observation: that the margin-based algorithm actually performs signifcantly *better* (significantly improved recall) on the test data than on the validation data! This again illustrates the effect of the variation of the isocurves of $T10S$ with number of relevant documents for a category; a system tuned to an optimal region on the validation data may find itself in a very suboptimal region when evaluated on the test data and the isocurves of $T10S$ shift with the change in category frequency, even though its relative performance on both validation and test data remains approximately that same.

Note also that F_β and U'_f have similar values for both batch and adaptive filtering and validation and test conditions. This is to be expected, since for most categories, we are operating in the region ($p > 1/3$) where F_β and U'_f have similar isocurves.

5. Conclusions

In our TREC-10 experiments and analysis, we observed the following:

- Standard Rocchio using relevance feedback to update the profiles but not the threshold performed surprisingly well: ranking fourth of thirty runs for both the F_β and $T10S$ metrics.

- Rocchio using relevance feedback and margin-based local regression (our new approach to adaptive thresholding) significantly outperformed the baseline Rocchio using relevance feedback and constant thresholds.

- The isocurves of the $T10S$ metric vary their locations in precision-recall space with the number of documents relevant to a particular category, causing this metric to favor common categories over rare ones and potentially obscuring important observations. We propose a slight but important modification to $T10S$ which removes these undesirable properties.

For future research, we would like to consider the following open questions:

- Why Rocchio produced more separable scores than kNN remains an open question. More failure analysis with methods other than kNN and Rocchio would be helpful in understanding the nature of adaptive filtering.

- Are all of the current classifiers used for adaptive filtering only finding those relevant documents which surround the initial two positive examples for each category? How can a classifier obtain relevance feedback for positive examples in clusters other than the initial one?

- How can we measure redundant information and return the set of documents which best covers what the user needs to know? What sorts of metrics are best suited for measuring this task?

- Why did feature selection fail to produce any improvement for our batch filtering results, when it has produced considerable improvement in other text categorization tasks on other corpora?

References

[1] Thomas Ault and Yiming Yang. knn at trec-9. In E. M. Voorhees and D.K. Harman, editors, *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*. Department of Commerce, National Institute of Standards and Technology, 2000.

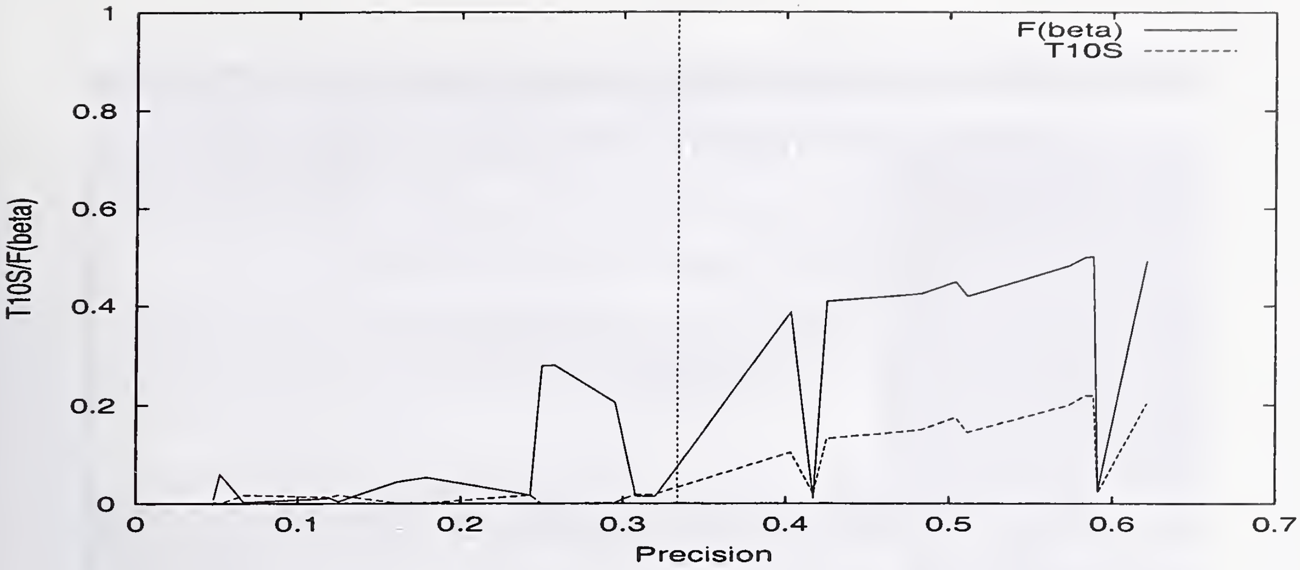


Figure 6. $T10S$ and F_β vs. precision for category R15

[2] Belur V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques.* McGraw-Hill Computer Science Series. IEEE Computer Society Press, Las Alamitos, California, 1991.

[3] J. J. Rocchio Jr. Relevance feedback in information retrieval. In G. Salton Ed., editor, *The SMART Retrieval System: Experiments in Automatic Document Retrieval*, pages 313–323, Englewood Cliffs, New Jersay, 1971. Prentice-Hall, Inc.

[4] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24:513–523, 1988.

[5] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of American Society for Information Sciences*, 41:288–297, 1990.

[6] Robert .E. Schapire, Yoram Singer, and Amit Singhal. Boosting and rocchio applied to text filtering. In *21th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, pages 215–223, 1998.

[7] C.J. van Rijsbergen. *Information Retrieval.* Butterworths, London, 1979.

[8] Y. Yang. Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In *17th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*, pages 13–22, 1994.

[9] Y. Yang. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1(1/2):67–88, 1999.

[10] Y. Yang and X. Liu. A re-examination of text categorization methods. In *The 22th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 42–49, 1999.

[11] Y. Yang and J.P. Pedersen. A comparative study on feature selection in text categorization. In Jr. D. H. Fisher, editor, *The Fourteenth International Conference on Machine Learning*, pages 412–420. Morgan Kaufmann, 1997.

[12] Yiming Yang. A study on thresholding strategies for text categorization. In *The Twenty-Fourth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01)*, New York, 2001. The Association for Computing Machinery.

[13] Yiming Yang, Thomas Ault, and Thomas Pierce. Improving text categorization methods for event tracking. In *The 23th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'00)*, page (submitted), 2000.

[14] Y. Zhang and J. Callan. Maximum likelihood estimation for filtering thresholds. In *Proceedings of the 24th Annual ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR2001)*, New York, 2001. The Association for Computing Machinery.

Video Retrieval with the Informedia Digital Video Library System

Alexander Hauptmann[1], Rong Jin[1], Norman Papernick[1], Dorbin Ng[1], Yanjun Qi[1], Ricky Houghton[2], and Sue Thornton[2]

[1]School of Computer Science,
Carnegie Mellon University
Pittsburgh, PA

[2]Sonic Foundry - MediaSite Systems
Pittsburgh, PA

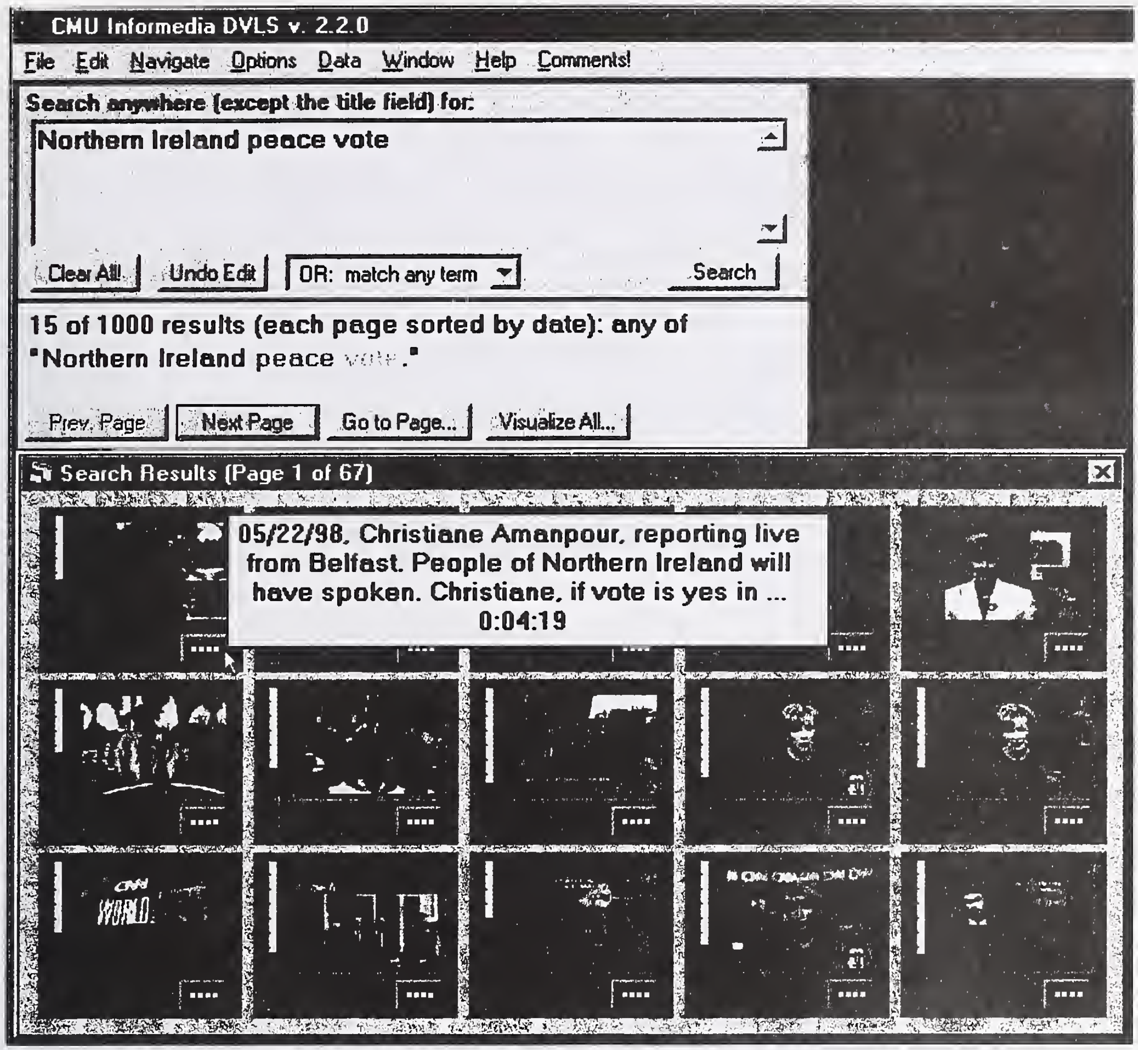
Background: The Informedia Digital Video Library System.

The Informedia Digital Video Library [1] was the only NSF DLI project focusing specifically on information extraction from video and audio content. Over a terabyte of online data was collected, with automatically generated metadata and indices for retrieving videos from this library. The architecture for the project was based on the premise that real-time constraints on library and associated metadata creation could be relaxed in order to realize increased automation and deeper parsing and indexing for identifying the library contents and breaking it into segments. Library creation was an offline activity, with library exploration by users occurring online and making use of the generated metadata and segmentation.

The goal of the Informedia interface was to enable quick access to relevant information in a digital video library, leveraging from derived metadata and the partitioning of the video into small segments. Figure 1 shows the IDVLS interface following a query. In this figure, a set of results is displayed at the bottom. The display includes a window containing a headline, and a pictorial menu of video segments each represented with a thumbnail image at approximately ¼ resolution of the video in the horizontal and vertical dimensions. The headline window automatically pops up whenever the mouse is positioned over a result item; the headline window for the first result is shown.

IDVLS also supports other ways of navigating and browsing the digital video library. These interface features were essential to deal with the ambiguity of the derived data generated by speech recognition, image processing, and natural language processing. Consider the filmstrip and video playback IDVLS window shown in Figure 2. For this actual video in the IDVLS library, the segmentation process failed, resulting in a thirty-minute segment. This long segment was one of the returned results for the query "Mir collision." The filmstrip in Figure 2 shows that the segment is more than just a story on the Russian space station, but rather begins with a commercial, then the weather, and then coverage of Hong Kong before addressing Mir. By overlaying the filmstrip and video playback windows with match location information, the user can quickly see that matches don't occur until later in the segment, after these other stories that were irrelevant to the query. The match bars are optionally color-coded to specific query words; in Figure 2 "Mir" matches are in red and "collision" matches in purple. When the user moved the mouse over the match bars in the filmstrip, a text window displayed the actual matching word from the transcript or Video OCR metadata for that particular match; "Mir" is shown in one such text window in Figure 2.

By investigating the distribution of match locations on the filmstrip, the user can determine the relevance of the returned result and the location of interest within the segment. The user can click on a match bar to jump directly to that point in the video segment. Hence, clicking the mouse as shown in Figure 2 would start playing the video at this mention of "Mir" with the overhead shot of people at desks. Similarly, IDVLS provided "seek to next match" and "seek to previous match" buttons in the video player allowing the user to quickly jump from one match to the next. In the example of Figure 2, these interface features allowed the user to bypass problems in segmentation and jump directly to the "Mir" story without having to first watch the opening video on other topics.



Search anywhere (except the title field) for:

Northern Ireland peace vote

15 of 1000 results (each page sorted by date): any of "Northern Ireland peace vote."

05/22/98. Christiane Amanpour, reporting live from Belfast. People of Northern Ireland will have spoken. Christiane, if vote is yes in ...
0:04:19

Figure 1. Text Query and Result Set in the Informedia System.

From the 11 hours of video, we extracted about 8000 shots, where a shotbreak was defined as an edited camera cut, fade or dissolve using standard color histogram measures. Instead of documents, the Video TREC track had defined shots as the unit of retrieval. We aggregated the MPEG I-frames for each shot to be alternative images for each shot. Whenever something matched to an image within a shot, the complete shot was returned as relevant. In total, there were about 80,000 images to be searched.

IDVLS Processing Components:

a. IMAGE PROCESSING

SHOT BREAKS: Color histogram analysis is applied to the MPEG-encoded video. This enables the software to identify editing effects such as cuts that mark shot changes. A single representative frame from each shot is chosen for use in poster frames or in the filmstrip view.

VIDEO OCR: The majority of traditional image processing techniques like optical character recognition (OCR) assume they work with a single image, but image processing for video works with image sequences where each image in the sequence often changes only slightly from the previous image. An overview of the Informedia Project's Video OCR (VOCR) process illustrates these points; Sato et. al discuss VOCR elsewhere in detail [14].

The goal of VOCR was to generate an accurate text representation for text superimposed on video frames. The VOCR process is as follows:

- Identify video frames containing probable text regions, in part through horizontal differential filters with binary thresholding.
- Filter the probable text region across the multiple video frames where that region is identified as containing approximately the same data. This time-based filter improves the quality of the image used as input for OCR processing.
- Use commercial OCR software to process the final filtered image of alphanumeric symbols into text. Optionally improve the text further through the use of dictionaries and thesauri.

The text detection phase can be used in key frame selection heuristics, just as face detection is used. The resulting text from VOCR processing has been used as additional metadata to document the contents of a video segment; this text can be searched just like transcript text.

OCR technology has been commercially available for many years. However, reading the text present in the video stream requires a number of processing steps in addition to the actual character recognition: First the text must be detected. The it must be extracted from the image, and finally converted into a binary black and white representation, since the commercially available OCR engines do not recognize colored text on a variably colored background. Since the extraction and binarization steps are quite noisy and do not produce perfect results, we decided to run the OCR engine on every 3rd frame where text was detected. Thus we obtained over 100 OCR results for a single occurrence of text on the screen that might last for just over 10 seconds. Frequently many of the results would be slightly different from each other, with a very high error rate. On this video collection, the word accuracy for detected text was estimated to be 27%.

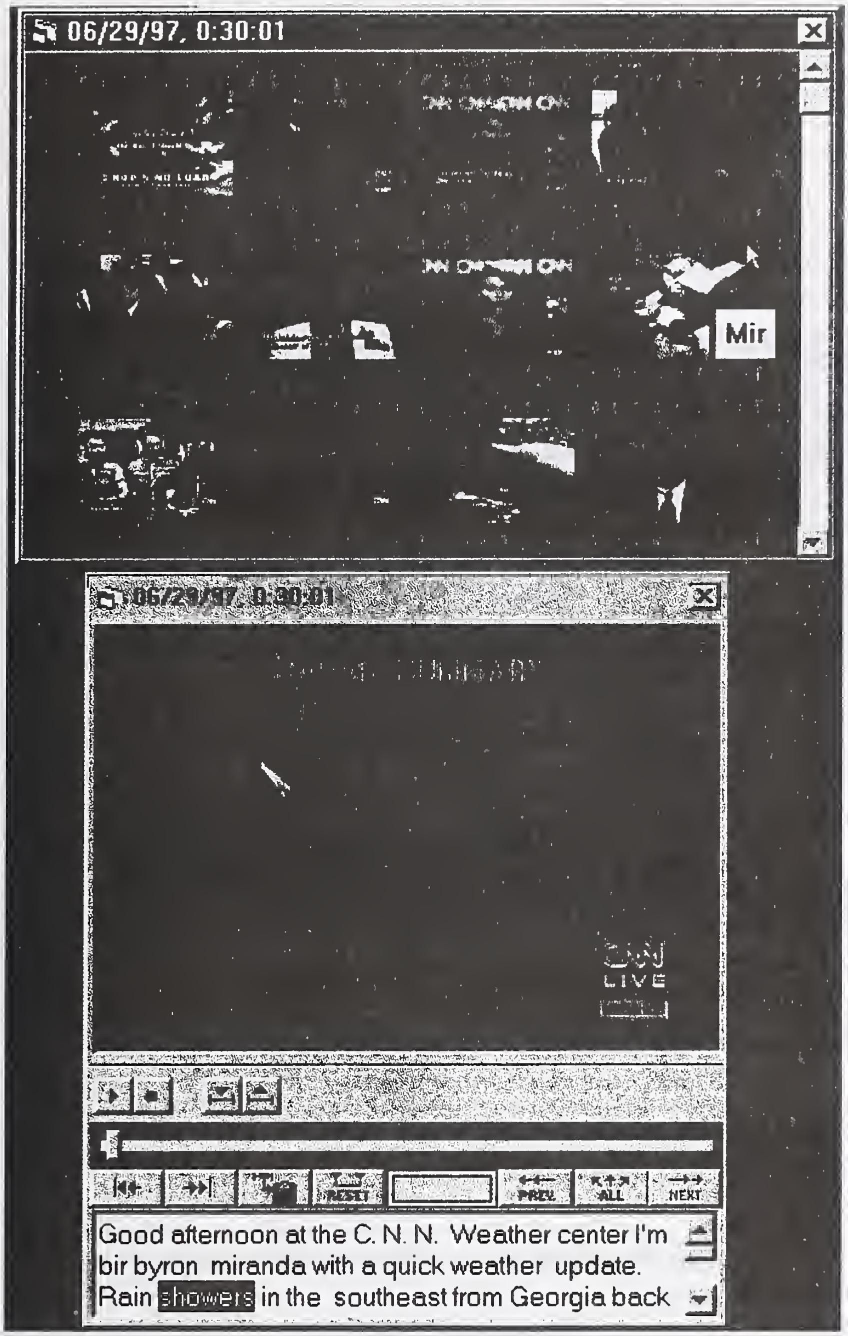
Face Detection and Matching: The Informedia system implements the detection of faces in images as described in [2] and face matching through 'eigenfaces'. While we experimented with face recognition using a commercial system [22] as well as an implementation of Eigenfaces [15], the accuracy of face recognition in this type of video collection was so poor, that it proved useless. Therefore, we only used a face detector that reported the presence of faces in each key frame.

Image Matching: Color histograms have been widely adopted by many image retrieval systems [5, 6,11] and, they served as the initial image query technique available to IDVLS users. While color histograms were applicable to the broad range of images accessible in the IDVLS library, their use in image indexing and retrieval revealed a number of problems. The histograms did not include any spatial information and hence were prone to false positives. Finally, they were unsuited for retrieving images in finer granularities, e.g., particular colors or regions. Referring to Figure 3, a user looking for a shot of grasslands could instead have retrieved these assorted images of predominantly blue and green colors.

b. Audio Processing

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 [18]. 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 just over 30%.



Good afternoon at the C. N. N. Weather center I'm bir byron miranda with a quick weather update. Rain showers in the southeast from Georgia back

Figure 2. Marking the filmstrip of a query with word location in the transcript and OCR.

Speaker Identification

Our speaker identification technology is based on standard Gaussian mixture models as defined by [9]. We use the segmented method with multiple training samples derived from chunks of audio that are 30 seconds in duration. We use weighted rank scoring as defined by Markov and Nakagawa [10].

c. Text Analysis

Titles: Segments are scanned for words that have a high inverse document frequency, and that are strongly distinguishing segments. All text is indexed and searchable. All retrieval of textual material was done using the OKAPI formula [13]. 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.

Figure 3. Result of a color histogram search

Approach to the TREC Video Track

Our approach to the Video Track in Trec was to use the Informedia system with only minor changes and see how well it would work. We treated general information queriès the same as known item queries. Specific modifications are discussed in the sections for the interactive and automatic system. For simplicity, we always assumed that the unit of retrieval was a single shot.

The Interactive Retrieval System

Since Informedia only uses static images for image matching, we decided make up for this shortcoming by utilizing multiple image search engines:

- Histo144v50 Image Search

Histo144v50 is based on a simple color histogram of the target image. First, the image is converted to the Munsell color space. We are using the Munsell Color space as described in [8]. The hue is isolated. Miyahara and Yoshida describe using Godlove's formula to represent the perceptual distance between some colors in the HVC space. We are using Euclidian distance to approximate Godlove's formula. The image is broken into 9 equally sized regions. A 16-bin histogram is taken of each region. The histograms are appended to each other to form a 144 dimensional vector. The vector is then reduced in dimensionality to 50 by multiplying with a previously computed singular value decomposition. Each vector is then placed in a tree data structure that allows K-nearest-neighbors searches.

- MCPv50 Image Search

MCPv50 computes the color and texture of the target image. The image is broken into 9 equally sized regions. A 15-bin histogram is taken for the Red, Green, and Blue. Then, six texture histograms of 15 bins each are taken. All of these vectors are append to make a 1215 dim vector. This vector is reduced to 50 dim by multiplying with a previously computed singular value decomposition. Each vector is than placed in a tree data structure that allows K-nearest-neighbors searches.

- Cuebik Image Search

Cuebik is based on one of the behaviors of the IBM QBIC image search engine. A palette of 255 colors is chosen for a database by marking the strongest colors found in a large sample of images. The target image is reduced to 256 equally sized regions. Each region is mapped to one of the palette colors, and recorded. A search is done by choosing a set of regions and finding all images that have the same color in the same region.

In addition to the above image search engines, we also used a downloadable version of the original IBM QBIC system as well as a search engine provided by James Wang from the University of Pennsylvania.

The search process foe each interactive query was as follows:
1. Determine key words in the text description of the query and use Video OCR text search to find them.
2. Use the supplied query images to initiate a search for relevant segments.
3. If a segment key frame or title looks related to the answer, open up its filmstrip and view details.
4. If the segment filmstrip looks related to the topic, but does not provide an answer, look one segment forward and back. If the topic in the adjacent segment is the same, scan the filmstrip of an additional segment forward or back.
5. If a frame answers the query, use that frame for relevance feedback with each of the image search engines to find more like it.
6. If a frame seems to be related, but does not answer the question, use that frame with each of the search engines to find more like it.
7. Repeat all steps as needed.

Automatic Retrieval

In the following we will elaborate only on the known item query set, because comprehensive relevance judgments were available for this set allowing automatic estimation of precision and recall for variations of our video retrieval system. The 34 known item queries are distinguished from the remaining 'general search' queries in that the information need tends to be more focused and all instances of query-relevant items in the corpus are known. This allows an experimental comparison of systems without the need for further human evaluations.

Since the evaluation could be done automatically, the top 100 search results were scored for all systems. The general unit of retrieval was a 'shot', in other words a time range between two shot changes, for

```
<videoTopic num="005" interactive="N-I" automatic="Y-A" knownItems="Y-K">
  <textDescription text="Scenes that show water skiing"/>
  <videoExample src="BOR17.MPG"  start="0h01m08s" stop="0h01m18s"/>
</videoTopic>
```

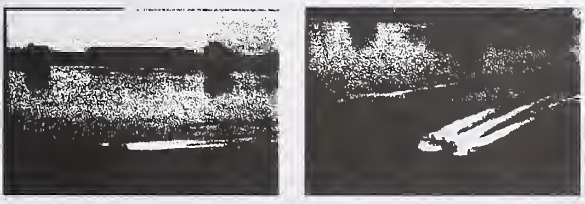


Figure 3. A sample known-item query in the automatic condition.

assuming that if two images are similar, their underlying generation models should also be similar, we can compute the similarity of image I_1 to image I_2 as $P(I_1 \mid M_2)$, i.e. the probability of generating image I_1 from the statistical model M_2. Preliminary experiments had shown that this model is more effective for image retrieval from the Video TREC collection than some of the traditional vector methods working on extracted features like e.g. QBIC [6,11].

Automatically Combining Metadata

When the various sources of data were combined for information retrieval, we used a linear interpolation with very high weights on the binary features such as face detection or speaker identification. This allowed these features to function as almost binary filters instead of being considered more or less equal to OCR, speech transcripts or image retrieval.

Experimental Results for the Automatic System

Evaluation Metrics

There are two aspects involved in any retrieval evaluation:

- **Recall.** A good retrieval system should retrieve as many relevant items as possible.
- **Precision.** A good retrieval system should only retrieve relevant items.

Many evaluation metrics have been used in information retrieval [21] to balance these two aspects. In the video retrieval track at TREC, a simple measure of precision at 100 items retrieved was used for scoring the systems. However, since there were only an average of 5.5 items relevant for each query, a perfect retrieval system that returned all relevant items at the top and filled the rest of the top 100 result slots with irrelevant items would only achieve a precision of 5.5 %.

Because our collection contains only small numbers of relevant items, we adopted the average reciprocal rank (ARR) [23] as our evaluation metric, similar the TREC Question Answering Track. ARR is defined as follows:

For a given query, there are a total of N_r items in the collection that are relevant to this query. Assume that the system only retrieves k relevant items and they are ranked as $r_1, r_2, \ldots, r_k$. Then, the average reciprocal rank is computed as

$$ARR = \left\{ \sum_{i=1}^{k} i / r_i \right\} / N_r \qquad (1)$$

As shown in Equation (1), there are two interesting aspects of the metric: first, it rewards the systems that put the relevant items near the top of the retrieval list and punish those that add relevant items near the bottom of the list. Secondly, the score is divided by the total number of relevant items for a given query. Since queries with more answer items are much easier than those with only a few answer items, this factor will balance the difficulty of queries and avoid the predominance of easy queries.

Table 1. Results of video retrieval for each type of extracted data and combinations.

Retrieval using:	Average Reciprocal Rank	Recall
Speech Recognition Transcripts only	1.84 %	13.2 %
Raw Video OCR only	5.21 %	6.10 %
Raw Video OCR + Speech Transcripts	6.36 %	19.30 %
Enhanced VOCR with dictionary post-processing	5.93 %%	7.52 %
Speech Transcripts + Enhanced Video OCR	7.07 %	20.74 %
Image Retrieval only using a probabilistic Model	14.99 %	24.45 %
Image Retrieval + Speech Transcripts	14.99 %	24.45 %
Image Retrieval + Face Detection	15.04 %	25.08 %
Image Retrieval + Raw VOCR	17.34 %	26.95 %
Image Retrieval + Enhanced VOCR	18.90 %	28.52 %
Image Retrieval + Face Detection + Enhanced VOCR	18.90 %	28.52 %
Image Retrieval + Speech Transcripts + Enhanced VOCR	18.90 %	28.52 %
Image Retrieval + Face Detection + Speech Transcripts +Enhanced VOCR	18.90 %	28.52 %

Results for Individual Types of Metadata

The results are shown in Table 1. The average reciprocal rank (ARR) and recall for retrieval using only the speech recognition transcripts was 1.84% with a recall of 13.2%. Since the queries were designed for video documents, it is perhaps not too surprising that information retrieval using only the OCR transcripts show much higher retrieval effectiveness to an ARR of 5.21% (6.10% recall). The effects of post-processing on the OCR data were beneficial, the dictionary-based OCR post-processing gave a more than 10% boost to 5.93 % ARR and 7.52 % recall. Again, perhaps not too surprisingly, the image retrieval component obtained the best individual result with an ARR of 14.99 % and recall of 24.45 %. Since the face detection could only provide a binary score in the results, we only evaluated its effect in combination with other metadata.

Results When Combining Metadata

Combining the OCR and the speech transcripts gave an increase in ARR and recall at 6.36 % and 19.30 % respectively. Again post-processing of the OCR improved performance to 7.07 % ARR and 20.74 % recall. Combining speech transcripts and image retrieval showed no gain over video retrieval with just images (14.88 % ARR, 24.45 % recall). However, when face detection was combined with image retrieval, a slight improvement was observed (15.04 % ARR, 25.08 % recall).

Combining OCR and image retrieval yielded the biggest jump in accuracy to an ARR of 17.34 % and recall of 26.95 % for raw VOCR and to an ARR of 18.90 % and recall of 28.52 % for enhanced VOCR. Further combinations of image retrieval and enhanced OCR with faces, and speech transcripts yielded no additional improvement. The probably cause for this lack of improvement is the redundancy to the other extracted metadata.

Discussion

What have learned from this first evaluation of video information retrieval? Perhaps it is not too surprising that the results indicate that image retrieval was the single biggest factor in video retrieval for this evaluation. Good image retrieval was the key to good performance in this evaluation, which is consistent with the intuition that video retrieval depends on finding good video images when given queries that include images or video.

One somewhat surprising finding was that the speech recognition transcripts played a relatively minimal role in video retrieval for the known-item queries in our task. This may be explained by the fact that discussions among the track organizers and participants prior to the evaluation emphasized the importance of a video retrieval task as opposed to ‘spoken document retrieval with pictures’.

There was a strong contribution of the OCR data to the final results. The results also underscore the fact that video contains information not available in the audio track. As a previous study noted, only about 50% of the words that appear as written text in the video are also spoken in the audio track [14], so the information contained in the text of the pictures is not redundant to the spoken words in the transcripts.

Overall, the queries presented a very challenging task for an automatic system. While the overall ARR and recall numbers seem small it should be noted that about one third of the queries were unanswerable by any of the automatic systems participating in the Video Retrieval Track. Thus for these queries nothing relevant was returned by any method or system.

We would like to caution that the known-item queries do not represent a complete sample of video queries. Video retrieval on general search queries, with less specific information needs, might result in a somewhat different conclusion about the combination of information sources. A preliminary analysis showed that ‘general search’ queries in the video track tended to be much more ‘speech oriented’, which is why the best performing system on that set of queries was entirely based on speech recognition transcripts.

Clearly, we can think of a number of improvements to the speech recognition component, using a parallel corpus for document and query expansion, and relevance feedback. However, the same techniques could be used to improve the OCR transcriptions as well.

References

[1] 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.

[2] Rowley, H., Baluja, S., and Kanade, T. "Human face detection in visual scenes", CMU, 1995. Technical Report CMU-CS-95-158.

[3] *Informedia Digital Video Library Project Web Site.* Carnegie Mellon University, Pittsburgh, PA, USA. URL http://www.informedia.cs.cmu.edu

[4] Hauptmann, A.G., Witbrock, M.J. and Christel, M.G. Artificial Intelligence Techniques in the Interface to a Digital Video Library, *Extended Abstracts of the ACM CHI'97 Conference on Human Factors in Computing Systems,* (New Orleans LA, March 1997), 2-3.

[5] Gong, Y. *Intelligent Image Databases: Toward Advanced Image Retrieval.* Kluwer Academic Publishers: Hingham, MA.

[6] Faloutsos, C., Barber, R., Flickner, M., Hafner, J., Niblack, W., Petkovic, D., Equitz, W. Efficient and Effective Querying by Image Content. *Journal of Intelligent Information Systems* 3(3/4), 231-262.

[7] Christel, M., Winkler, D., and Taylor, R. Multimedia Abstractions for a Digital Video Library. ACM Digital Libraries '97 (Philadelphia, PA, July 1997).

[8] M. Miyahara and Y. Yoshida, "Mathmatical transform of (R,G,B) color data to Munsell (H,V,C) color data", SPIE Vol 1001 "Visual Communications and Image Processing '88", pp 650-7. 1988.

[9] H. Gish and M. Schmidt, Text-Independent Speaker Identification, IEEE Signal Processing Magazine, October 1994, pages 18 – 32.

[10] K. Markov and S. Nakagawa, Frame Level Likelihood Normalization For Text-Independent Speaker Identification using Gaussian Mixture Models, ICSLP 96 1764-1767, 1996.

[11] 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.

[12] Kantor, P. and Voorhees E.M, Report on the Confusion Track, in Voorhees E.M, Harman, D.K., (eds.) "The Fifth Text Retrieval Conference, (TREC-5) 1997.

[13] Robertson S.E., et al.. Okapi at TREC-4. In The Fourth Text Retrieval Conference (TREC-4). 1993.

[14] 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.

[15] Satoh, S., and Kanade, T. NAME-IT: Association of Face and Name in Video. *IEEE CVPR97,* Puerto Rico, 1997.

[16] Schmidt, M., Golden, J., and Gish, H. "GMM sample statistic log-likelihoods for text-independent speaker recognition," *Eurospeech-9*, Rhodes, Greece, September 1997, pp.855 - 858.

[17] Schneiderman, H. and Kanade, T. Probabilistic Modeling of Local Appearance and Spatial Relationships of Object Recognition, *IEEE CVPR*, Santa Barbara, 1998

[18] 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.

[19] 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.

[20] Swain M.J. and Ballard, B.H. "Color Indexing," Int'l J. Computer Vision, vol. 7, no. 1, pp. 11-32, 1991.

[21] Tague-Sutcliffe, J.M., "The Pragmatics of Information Retrieval Experimentation, revised," Information Processing and Management, 28, 467-490, 1992.

[22] Visionics Corporate Web Site, FaceIt Developer Kit Software, http://www.visionics.com, 2002.

[23] Voorhees E.M, and Tice, D.M., "The TREC-8 Question Answering Track Report," The Eighth Text Retrieval Conference (TREC-8), 2000

Experiments Using the Lemur Toolkit

Paul Ogilvie and Jamie Callan
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{pto, callan}@cs.cmu.edu

Introduction

This paper describes experiments using the Lemur toolkit. We participated in the ad-hoc retrieval task of the Web Track. First, we describe Lemur in the System Description section and discuss parsing decisions. In the Experimental Results section, we discuss the official Lemur run and its retrieval parameters and we also discuss several unofficial runs. Finally, we summarize and draw conclusions.

System Description

The Lemur Toolkit [Lemur] is an information retrieval toolkit designed with language modeling in mind. Information about the toolkit is available at http://www.cs.cmu.edu/~lemur/. The toolkit is being developed as part of the Lemur Project, a collaboration between the Center for Intelligent Information Retrieval (CIIR) at the University of Massachusetts and the Language Technologies Institute (LTI) at Carnegie Mellon University. Lemur is written in C++ and C for use under UNIX and Windows NT.

Index

Lemur supports two types of indexes: one storing a bag-of-words representations for documents, the other storing term location information. In addition to storing standard information (document lengths, document frequency, collection term frequency, etc.) a user can build auxiliary files storing information useful for language modeling algorithms. For example, smoothing support files enable efficiency comparable to that of standard IR algorithms.

Retrieval

Lemur currently supports several retrieval algorithms. The primary retrieval model is a unigram language-modeling algorithm based on Kullback-Leibler divergence [Cover and Thomas 1991], also known as relative entropy. Also included is the OKAPI retrieval algorithm [Walker et al. 1998] and a dot-product function using TF-IDF weighting [Zhai 2001]. Our official submission was based on the Kullback-Leibler (KL) divergence algorithm.

The KL-divergence algorithm is derived as follows. Documents are ranked according to the negative of the divergence of the query's language model from the document's language model. Let θ_Q be the language model for the query Q and θ_D be the language model for document D. The documents are ranked by $-\mathrm{D}(\theta_Q \| \theta_D)$, where the function D denotes KL-divergence, as defined below.

$$\mathrm{D}(\theta_Q \| \theta_D) = \sum_w p(w|\theta_Q) \log \frac{p(w|\theta_Q)}{p(w|\theta_D)} \tag{1}$$

We assume that $p(w|\theta_D)$ has the following form:

$$p(w|\theta_D) = \begin{cases} p_s(w|\theta_D) & w \in D \\ \alpha_D p(w|\theta_C) & \mathit{otherwise} \end{cases} \tag{2}$$

where p_s is short for the probability given that the word was seen in the document, and α_d is a document dependent constant. α_d is used to reserve some portion of the probability distribution of the document's language model for words that are not present in the document. We also assume

$$p(w|\theta_Q, w \notin Q) = 0 \tag{3}$$

which is equivalent to stating

$$\sum_{w \in Q} p(w|\theta_Q) = 1 \tag{4}$$

Equation 4 states the sum of probabilities of all of the query terms is equal to 1. Equation 1 can be rewritten using properties of logarithms as

$$\sum_w p(w|\theta_Q)\log p(w|\theta_Q) - \sum_w p(w|\theta_Q)\log p(w|\theta_D) \tag{5}$$

Recall that we rank by $-\mathrm{D}(\theta_Q \| \theta_D)$. Since the first summation of Equation 5 is independent of the document, we can ignore this during ranking, yielding a ranking function of:

$$\sum_w p(w|\theta_Q)\log p(w|\theta_D) \tag{6}$$

Given equations 2 and 7, this can be rewritten as

$$\sum_{w \in D \cap Q} p(w|\theta_Q)\log p_s(w|\theta_D) + \sum_{w \in D \cap Q} p(w|\theta_Q)\log \alpha_D p(w|\theta_C) \tag{7}$$

We can add any number and subtract it and get the same results:

$$\begin{aligned} &\sum_{w \in D \cap Q} p(w|\theta_Q)\log p_s(w|\theta_D) + \sum_{w \in D \cap Q} p(w|\theta_Q)\log \alpha_D p(w|\theta_C) \\ &+ \sum_{w \in D \cap Q} p(w|\theta_Q)\log \alpha_D p(w|\theta_C) - \sum_{w \in D \cap Q} p(w|\theta_Q)\log \alpha_D p(w|\theta_C) \end{aligned} \tag{8}$$

Grouping the middle terms together and the first and last terms we get

$$\sum_{w \in D \cap Q} p(w|\theta_Q)\log \frac{p_s(w|\theta_D)}{\alpha_D p(w|\theta_C)} + \sum_{w \in Q} p(w|\theta_Q)\log \alpha_D p(w|\theta_C) \tag{9}$$

We can split the second summation by the terms in the logarithm. The cross-entropy of the query with the collection is a constant irrelevant to ranking. Using Equation 4, we can reduce the rest to:

$$\sum_{w \in D \cap Q} p(w|\theta_Q)\log \frac{p_s(w|\theta_D)}{\alpha_D p(w|\theta_C)} + \log \alpha_D \tag{10}$$

Equation 10 is the ranking equation implemented in Lemur.

As shown in [Lafferty and Zhai 2001], using the negative divergence of the query from the document is identical to the query likelihood model when using a maximum likelihood estimator for the query's language model. The query likelihood model was used in [Zhai and Lafferty 2001]. To see this, simply multiply equation 6 by the query length:

$$\sum_w c(w;Q)\log p(w|\theta_D) = \log \prod_w p(w|\theta_D)^{c(w;Q)} = \log P(Q|\theta_D) \propto P(Q|\theta_D) \tag{11}$$

where $c(w;Q)$ denotes the count of word w in the query Q.

Lemur can estimate $p_s(w|\theta_D)$ using maximum likelihood, a simple form of Jelinek-Mercer smoothing [Jelinek and Mercer 1985], Bayesian smoothing using Dirichlet priors [Berger 1985], and absolute discounting [Ney et al. 1994]. For TREC-10, we only considered Bayesian smoothing using a Dirichlet prior. The formula is given below.

$$p_s(w|\theta_D) = \frac{c(w;D) + \mu\, p(w|\theta_C)}{\mu + \sum_w c(w;D)} \tag{12}$$

where μ is a parameter and $c(w;D)$ denotes the count of word w in document D. The higher the value of μ, the more emphasis is placed on the collection language model. We estimate $p(w|\theta_C)$ using a maximum likelihood estimate:

$$p(w|\theta_C) = \frac{c(w;C)}{\sum_{w' \in C} c(w';C)} \tag{13}$$

The numerator is the collection term frequency of the word w, and denominator is simply the number of word occurrences in the collection.

Before we discuss parsing, we examine Lemur in terms of speed and space efficiency. We ran all tests on a computer with 2 GB RAM running Solaris. Table 1 displays the time spent on various indexing and retrieval tasks. The indexing speed of Lemur is about 2 GB per hour, including the time required to generate the smoothing support files. Table 2 shows the size of the built database. Currently, none of the data in Lemur is compressed by the index storing term locations, resulting in a database as large as the original data files. The bag-of-words index does compress data. Lemur stores a document posting list, which stores the words that occur in each document. This is not necessary for our retrieval algorithms after the smoothing support has been built, so it can be ignored or removed. If we do this, the size of database goes down from 10.3 GB to 5.2 GB.

Task	Time Required (hours:minutes:seconds)
Build Index	4:29:13
Generate Smoothing Support	0:30:55
Load Index	0:02:37
Run Queries (LM)	0:03:10
Run Queries (Okapi)	0:02:00

Table 1: Speed of Lemur

Component	MB	GB	Compression
Inverted index w/ locations	5110	5.0	none
Term / Doc Ids	130	0.1	none
Smoothing support	34	0.0	none
Total needed	5274	5.2	none
Doc posting list (sequential)	5232	5.1	none
Total	10506	10.3	none

Table 2: Database Size

Parsing

For document parsing, we used case-folding, the Porter stemmer [Porter 1980], and Inquery's stopword list (418 words) [Allan et al. 1980]. Possessive endings ("'s") were removed. We removed HTML tags and ignored text in script tags and HMTL comments.

We also used a very simple acronym recognizer that converts acronyms such as "F.H.A.", "FHA's", "FHAs", and "F.H.A.'s" into "FHA". We do not recognize acronyms containing numbers. If an uppercase word is found in text, it is checked against an acronym list. If the word is in the list, it is indexed uppercase. Otherwise it is converted to lowercase. To generate an acronym list, we assumed that text found in uppercase was generated by one of two language models. That is, an uppercase word is generated by either a language model describing general English or a model for special uppercase words. This is quantified by the following equation.

$$p(w|\,uppercase) = \lambda\; p(w\,|\,\theta_C) + (1-\lambda)p(w|\theta_{ucase}) \qquad (14)$$

We empirically set λ equal to 0.9. We estimate $p(w\,|\,\theta_C)$ as above and $p(w|uppercase)$ using a maximum likelihood estimate based on all uppercase words in the corpus. We then use the Expectation Maximization algorithm to recover $p(w|\theta_{ucase})$. From this we choose the acronym list by selecting all words occurring at least 10 times and where $(1-\lambda)p(w|\theta_{ucase}) > p(w|\theta_C)$. This gives a rather long list of 9,863 words for the WT10g corpus. We inspected this list manually. It looked reasonable, but it mostly consisted of 3 or 4 letter acronyms that would not be mistaken as words, such as "UMCP" and "AFROTC". However, it did contain some highly desirable words, such as "AIDS" and "US". An alternative to automatically generating an acronym list would be to compile a shorter list by hand that contained known problem acronyms, like "AIDS".

Query parsing was done almost identically to document parsing. Additionally, if a word occurred more times in the index in uppercase than in lowercase, the uppercase word was added to the query. This was done because sometimes users submit queries containing lowercase forms of the acronyms (e.g., "aids" instead of "AIDS"). We used no query expansion.

Experimental Results

We only submitted one official run, given the time constraints and youth of the toolkit. The official Lemur run used the toolkit's unigram language modeling retrieval algorithm, described above. We also present an unofficial Okapi run, in hopes that it will be interesting and demonstrate the validity of our Okapi implementation.

Official Run

For our official run, we use Dirichlet prior smoothing with a μ of 1600. We set this parameter empirically by tuning it to the WT10g data set using queries constructed from the web TREC9 topics. Figure 1 compares Lemur at relevant retrieved at 100 to the best and median systems. The Lemur run is marked with a cross. The median performance for each query is marked with a plus. The bar marks the best performance for the query.

The graph shows that for most queries the performance of the Lemur run was well above the median performance. For the few runs where Lemur's performance was below the median, the Lemur run was usually close to the median performance. Since "who and whom" were in our stopword list, we retrieved no documents for query 542.

Table 3 is a more quantified description of Lemur's performance. For precision at 100, precision at 1000, and average precision we counted the number of queries where Lemur performed the best (not counting cases where the median performance was also the best), better than the median (but not the best), equal to the median, and worse than median performance. From this table, it is easy to see that the performance of Lemur on the majority of the queries is above the median performance. Table 4 at the end of the paper shows recall and precision measures.

Unofficial Runs

In order to validate Lemur's implementation of Okapi, we present an unofficial Okapi run using Lemur. For this run, we do not use any pseudo-relevance-feedback. The parameters we use are those suggested by Walker et al. [1998]: avdl = 900, b = 0.75, k1 = 1.2, and k3 = 1000. Note that these parameters are probably not optimal for the web documents. Figure 2 and Table 3 illustrate that our OKAPI implementation performs reasonably well. Table 4 at the end of the paper shows recall and precision measures.

We also wanted to evaluate the effectiveness of the acronym list and recognizer. Only two queries had acronyms in their title, and both were lowercase in the original topic. Topic 526 was "bmi" and topic 538 was "fha". In both cases, the acronym was the only word. Our heuristic query parser added the uppercase acronyms to the original queries, resulting in "bmi BMI" and "fha FHA". For query 526, using acronyms had 14 relevant retrieved documents in the top 100, while the index without acronyms returned only 8 relevant documents in the top 100. Average precision dropped 42% when disabling acronyms. On the other hand, disabling acronyms worked better

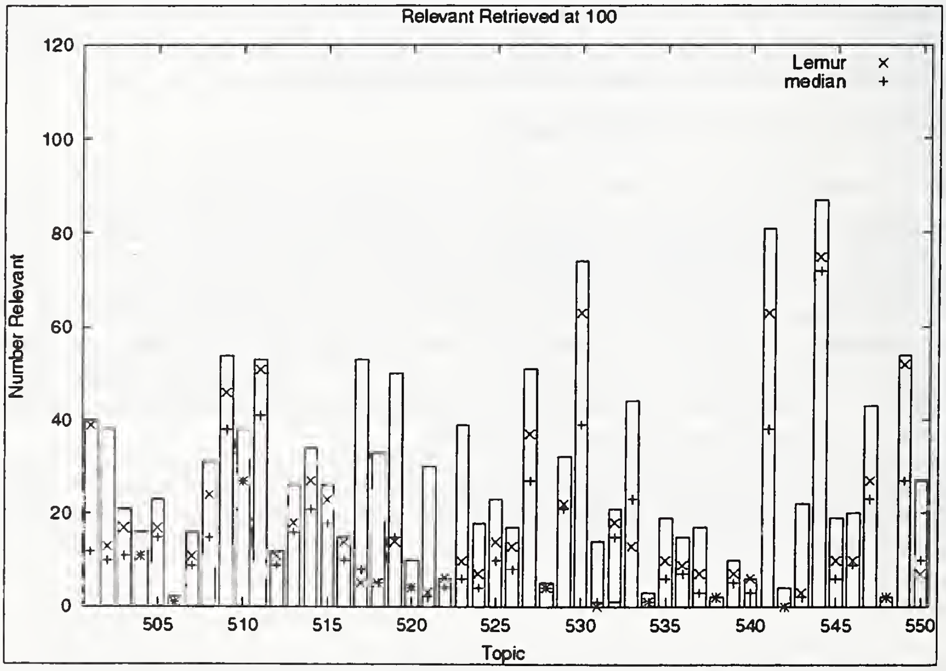


Figure 1: Relevant retrieved at 100 documents comparing Lemur with other systems

	Official (Lemur) Language Modeling			Unofficial Okapi		
	Precision at 100	Precision at 1000	Average Precision	Precision at 100	Precision at 1000	Average Precision
best && !median	2	2	2	0	0	0
> median && !best	33	32	43	33	26	38
= median	10	11	1	9	15	0
< median	5	5	4	8	9	12

Table 3: Performance of Lemur compared with other systems

for query 538. Both precision at 100 and 1000 were identical, but disabling the acronyms caused the average precision to rise 164%. All other queries performed almost identically whether acronyms were disabled or enabled.

Conclusions

Our primary goal was to demonstrate experiments using the new open-source Lemur toolkit to the Information Retrieval community on a commonly used research collection. Given Lemur's official run, simple language modeling techniques compare well with other TREC systems. This is particularly pleasing, given the lack of automatic query expansion or pseudo-relevance feedback.

Acknowledgements

We thank Thi Nhu Truong for assistance with the Lemur toolkit and getting it running on large data sets. We thank Chengxiang Zhai for help with Lemur and the paper. We thank John Lafferty for contributions to the toolkit and the paper. This work was sponsored in full by the Advanced Research and Development Activity in Information Technology (ARDA) under its Statistical Language Modeling for Information Retrieval Research Program. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors, and do not necessarily reflect those of the sponsor.

References

[Allan et al. 2000] J. Allan, M. Connell, W.B. Croft, F.F. Feng, D. Fisher, X. Li. INQUERY and TREC-9. In *TREC-9*, 2000, 504-513.

[Berger 1985] J. Berger. Statistical Decision Theory and Bayesian Analysis. New York: Springer-Verlag, 1985.

[Cover and Thomas 1991] T.M. Cover and J.A. Thomas. Elements of Information Theory. New York: Wiley, 1991.

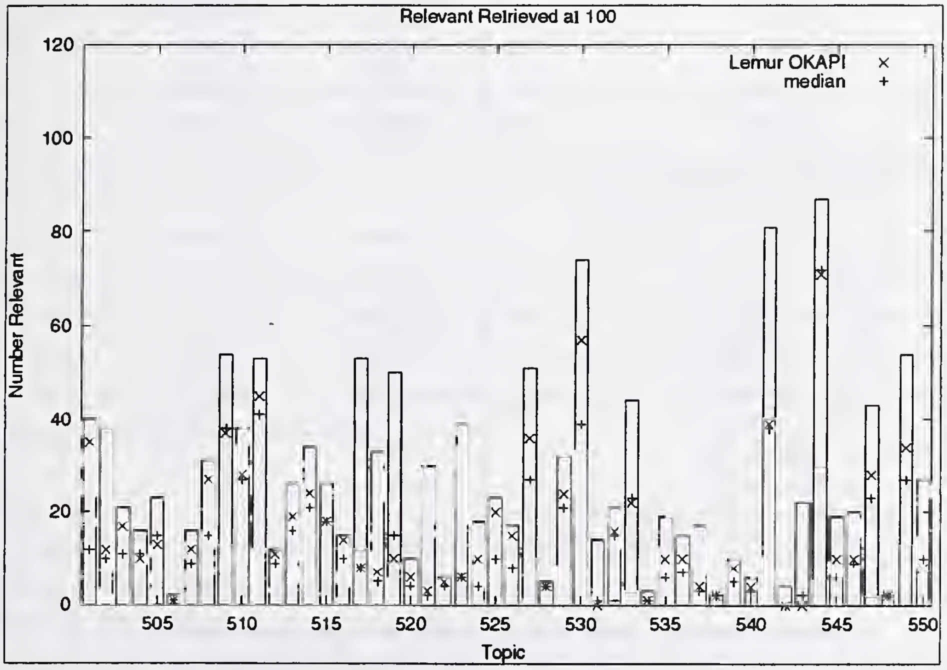


Figure 2: Relevant retrieved at 100 documents comparing Lemur's OKAPI with other systems

[Jelinek and Mercer 1985] F. Jelinek and R. Mercer. Probability Distribution Estimation from Sparse Data. *IBM Technical Disclosure Bulletin 28*, 1985, 2591-2594

[Lafferty and Zhai 2001] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 2001 ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01)*, 2001.

[Lemur] The Lemur Toolkit. http://www.cs.cmu.edu/~lemur

[Ney et al. 1994] H. Ney, U. Essen, R. Kneser. On structuring probabilistic dependencies in stochastic language modeling. In *Computer Speech and Language*, 8, 1994, 1-38.

[Porter 1980] M. Porter. An algorithm for suffix stripping. In *Program*, 14(3), July 1980, 130-137.

[Walker et al. 1998] S. Walker, S.E. Robertson, M. Boughamen, G.J.F. Jones, K. Sparck-Jones. Okapi at TREC-6: Automatic ad hoc, VLC, routing, filtering and QSDR. In *TREC-6*, 1998, 125-136.

[Zhai 2001] C. Zhai. Notes on the Lemur TFIDF model. http://www.cs.cmu.edu/~lemur/tfidf.ps

[Zhai and Lafferty 2001] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 2001 ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01)*, 2001.

	Official (Lemur)	**Unofficial**
	LM	Okapi
Total number of documents over all queries		
Retrieved:	48667	48667
Relevant:	3363	3363
Rel_ret:	2400	2334
Interpolated Recall - Precision Averages:		
at 0.00	0.6894	0.6146
at 0.10	0.4018	0.4073
at 0.20	0.3106	0.3306
at 0.30	0.2773	0.2785
at 0.40	0.2355	0.2315
at 0.50	0.2072	0.1929
at 0.60	0.1342	0.1212
at 0.70	0.0965	0.0878
at 0.80	0.0718	0.0589
at 0.90	0.0437	0.0245
at 1.00	0.0138	0.0140
Average precision (non-interpolated) for all rel docs(averaged over queries)	0.1985	0.1950
Precision:		
At 5 docs:	0.3720	0.3440
At 10 docs:	0.3200	0.3300
At 15 docs:	0.3093	0.3187
At 20 docs:	0.2920	0.2920
At 30 docs:	0.2580	0.2573
At 100 docs:	0.1758	0.1630
At 200 docs:	0.1305	0.1183
At 500 docs:	0.0770	0.0736
At 1000 docs:	0.0480	0.0467
R-Precision (precision after R (= num_rel for a query) docs retrieved):		
Exact:	0.2299	0.2304

Table 4: Summary Statistics for Official and Unofficial Runs

TREC-10 Experiments at CAS-ICT: Filtering, Web and QA

Bin Wang, Hongbo Xu, Zhifeng Yang, Yue Liu, Xueqi Cheng, Dongbo Bu, Shuo Bai
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
{wangbin,hbxu,zfyang,yliu,cxq,bdb,bai}@ict.ac.cn
http://www.ict.ac.cn/

Abstract

CAS-ICT took part in the TREC conference for the first time this year. We have participated in three tracks of TREC-10. For adaptive filtering track, we paid more attention to feature selection and profile adaptation. For web track, we tried to integrate different ranking methods to improve system performance. For QA track, we focused on question type identification, named entity tagging and answer matching. This paper describes our methods in detail.

Keywords: TREC-10, Filtering, Web track, QA

1. Introduction

CAS-ICT took part in the TREC conference for the first time this year. Among the total six tracks of TREC-10, we choose three of them: Filtering, Web and QA.

For filtering track, we undertook the adaptive filtering subtask. Our model is still based on vector representation and computation. A topic-term relevance function is defined to guide feature selection. For profile adaptation, we use a Rocchio-like algorithm. Four runs have been submitted for evaluation: three of them are optimized for T10U measure, another one for T10F measure. We use very simple optimization methods in our experiments and we do not use any other resource except the new Reuters Corpus.

For web track, we undertook the ad-hoc subtask. Our system is based on a general-purpose search engine developed by us alone. We try to improve system performance by integrating different ranking methods. Query expansion technology is used to modify the initial query. The *PageRank* algorithm is investigated in our experiments. Four runs have been submitted and two of them use hyperlink information.

For QA track, we undertook the main subtask. We first use SMART search engine to retrieve a set of documents from the TREC data sets. At the same time, a question analyzer is used to analyze the given 500 questions of TREC-10 and generates the question types and keyword lists. Then we use GATE to analyze the top 50 retrieved documents and extract the named entities from them. Finally, an answer extractor extracts the relevant answers from the named entities. Three QA runs have been submitted for evaluation.

2. Filtering

In the filtering task, we undertook the adaptive filtering subtask, which we think, is more interesting and realistic than the other two subtasks.

2.1 Problem Description

This year the filtering task has 84 topics, which are exactly the categories in Reuters Corpus.

The total documents for this task (new Reuters Corpus) are divided into two parts: 23,307 documents for training (training set) and the remaining about 783,484 documents for testing (testing set). All the documents are Reuters everyday news, dating from August 1996 to August 1997. Two measures are given for adaptive filtering: T10U and T10F, the former is a linear utility measure and the latter is a kind of F-measure. In the adaptive task, only two positive samples in training set are given for each topic, the goal is to retrieve relevant documents one by one from the coming testing documents stream and get maximum T10U or T10F value at the same time.

2.2 System Description

Our adaptive filtering system consists of two components: the initializing component and the adaptation component. The former is used to get the initial data through training and the latter is to adapt these data when retrieving testing documents.

2.3 Training

In training procedure, we first process the training set for basic term statistics, this includes term tokenization, stemming and frequency counting. Then we can select terms from the positive and some pseudo-negative samples. After topic processing, we can get the initial profile vector by summing up the topic and feature vectors with different weight. Finally, we can compute the similarity between the initial profile vector and the positive documents to set the initial threshold.

2.3.1 Training set processing

In this step, all the training documents are processed. First we tokenize each document into single words, then eliminate the stop words and some other words with low frequency in the training set, and then we stem each word using the Porter Stemmer(*http://www.cs.jhu.edu /~weiss/*). Finally, we count each word's frequency(*TF*) within each document and the word's document frequency(DF) across the training set. When processing the training texts, we only use the <*title*> and <*text*> fields. Thus each document can be represented with its term frequency vector. Meanwhile, we can get the *IDF* statistics of the training documents. Since we can't use the *IDF* statistics of testing set, we use in the following steps the *IDF* statistics of the training document in term weighting. Ideally, we can update the IDF statistics when retrieving documents from the testing documents stream. But [16] has indicated that doing so does not seem to improve the overall filtering performance. So we use the same *IDF* statistics of the training documents all over our experiments.

2.3.2 Topic set processing

This year, each topic consists of three short parts: the <*num*> field, the <*Reuters-code*> field and the <*title*> filed. The <*title*> field includes only one or two words. We can't get more information from such kind of topics than from the topics of TREC-8 or TREC-9, which are described with more words. Of the three parts, we regard the <*title*> field as the most important. Though the <*Reuters-code*> fields may provide some information about relationships between different topics, we do not use them at all. Processing the topics is very simple, we only extract and stem the <*title*> field words to construct the vector for each topic.

2.3.3 Term selection

To reduce the computation complexity, we apply a method for feature selection. Here the features are all terms, each term is a word.

For each topic, we have two positive samples. So for all 84 topics, we have 168 positive samples. Thus, of the 168 samples, each topic has two positive samples, and we can suppose the other 166 samples are *negative* to the topic. Because one document may be relevant to more topics, our supposition is not very correct. But we try it for lack of information.

We define a word-topic correlation function as:

$$Cor(w_i,T_j)=\log({P(w_i\,|\,T_j)}\big/{P(w_i\,|\,\neg T_j)}) \tag{2.1}$$

Where $P(w_i|T_j)$ means the probability that word w_i exists in the relevant documents of topic T_j. On the contrary, $P(w_i\,|\,\neg T_j)$ means the probability that word w_j exists in non-relevant documents of topic T_j. For each topic, we compute the *Cor* value of each word in the positive two samples and choose the words with high *Cor* values as the features. Here we use maximum likelihood estimation. We compute the frequency that a word exists in the two positive documents as the estimation of $P(w_i|T_j)$, and the frequency that a word exists in the other "*negative*" documents as the estimation of $P(w_i\,|\,\neg T_j)$. If the estimation of $P(w_i\,|\,\neg T_j)$ is equal to zero, we give the *Cor* a big value.

After getting the feature words for each topic, we combine them to construct one feature space; the topic vector and the feature vector must be mapped into this space.

2.3.4 Profile initialization

For each topic, the profile vector(denoted as $\bar{P}$) is the weighted sum of the topic vector(denoted as $\bar{T}$) and the feature vector(denoted as $\bar{F}$), which is the sum of the two positive documents vectors. The formula is:

$$\bar{P}=\alpha*\bar{F}+\beta*\bar{T} \tag{2.2}$$

In our experiments, we set α=1,β=2 to give prominence to the topic words. So far, each component of the vectors is represented with *TF* values. Then we change it by multiplying with its *IDF* coefficient.

2.3.5 Similarity computation

To compute the similarity between a topic profile($\bar{P}_i$) and a document($\bar{D}_j$), we use the vector *cosine* similarity formula:

$$sim(\bar{P}_i,\bar{D}_j)=\cos(\bar{P}_i,\bar{D}_j)=\frac{\bar{P}_i\bullet\bar{D}_j}{|\bar{P}_i|\times|\bar{D}_j|} \tag{2.3}$$

Each component of the vectors is represented with *TFIDF* value. Here we use $TF_i*\log(1+\frac{N}{DF_i})$ formula. We also try other formulae in our experiment, but the results are almost the same.

2.3.6 Initial threshold setting

We do not have good idea to set the initial threshold. According to our understanding, we believe we can't use the training documents to train the initial threshold because they are prepared for batch filtering task, not for adaptive task, we cannot use the relevance information of the other documents in training set except the two positive ones. Thus we have to use a very simple method, for each topic, we choose a small fixed value as the initial threshold which is smaller than the similarity between the initial profile vector and the two positive samples.

2.4 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 and meanwhile the system can tell you the document is really relevant or not. With this information, some kind of adaptation may need to take to improve system performance. The adaptation may include threshold updating or profile updating.

2.4.1 Threshold adaptation

In TREC-10, two measures are defined to measure the performance of an adaptive filtering system. One is T10U, which is a linear utility; another is T10F, which is a kind of F-value. The goal of the adaptive filtering system is to get maximum T10U or T10F.

For T10U, we have two goals: one is to avoid negative T10U as far as we can, another is to improve the precision while the recall can't be greatly reduced. We only apply method for the former goal in our experiments.

For T10F, we also have two goals: one is to avoid retrieving zero documents, another is also to improve the precision while the recall can't be greatly reduced. We only apply method for the former goal, too.

2.4.2 Profile adaptation

After retrieving more and more relevant or non-relevant documents, we can get more useful information and understand the user's interest better. Thus we can adapt each profile vector, which represents each user's interest. Our profile adaptation includes positive adaptation and negative adaptation. For positive adaptation, we add the document vector of the positive documents to the old profile vector. For negative adaptation, we subtract the document vector of the negative documents from the old profile vector. When retrieving the *n+1* th document D_{n+1}, we can adapt the *n*th profile to the *n+1* th profile according the following formula:

$$\bar{P}_{n+1} = \begin{cases} \bar{P}_n + \alpha * \bar{D}_{n+1} & \text{if } D_{n+1} \text{ is relevant} \\ \bar{P}_n - \beta * \bar{D}_{n+1} & \text{otherwise} \end{cases} \tag{2.4}$$

Thus after retrieving *n+1* documents, all the retrieving relevant documents make the positive set denoted as $\{D^+\}$, the other documents set is $\{D^-\}$. Then the new profile vector become

$$\bar{P}_{n+1} = \bar{P}_0 + \alpha * \sum_{D_i \in \{D^+\}} \bar{D}_i - \beta * \sum_{D_j \in \{D^-\}} \bar{D}_j$$
$$\text{here } \{D^+\} \cup \{D^-\} = \{D_1, D_2, ..., D_{n+1}\}, \{D^+\} \cap \{D^-\} = \phi \tag{2.5}$$

Formula (2.5) is some kind of the Rocchio[19] algorithm except one point: we do not compute the centroid vector of the positive set or negative set and regard it as one vector. In other words, we pay more attention to the retrieving documents than the initial profile vector. Furthermore, we investigate the values of α and β. We found without negative feedback, the result is worse. In our experiments, we set α=1, β=1.1 or 1.3.

2.5 Evaluation Results and Analysis

We have submitted four adaptive filtering runs: one for T10F optimization, three for T10U optimization. Because we do not use complex optimization method, the results of the 4 runs are similar to each other. The evaluation results are shown in Table 2.1.

Table 2.1 shows that in each run, the results of about 2/3 of all topics are better than the medians, and most of the remaining results are worse. Unfortunately, about 1/3 of the worse results are worst results and most of the worst results are zero. Thus the overall performance of our runs is not high.

Run ID	MeanT10SU	T10SU vs. median(topic nums)			MeanT10F	T10F vs. median(topic nums)		
		>(Best)	=	<(Worst/Zero)		>(Best)	=	<(Worst/Zero)
ICTAdaFT10Ua	0.204	55(8)	1	28(11/11)	0.368	59(3)	2	23(8/7)
ICTAdaFT10Ub	0.205	51(0)	3	30(12/12)	0.366	55(0)	4	25(7/6)
ICTAdaFT10Uc	0.207	52(3)	2	30(11/11)	0.354	51(1)	4	29(12/9)
ICTAdaFT10Fa	0.206	55(4)	1	28(14/14)	0.387	62(1)	1	21(6/5)

Table 2.1 ICT adaptive filtering runs in TREC-10

We focused on the worst zero results and the best results. We found, on one hand, most of the zero results belong to "small" topics, which have small amount of relevant documents in the whole testing set and we called them "hard" topics. On the other hand, most of the best results belong to "big" topics. That is to say, our method is somehow fitful for "big" topics but not very fitful for "small" topics. This may result from two reasons: First, our feature selection method cannot find the best features of these "hard" topics from only two positive samples. Second, our optimization method is too simple to satisfy different cases.

In the future work, we will pay more attention to three aspects. For feature selection, we will try more effective methods. For optimization, we will try more complex methods. For profile adaptation, we may add feature reselection module.

3. Web Track

3.1 System Description

This year, the goal of the main web track is to retrieve the most relevant documents for each topic in the topic set (501~550) from the WT10G collection. Our system is based on a general-purpose search engine which we have been developing and improving since 1998. The system consists of four basic components: indexer, query generator, search agent, and search server. The indexer scans all documents of the WT10G and generates full text indexes and term statistics. The query generator analyzes the TREC topics and generates real queries. The search agent submits real queries to search server and shows the return results in visible format. The search server receives queries, searches documents and returns results to the search agent

3.2 Searching Process

3.2.1 Query construction

Query generator reads TREC topics, extracts valuable terms and submits them to the search system. Therefore, the initial query for each topic is a set of some useful terms of the topic.

At this stage all stop words are removed. In addition to the basic stop words, we have also removed some other stop words that carry little information in the topics, such as *find*. While processing each topic, we only use the <*title*> field.

3.2.2 Initial retrieval

For each initial query, the search system retrieves some documents and ranks them using formula (3.1)[7]. Some top ranked documents are returned as the initial search results.

$$S_{q,d} = \frac{\sum_{t \in q \cap d} (1 + \log f_{t,d}) * \log(1 + \frac{N}{f_t})}{\sqrt{\sum (1 + \log f_{t,d})^2} * \sqrt{\sum \log^2 (1 + \frac{N}{f_t})}} \qquad (3.1)$$

Where $S_{q,d}$ is the similarity of document d and query q, f_t is the number of documents in which term t occurs in WT10G, $f_{t,d}$ is the frequency term t occurs in document d (within-document frequency), and N is the total number of documents in WT10G.

3.2.3 Query expansion

After first retrieval we can get many ranked documents. Then we can regard some top ranked documents as *relevant*. All the terms in these documents are weighted according to formula (3.2)[1][8].

$$TSV = r * W_t = r * (\frac{k_5}{k_5 + \sqrt{R}} \log(k_4^{'} \frac{N}{N-n} + \frac{n}{N-n}) + \frac{\sqrt{R}}{k_5 + \sqrt{R}} \log \frac{r + 0.5}{R - r + 0.5}$$
$$- \frac{k_6}{k_6 + \sqrt{S}} \log \frac{n}{N-n} - \frac{\sqrt{S}}{k_6 + \sqrt{S}} \log \frac{s + 0.5}{S - s + 0.5}) \qquad (3.2)$$

Where *TSV* means *Term Selection Value* that is used to rank terms. N is the number of documents in WT10G, n is the number of documents in which term t occurs, R is the number of relevant documents, r is the number of relevant documents which contain term t, S is the number of non-relevant documents in WT10G, s is the number of non-relevant documents which contain term t in WT10G. $k_4^{'}$,k_5, k_6 are parameters.

Some top terms with their *TSVs* are selected for the new query vector construction.

3.2.4 Second retrieval

In this stage, the new query is submitted to the search system and new results are retrieved which are ranked by formula (3.3).

$$w_{q,d} = \sum_{t \in q} w_t \frac{(k_1 + 1) * f_{d,t} * b_{d,t}}{k_1 * [(1 - b) + b * \frac{L_d}{avr\_L_d}] + f_{d,t}} * \frac{(k_3 + 1) * f_{q,t}}{k_3 + f_{q,t}} \qquad (3.3)$$

Where L_d is the length of document d, avr_L_d is the average document length, $b_{d,t}$ is within-document importance of term t in document d, which includes multiple factors such as the frequency t occurs in document title, bolded text, and hyperlink text. k_1,k_2, k_3 are parameters.

3.3 Applying Link Analysis Technology in Web track

We also investigate link analysis technology that is used to rank web pages using link information. In our experiments, we mainly use an improved *PageRank* algorithm.

3.3.1 Basic *PageRank* algorithm

Brin & Page[5] suggested a link based search model called *PageRank* that first evaluated the importance of each web page based on its citation pattern. The *PageRank* algorithm re-ranks the retrieved pages of a traditional search scheme according to their *PageRank* values.

In this approach, a web page will have a higher score if more web pages link to it and this value will increase if those web pages' scores increase. The *PageRank* value of a given web page t, denoted as Pr(t),can be iteratively computed according to formula (3.4)[5].

$$\Pr(t) = (1-d) + d * \sum_{i=1}^{m} \frac{\Pr(t_i)}{c(t_i)} \tag{3.4}$$

Where t_1, t_2, ..., t_m are the web pages which link to page t, d is a parameter (set to 0.85 as suggested by [5]) and $c(t_i)$ is the number of outgoing links for page t_i. For simplification, all the pages which link to page t are called the *pre-set* of t, denoted as *pre-set(t)*,and all the pages which page t links to are called the *post-set* of t, denoted as *post-set(t)*.

3.3.2 Implementation of the *PageRank* algorithm

From formula (3.4) we can see that computing *PageRank* is very simple in itself. But because the numbers of web pages is usually very large (in WT10G the number is 1,692,097) and the number of hyperlinks is even larger, the iteration process may be time-consuming. In order to solve this problem, we do some useful pretreatment.

First, we try to get rid of all the noisy hyperlinks that have little information before iteration. Meanwhile, we mark the web pages that have empty *pre-set* and do not participate in the iteration.

Second, we pretreat the *post-sets* of all the web pages. From formula (3.4) we can see that, for web page t and one page in its *pre-set*, t_j, if $c(t_j)$ is sufficiently large, the value of $\Pr(t_j)/c(t_j)$ will be very small, intuitively speaking, that means the influence of webpage t_j to webpage t is very small in the web graph. Thus for web pages like t_j, we don't let them only simply take part in the iteration. A threshold is set in advance, if one page's *post-set* has bigger size than the threshold, we assign a fixed value as the *PageRank* for this page. In this way we can greatly reduce the iteration cycle in our experiments.

3.3.3 Convergence properties

According to the probabilistic meanings of formula (3.4), after each time of iteration, the sum of all pages' *PageRank* should be equal to 1 after standardization. To avoid that the *PageRank* values are too small and incomparable, we introduce a new standardization method.

In addition, we use formula (3.5) to determine whether we should stop or not after the $i+1$ th iteration.

$$\underset{t}{Max}(\Pr(t)^{(i)} - \Pr(t)^{(i+1)}) - \underset{t}{Min}(\Pr(t)^{(i)} - \Pr(t)^{(i+1)}) < \delta \tag{3.5}$$

In order to determine whether the iteration should be stopped or not, we consider not only the decreasing tendency of all the web pages, but also the low changeability between iterations. In our experiments, after 25 times' iteration we get a relatively steady convergence results.

3.3.4 Integrating ranking methods

We integrate the above ranking methods into formula (3.6) to get the integrated rank of one page:

$$W_d = W_{dc} + W_{dl} = W_{dc} + \frac{W_{dc}}{\log \frac{PR_{\max} * k}{PR_d}} \tag{3.6}$$

Where W_d is the final weight of a page (document), W_{dc} is the content-based document weight computed by formula (3.3), W_{dl} is the link-based document weight. PR_d is the *PageRank* value of document d computed by formula (3.4), PR_{max} is the maximum *PageRank* value among all the documents, k is a constant greater than 1.

3.4 Evaluation Results

We have submitted four runs to NIST. The results are listed in Table 3.1.

Technology(Run id)	Average Precision (non-interpolated) over all relevant docs	P@20 docs	R-Precision
Baseline(ICTWeb10n)	0.0860	0.1450	0.1244
Query Expansion(ICTWeb10f)	0.0464	0.0620	0.0657
Link Analysis(ICTWeb10nl)	0.0860	0.1410	0.1147
Query Expansion &Link Analysis(ICTWeb10nfl)	0.0464	0.0620	0.0657

Table 3.1 Web track results

Table 3.1 shows that our query expansion method leads to performance degradation. The reason may be that we first use such a complex query expansion method in our system and some of the parameters need to be revised in future tests.

From the above table, we have also found that the results integrated with link analysis have little difference with the benchmark. We believe the reason is not algorithm itself but the small size of the web pages set. In our experiments, we use WT10G which is a close set that doesn't link to outside, thus many informative hyperlink are not included. If the experimental data size is large enough, the results should be very good. The *PageRank* method offers an approach that evaluates the web pages objectively.

Our future work includes three aspects: First, we will try different probabilistic retrieval models; second, we will try alternative feedback methods; third, we will try new connectivity computation methods.

4. Questing Answering track

4.1 System Description

Our TREC-10 question answering system consists of four basic components: IR search engine, question analyzer, named entity tagger and answer extractor. Figure 4.1 illustrates the whole architecture.

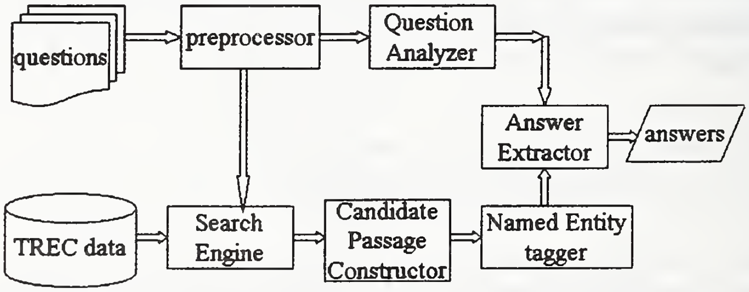


Figure 4.1: Architecture of the ICT TREC-10 QA System

We first use an IR search engine to retrieve a set of documents from the TREC data sets. At the same time, a question analyzer analyzes the given 500 questions of TREC-10 and generates the question type and keyword-list for each question. Then we use a named entity tagger to analyze the top 50 documents retrieved by the search engine and extract the named entities from them. Finally, an answer extractor determines the relevant answers from the named entities using the question type and keyword-list.

4.2 SMART Search Engine

As we know, SMART(*ftp://ftp.cs.cornell.edu/pub/smart*) is an implementation of the vector-space model of information retrieval proposed by Salton dating back in the 60's. The primary purpose of SMART is to provide a framework in which one conducts information retrieval research. It is (as we heard) distributed for research purpose only. Since TREC-QA allows us to use search engine freely, we choose SMART as our IR search engine because it is easy to use. To meet the need of our QA system, we add some new components into SMART. We also use the feedback function of SMART to generate a set of retrieved documents, based on which we make a run ICTQA10c. But this run becomes the worst of all our three runs, which proves we failed in applying the feedback. This indicates that we need to do further research on feedback, such as using the LCA (Local Context Analysis) feedback technique[14].

4.3 Question Analyzer

Main answer and question types what we can extract are listed in table 4.1.

Answer type	Question type	Example
PERSON	Who/Whom/What-person/Which-person	Mr. Mulroney
LOCATION	Where/What-location/Which-location	Orange County
ORGANIZATION	What-org/Which-org/Who	Penn Central Corp.
MONEY	How much	$28.5 million
PERCENT	How much/What percentage	4%
DATE	When/What date	Aug. 6, 1945
NUMBER	How many	Five
DURATION	How long	20 years
DISTANCE	How long/how far/How tall	10 miles
AREA	How large/How big	100 square kms
MEASUREMENT	How heavy/How fast	4 tons
CURRENCY	What currency	Dollar
NATIONALITY	What nation/What language	Icelandic
REASON	Why/How	-
NAME	What/Which is,are NP	small brain defect typical victims
No Answer	All	NIL

Table 4.1 Answer and question types

We determine the question type based on two rule sets. First rule set is keyword-based, which consists of some patterns as follows:

Who : PERSON
Where : LOCATION

What day : DATE
How many : NUMBER

Such patterns determine the question type only according to the interrogative of the question. This kind of rules can be applied to the PERSON, LOCATION, DATE, NUMBER types and so on. The second rule set is template-based. Some examples of this kind of rules are as follows:

What do,does,did @2 cost? : MONEY
What person @3? : PERSON
Which person @3? : PERSON
What is,was @1`s birthday? : DATE
What is,are,was,were @1's employer? : ORGANIZATION

Such rules are useful to determine those questions led by "What", whose question types are hard to determine only by keyword-based rules.

4.4 Named Entity Tagger

We use GATE(*http://gate.ac.uk*) as our Named Entity tagger. GATE is developed by Sheffield NLP group. It can extract the named entities of PERSON, LOCATION, MONEY, PERCENT, DATE, ORGANIZATION and so on. But it has some obvious shortcomings, for example, it can't extract the pure NUMBER and most MEASUREMENT entities. Furthermore, many entities are incorrectly identified in GATE. Therefore, we revise the basic GATE system. First, we add several new components to identify the NUMBER and MEASUREMENT entities. Second, we modify GATE to improve the tagging correctness, mainly for the MONEY and PERCENT entities. As to some abstract question types, such as REASON(why), MEANING(what), METHOD(how) and so on, we apply some rules based on certain features to tag a snippet from the candidate passage as the candidate answer.

Before identifying the named entities, we need to construct candidate passages using the top 50 documents retrieved by SMART search engine. The algorithm[10][11][15] is as follows:

Step 1:Parse the sentences of the documents.

Step 2:Retrieve the sentences that contain keywords in the question.

Step 3:Construct a candidate passage every two sentences. If one sentence is long enough, it becomes a candidate passage itself.

Step 4:Assign each candidate passage a initial score equal to the score ranked by SMART search engine, i.e. $score(P)=IR(D)$, D is the document where the candidate passage P lies.

Step 5:Add the *idf* value of all matched keywords contained by a candidate passage to its score.

Step 6:Calculate the number of matched keywords($count\_m$) in each candidate passage. P. Add 0 to $score(P)$ if the number of matched keywords is less than the *threshold*. Otherwise, add $count\_m$ to $score(P)$. The *threshold* is defined as follows:

$threshold=count\_q$ if $count\_q<4$;
$threshold=count\_q/2.0+1.0$ if $4<=count\_q<=8$;
$threshold=count\_q/3.0+2.0$ if $count\_q>8$;
here $count\_q$ is the number of keywords in the question.

Step 7:Calculate the size of matching window, then add $40*count\_m/size(matching\_window)$ to $score(P)$. The size of matching window is defined as the number of keywords in the candidate passage between the first matched keyword and the last one.

Step 8:Re-rank the candidate passages by their final scores and output the top *10* or *20* passages to the Named Entity tagger.

4.5 Answer Extractor

The answer extractor compares the question type with each named entity in candidate passages. If a candidate passage contains a named entity matching the question type, we add *100* to the score of the passage. When there is more than one matched named entity, we only count once.

After the process above, we re-rank the candidate passages according to the named entity and question types, then output the top 5 named entities as the final answers. If the question type is unknown, we intercept a snippet with the largest density of keywords in the candidate passage to answer the question.

4.6 Results and Analysis

There are three subtasks in TREC-10 QA: main, list and context. We only participate in the main task and submit three runs in the 50-byte category. ICTQA10a uses the top 20 candidate passages for each question. Both ICTQA10b and ICTQA10c use only top 10 candidate passages for each question, the difference is that ICTQA10c uses the feedback of SMART in the IR phase. The evaluation results are presented in table 4.2 and 4.3. Table 4.2 shows the results in strict evaluation while table 4.3 does in lenient evaluation. (MRR means "Mean Reciprocal Rank".)

ICTQA10b is better than ICTQA10a, which shows that more candidate passages can't guarantee to generate better results.

Task	Run	Correct # (strict)	Correct % (strict)	MRR(strict)	Correct # of final answer	Correct % of final answer
main	ICTQA10a	63	12.8%	0.090	26	8%
main	ICTQA10b	67	13.6%	0.100	34	10%
main	ICTQA10c	56	11.4%	0.077	20	6%

Table 4.2 Strict performance in TREC-10

Task	Run	Correct # (lenient)	Correct % (lenient)	MRR(lenient)	Correct # of final answer	Correct % of final answer
main	ICTQA10a	74	15.0%	0.102	26	8%
main	ICTQA10b	76 .	15.4%	0.109	34	10%
main	ICTQA10c	66	13.4%	0.089	20	6%

Table 4.3 Lenient performance in TREC-10

Table 4.4 shows some statistical results by question types. Our system is pleasant for the NATIONALITY, DURATION, CURRENCY, LOCATION and No Answer question types, but disappointing on the DATE, PERSON, MONEY and REASON question types, though these question types are easy to determine. The main reason is that some bugs exist in our ranking strategy when there are too many candidate named entities matching these question types. We try to find more detailed ranking strategies to solve this problem in the future work. We also try to introduce some syntactic and semantic parsing technology to solve other problems, especially the NAME(*What is NP?*) question type, which we badly handle in TREC-10.

Question type	# of question	Correct #	Correct %	MRR
PERSON	53	5	9.43%	0.04
LOCATION	29	10	34.48%	0.24
MONEY	2	0	0	0
PERCENT	5	1	20%	0.20
DATE	33	6	18.18%	0.16
NUMBER	13	3	23.1%	0.13
DURATION	4	3	75%	0.75
MEASUREMENT	30	7	23.33%	0.217
CURRENCY	5	2	40%	0.40
NATIONALITY	2	2	100%	0.625
REASON	4	0	0	0
NAME	130	18	13.85%	0.08
No Answer	49	15	30.61%	0.306

Table 4.4 Lenient performance for each question type

5. Conclusion

This year we participate in the TREC conference for the first time, the main goal is to understand the process and the ideas of the TREC conference. We spend much more time on this goal than we do in system construction. We think we have achieved this goal though our results are not so satisfactory.

Before attending TREC-10, we have had some experiences in Chinese information processing. After attending TREC-10, we have got some experiences in English information processing and we will try to apply these useful experiences in our future work.

Acknowledgements

This research is supported by the national 973 fundamental research program under contact of G1998030413. We give our thanks to all the people who have contributed to this research and development, in particular Yanbo Han, Li Guo, Qun Liu, Hai Zhuge, Zhihua Yu and Lei Cheng.

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] Ellen M. Voorhees, Dawn M. Tice, The TREC-8 Question Answering Track Evaluation, In *The Eighth Text REtrieval Conference (TREC 8),1999.*

[10] Amit Singhal, Steven Abney, Michiel Bacchiani, Michael Collins, David Hindle and Fernando Pereira, AT&T at TREC-8, In *The Eighth Text REtrieval Conference (TREC 8),1999.*

[11] Toru Takaki, NTT DATA: Overview of system approach at TREC-8 ad-hoc and question answering, In *The Eighth Text REtrieval Conference (TREC 8),1999.*

[12] Rohini Srihari and Wei Li, Information Extraction Supported Question Answering, In *The Eighth Text REtrieval Conference (TREC 8),1999.*

[13] John Prager, Dragomir Radev, Eric Brown, Anni Coden and Valerie Samn, The Use of Predictive Annotation for Question Answering in TREC-8, In *The Eighth Text REtrieval Conference (TREC 8),1999.*

[14] Jinxi Xu and W. Bruce Croft, Query Expansion Using Local and Global Document Analysis, In *The Eighth Text REtrieval Conference (TREC 8),1999.*

[15] Xiaoyan Li and W. Bruce Croft, Evaluating Question-Answering Techniques in Chinese, Computer Science Department University of Massachusetts, Amherst, MA , *2001.*

[16] Chengxiang Zhai, Peter Jansen, Norbert Roma, Emilia Stoica, David A. Evans, Optimization in CLARIT TREC-8 Adaptive Filtering, In *The Eighth Text REtrieval Conference (TREC 8),1999.*

[17] Lide Wu, Xuanjing Huang,Yikun Guo, Bingwei Liu, Yuejie Zhang, FDU at TREC-9: CLIR, QA and Filtering Tasks. In *The Ninth Text REtrieval Conference (TREC 9),2000.*

[18] Stephen Robertson, David A. Hull, The TREC-9 Filtering Track Final Report. In *The Ninth Text REtrieval Conference (TREC 9),2000.*

[19] Rocchio, J. J. Relevance Feedback in Information Retrieval. In *The SMART Retrieval system, Prentice-Hall, Englewood NJ. 1971, 232-241.*

CL Research Experiments in TREC-10 Question Answering

Kenneth C. Litkowski
CL Research
9208 Gue Road
Damascus, MD 20872
ken@clres.com

Abstract

CL Research's question-answering system (DIMAP-QA) for TREC-10 only slightly extends its semantic relation triple (logical form) technology in which documents are fully parsed and databases built around discourse entities. Time constraints did not allow us to make various changes planned from TREC-9. TREC-10 changes made fuller use of the integrated machine-readable lexical resources and extended the question-answering capability to handle list and context questions. Experiments to further exploit the dictionary resources were not fully completed at the time of the TREC-10 submission, affecting planned revisions in other QA components.

The official score for the main TREC-10 QA task was 0.120 (compared to 0.135 in TREC-9), based on processing 10 of the top 50 documents provided by NIST, compared to the average of 0.235 for 67 submissions. Post-hoc analysis suggests a more accurate assessment of DIMAP-QA's performance in identifying answers is 0.217. For the list task, the CL Research average accuracy was 0.13 and 0.12 for two runs compared to the average of 0.222. For the context questions, CL Research had mean reciprocal rank score of 0.178, 5th of the 7 submissions.

1. Introduction

TREC-10 DIMAP-QA proceeded from last year's version (Litkowski, 2001) primarily by attempting to integrate dictionary definition lookup into **what** and **who** questions, extending our success from using definitions in handling **where** questions using the dictionary. However, our strategy for **where** questions did not generalize, in part because of the poor retrieval performance of the NIST top documents when dealing with definition questions. We also added mechanisms for answering list questions, involving only a test for a numerical term in the question, but keeping the remaining functionality the same as for **what** questions, just returning the number of answers required. For context questions, we made no changes whatever to the system, yet still managed to obtain results consistent with our general question answering, even for the later questions of a given set.

DIMAP-QA is a part of the DIMAP dictionary creation and maintenance software, which is primarily designed for making machine-readable dictionaries machine-tractable and suitable for NLP tasks, with some components intended for use as a lexicographer's workstation.[1] The TREC QA track provides an opportunity for experimenting with question answering using syntactical clues and semantic evidence from use of computational lexical resources (dictionary and thesaurus).

2. Problem Description

Participants in the main TREC-10 QA track were provided with 500 unseen questions to be answered from the TREC CD-ROMs, (about 1 gigabyte of compressed data), containing documents from the *Foreign Broadcast Information Service*, *Los Angeles Times*, *Financial Times*, *Wall Street Journal*, *Associated Press Newswire*, and *San Jose Mercury News*. These documents were stored with SGML formatting tags. 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 list task were given 25 questions, each of which required a specified number of answers; the top 50 documents were also provided. Participants in the context task were given 10 question sets, varying in number from 3 to 9 questions; the top 50 documents retrieved using the first question of each set were also provided.

[1] DIMAP, including the question-answering component, is available from CL Research. Demonstration versions are available at http://www.clres.com.

Participants in the main were required to answer the 500 questions in 50-byte answers. For each question, participants were to provide 5 answers, with a score attached to each for use in evaluating ties.[2] In TREC-10, a valid answer could be NIL, indicating that there was no answer in the document set; NIST included 49 questions for which no answer exists in the collection. For the list questions, participants were to return exactly the number of answers specified in the question. For the context questions, 5 answers were to be provided for each question of the set; the questions were constructed in a way so that later questions of the set depended on the answers to the earlier questions. NIST evaluators then judged whether each answer contained a correct answer. Scores were assigned as the inverse rank for the main and the context tasks. If question q contained a correct answer in rank r, the score received for that answer was 1/r. If none of the 5 submissions contained a correct answer, the score received was 0. If a NIL answer was returned, and was deemed valid, its position in the ranked list of answers was used as the rank. The final score was then computed as the average score over the entire set of questions. For the list questions, the "average accuracy" was computed as the number of correct answers divided by the number of required answers.

CL Research submitted 5 runs, 2 each for the main task and the list task and one for the context task. For the main and list tasks, one run analyzed only the top 10 documents and the other only the top 20 documents, to examine whether performance was degraded in going from 10 to 20 documents. For the context task, only the top 10 documents were included in attempting to answer each of the questions in the set.

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 (4) a question-answering program that (a) parsed the question into the same structure for the documents, except with an unbound variable, and (b) matched the question database records with the document database to answer the question. The matching process first identified candidate sentences from the database, extracted short answers from each sentence, developed a score for each sentence, and chose the top 5 answers for submission. For the list task, the specified number of answers was submitted.

3.1 Sentence Identification in Documents

The parser (described more fully in the next section) contains a function to recognize sentence breaks. However, the source documents do not contain crisply drawn paragraphs that could be submitted to this function. Thus, a sentence could be split across several lines in the source document, perhaps with intervening blank lines and SGML formatting codes. As a result, it was first necessary to reconstruct the sentences, interleaving the parser sentence recognizer.

At this stage, we also extracted the document identifier and the document date. Other SGML-tagged fields were not used. The question number, document number, and sentence number provided the unique identifier when questions were answered.

For TREC-10, 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 9889 documents from which 225,248 sentences were identified and presented to the parser. Thus, we used an average of 22.8 sentences per document (down from 28.9 in TREC-9 and 31.9 in TREC-8) or 228 sentences for the 10-document set and 456 for the 20-document set.

3.2 Parser

The parser in DIMAP (provided by Proximity Technology, Inc.) is a grammar checker that uses a context-sensitive, augmented transition network grammar of 350 rules, each consisting of a start state, a condition to be satisfied (either a non-terminal or a lexical category), and an end state. Satisfying a condition may result in an annotation (such as number and case) being added to the growing parse tree. Nodes (and possibly further annotations, such as potential attachment points for prepositional phrases) are added to the parse tree when reaching some end states. The

[2] Although this statement appears in one of the problem specifications, the score is not used and only the position of the answer is considered.

parser is accompanied by an extensible dictionary containing the parts of speech (and frequently other information) associated with each lexical entry. The dictionary information allows for the recognition of phrases (as single entities) and uses 36 different verb government patterns to create dynamic parsing goals and to recognize particles and idioms associated with the verbs (the context-sensitive portion of the parser).

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. The parser does not always produce a correct parse, but is very robust since the parse tree is constructed bottom-up from the leaf nodes, making it possible to examine the local context of a word even when the parse is incorrect. In TREC-10, parsing exceptions occurred for only 543 sentences out of 225069 (0.0024, up from 0.0002), with another 179 "sentences" (usually tabular data) not submitted to the parser. Usable output was available despite the fact that there was at least one word unknown to the parsing dictionary in 10,916 (4.8 percent, down from 7.9 percent). For TREC-10, we were able to make use of the integrated dictionary to dynamically create entries for the parsing dictionary.

3.3 Document and Question Database Development

A key step of DIMAP-QA is analysis of the parse tree to extract semantic relation triples and populate the databases used to answer the question. A **semantic relation triple** consists of a discourse entity, a 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.

The first step of discourse processing is identification of suitable discourse entities. This involves analyzing the parse tree **node** to extract numbers, adjective sequences, possessives, leading noun sequences, ordinals, time phrases, predicative adjective phrases, conjuncts, and noun constituents as discourse entities. To a large extent, named entities, as traditionally viewed in information extraction, are identified as discourse entities (although not specifically identified as such in the databases).

The semantic relations in which entities participate are intended to capture the semantic roles of the entities, as generally understood in linguistics. This includes such roles as agent, theme, location, manner, modifier, purpose, and time. For TREC-10, we did not fully characterize the entities in these terms, but generally used surrogate place holders. These included "SUBJ," "OBJ", "TIME," "NUM," "ADJMOD," and the prepositions heading prepositional phrases. Appositive phrases were characterized by identifying the sentence word they modified and the beginning and ending words of the phrase; their use is described particularly for answering **Who** and **What** questions.

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.

The semantic relation and the governing word were not identified for all discourse entities, but a record for each entity was still added to the database for the sentence. Overall,2,174,332 semantic relation triples were created in parsing the 225,248 sentences, an average of 9.7 triples per sentence (about the same as in TREC-9).

The same functionality was used to create database records for the 500 questions. The same parse tree analysis was performed to create a set of records for each question. The only difference is that one semantic relation triple for the question contained an unbound variable as a discourse entity, corresponding to the type of question. The question database contained 1576 triples, an average of 3.15 triples per question. This is down from 3.3 per question in TREC-9 and 4.5 triples per question in TREC-8. This is indicative of the fact that the questions were "simpler", making them more difficult to answer, since there was less information on which to match.

3.4 Lexical Resources

A major component of the question-answering system is an integrated machine-tractable dictionary and thesaurus. These were provided in machine-readable form by The Macquarie Library Pty Ltd of Australia. The dictionary, known as Big Mac (The Macquarie Dictionary 1997), was converted into a format suitable for uploading into DIMAP dictionaries, during which most of the raw data were put into specific fields of a DIMAP dictionary (e.g., headword, part of speech, definitions, example usages, and many "features" characterizing syntactic properties and other information, particularly a link to Macquarie's thesaurus and identification of a "derivational" link for undefined words to their root form).

After conversion and upload, the entire dictionary of 270,000 definitions was parsed to populate the raw dictionary data by adding semantic relations links with other words. The most important result was the identification of the hypernyms of each sense. Other relations include synonyms (discernible in the definitions), typical subjects and objects for verbs, and various semantic components (such as manner, purpose, location, class membership, and class inclusion). This dictionary, accessed during the question-answering process, is thus similar in structure to MindNet (Richardson, 1997). For TREC-10, the entire dictionary was reparsed to reflect improvements in the semantic creation techniques since TREC-9.

The Macquarie thesaurus is in the form of a list of the words belonging to 812 categories, which are broken down into paragraphs (3 or 4 for each part of speech) and subparagraphs, each containing about 10 words that are generally synonymous. With a set of perl scripts, the thesaurus data has been inverted into alphabetical order, where each word or phrase was listed along with the number of entries for each part of speech, and an entry for each distinct sense identifying the category, paragraph, and subparagraph to which the word or phrase belongs.

The resultant thesaurus is thus in the precise format of the combined WordNet index and data files (Fellbaum, 1998), facilitating thesaurus lookup.

3.5 Question Answering Routines

For TREC-10, 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 list task, and one overall database to handle each of the questions in each context set. 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 three 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 for creation of the answer files submitted to NIST. Few changes were made in each of these steps from TREC-9, so the description is largely the same, with some discussion of changes planned but not implemented in time for TREC-10.

3.5.1 Identification of Key Question Elements

As indicated above, one record associated with each question contained an unbound variable as a discourse entity. The type of variable was identified when the question was parsed and this variable was used to determine which type of processing was to be performed.

The question-answering system categorized questions into six types (usually with typical question elements): (1) **time** questions ("when"), (2) **location** questions ("where"), (3) **who** questions ("who" or "whose"), (4) **what** questions ("what" or "which," used alone or as question determiners), (5) **size** questions ("how" followed by an adjective), and (6) **number** questions ("how many"). Other question types not included above (principally "why" questions or non-questions beginning with verbs "name the ...") were assigned to the **what** category, so that question elements would be present for each question. **What** questions were further analyzed to determine if they have a number modifying the head noun, in which case these were treated as list questions. (A few questions in the main task were thereby turned into list questions, limiting the number of answers returned.)

Some adjustments to the questions were made. There was a phase of consolidating triples so that contiguous named entities were made into a single triple. Then, it was recognized that questions like "what was the year" or "what was the date" and "what was the number" were not **what** questions, but rather **time** or **number** questions. Questions containing the phrase "who was the author" were converted into "who wrote"; in those with "what is the name of", the triple for "name" was removed so that the words in the "of" phrase would be identified as the principal noun. Other phraseological variations of questions are likely and could be made at this stage.

Once the question type had been determined and the initial set of sentences selected, further processing took place based on the question type. Key elements of the question were determined for each question type, with some specific processing based on the particular question type. In general, we determined the key noun, the key verb, and any adjective modifier of the key noun for each question type. For **who** questions, we looked for a year restriction. For **what** questions, we looked for a year restriction, noted whether the answer could be the object of the key verb, and formed a base set of thesaurus categories for the key noun. For both **who** and **what** definition questions, an attempt was made to find the key noun in the Macquarie dictionary, creating a list of content words in its definitions for comparison with discourse entities in the sentences. For **where** questions, we looked up the key noun in the Macquarie dictionary and identified all proper nouns in all its definitions (hence available for comparison with short answers or other proper nouns in a sentence). For **size** questions, we identified the "size" word (e.g., "far" in "how far"). For **number** questions, we also looked for a year restriction.

3.5.2 Coarse Filtering of Sentences

The second step in the question-answering phase was the development of an initial set of sentences. In previous years, this was the first step, but with the addition of definition lookup as part of the analysis of question type, this was moved. Basically, the discourse entities in the question records are used to filter the records in the document database. However, this list is extended when a "definition" question is recognized, by adding words from the definition as obtained from the dictionary.[3] Since a discourse entity in a record could be a multiword unit (MWU), the initial filtering used all the individual words in the MWU. Question and sentence discourse entities were reduced to their root form, eliminating issues of tense and number. All words were reduced to lowercase, so that issues of case did not come into play during this filtering step. Finally, it was not necessary for the discourse entity in the sentence database to have a whole word matching a string from the question database. Thus, in this step, all records were selected from the document database having a discourse entity that contained a substring that was a word in the question discourse entities.

MWUs were analyzed in some detail to determine their type and to separate them into meaningful named entities. We examined the capitalization pattern of a phrase and whether particular subphrases were present in the Macquarie dictionary. We identified phrases such as "Charles Lindbergh" as a person (and hence possibly referred to as "Lindbergh"), "President McKinley" as a person with a title (since "president" is an uncapitalized word in the Macquarie dictionary), "Triangle Shirtwaist fire" as a proper noun followed by a common noun (hence looking for either "Triangle Shirtwaist" or "fire" as discourse entities).

The join between the question and document databases produced an initial set of unique (document number, sentence number) pairs that were passed to the next step. In TREC-10, each hit of a discourse entity in a sentence added a score of 5 points to the sentence; this score determined the order in which sentences were further evaluated. Sentences with MWU discourse entities having a question discourse entity as a substring were selected during this screening, but were given no points and hence examined last in the detailed evaluation of the sentences.

3.5.3 Extraction of Short Answers

After the detailed question analysis, processing for each question then examined each selected sentence, attempting to find a viable short answer and giving scores for various characteristics of the sentence. For **time**, **location**, **size**, and **number** questions, it was

[3] In TREC-9, there were 35 "definition" questions. In TREC-10, the number increased to 165 (including **where** questions). In addition, 43 questions were identified as susceptible of "dictionary support", where the answer could be looked up in the dictionary, with the expectation that the question elements would be discernible in the definition of the answer.

possible that a given sentence contained no information of the relevant type. In such cases, it was possible that a given sentence could be completely eliminated. In general, however, a data structure for a possible answer was initialized to hold a 50-byte answer and the sentence was assigned an initial score of 1000. An initial adjustment to the score was given for each sentence by comparing the question discourse entities (including subphrases of MWUs) with the sentence discourse entities, giving points for their presence and additional points when the discourse entities stood in the same semantic relation and had the same governing word as in the question. For **who**, **what,** and **location** definition questions, a background array of content words from the definitions was developed for later comparison with the answer.

1. Time Questions - The first criterion applied to a sentence was whether it contained a record that has a TIME semantic relation. The parser labels prepositional phrases of time or other temporal expressions (e.g., "last Thursday"); database records for these expressions were given a TIME semantic relation. We also examined triples containing "in" or "on" as the governing word (looking for phrases like "on the 21st", which may not have been characterized as a TIME phrase) or numbers that could conceivably be years. After screening the database for such records, the discourse entity of such a record was then examined further. If the discourse entity contained an integer or any of its words were marked in the parser's dictionary as representing a time period, measurement time, month, or weekday, the discourse entity was selected as a potential answer.

2. Where Questions - Each sentence was examined for the presence of "in", "at", "on", "of", or "from" as a semantic relation, or the presence of a capitalized word (not present in the question) modifying the key noun. The discourse entity for that record was selected as a potential answer. Discourse entities from "of" triples were slightly disfavored and given a slight decrease in score. If the answer also occurred in a triple as a governing word with a HAS relation, the discourse entity from that triple was inserted into the answer as a genitive determiner of the answer.

3. Who Questions - The first step in examining each sentence looked for the presence of appositives, relative clauses, and parentheticals. If a sentence contained any of these, an array was initialized to record its modificand and span. The short answer was initialized to the key noun. Next, all triples of the sentence were examined. First, the discourse entity (possibly an MWU) was examined to determine the overlap between it and the question discourse entities. The number of hits was then added to all appositives which include the word position of the discourse entity within its span. (A sentence could have nested appositives, so the number of hits can be recorded in multiple appositives.)

The next set steps involved looking for triples whose governing word matched the key verb, particularly the copular "be" and the verb "write". For copular verbs, if the key noun appeared as the subject, the answer was the object, and vice versa. For other verbs, we looked for objects matching the key noun, then taking the subject of the verb as the answer.

Another major test of each discourse entity that contained a substring matching the key noun was whether it was modified by an appositive. If this was the case, the appositive was taken as a possible short answer; the discourse entities of the appositive were then concatenated into a short answer. Numerical and time discourse entities were also examined when there was a date restriction specified in the question to ascertain if they could be years, and if so, whether they matched the year restriction. In the absence of a clear sentence year specification, the document date was used.

4. What Questions - The first step in examining the sentences was identical to that of the **who** questions, namely, looking for appositives in the sentence and determining whether a discourse entity had overlaps with question discourse entities. If the key noun was a part of a discourse entity, we would note the presence of the key noun; if this occurrence was in a discourse entity identified as an adjective modifier, the modificand was taken as a short answer and if this short answer was itself a substring of another sentence discourse entity, the fuller phrase was taken as the answer. Similarly, when the key noun was a proper part of a discourse entity and began the phrase (i.e., a noun-noun compound), the remaining part was taken as the short answer.

As with **who** questions, if the key noun was identified as the modificand of an appositive, the appositive was taken as the possible answer. Similarly to **who** questions, we also looked for the copular "be" with the key noun as either the subject or object, taking the other as a possible answer. When the key verb was "have" and the key noun was equal to the object, the subject of "have" was taken as the short answer. In cases like these, we would also insert any adjective modifiers of the noun discourse entities at the beginning of the short answer.

If the key noun was not equal to the discourse entity of the triple being examined, we tested whether the key noun against the DIMAP-enhanced Macquarie dictionary, looking for its presence (1) in the definition of the discourse entity, (2) as a hypernym of the discourse entity, or (3) in the same Macquarie thesaurus category. (For example, in examining "Belgium" in response to the question "what country", where country is not in definition and is not a hypernym, since it is defined as a "kingdom", we would find that "country" and "kingdom" are in the same thesaurus category.) Finally, as with **who** questions, we examined TIME and number discourse entities for the possible satisfaction of year restrictions.

5. Size Questions - For these questions, each triple of a selected sentence was examined for the presence of a NUM semantic relation or a discourse entity containing a digit. If a sentence contained no such triples, it was discarded from further processing. Each numerical discourse entity was taken as a possible short answer in the absence of further information. However, since a bare number was not a valid answer, we looked particularly for the presence of a measurement term associated with the number. This could be either a modificand of the number or part of the discourse entity itself, joined by a hyphen. If the discourse entity was a tightly joined number and measu2ement word or abbreviation (e.g., "6ft"), the measurement portion was separated out for lookup. The parsing dictionary characterizes measurement words as having a "measures", "unit", "MEASIZE", or "abbr" part of speech, so the modificand of the number was tested against these. If not so present in the parsing dictionary, the Macquarie definition was examined for the presence of the word "unit". When a measurement word was identified, it was concatenated with the number to provide the short answer.

6. Number Questions - The same criterion as used in size questions was applied to a sentence to see whether it contained a record that has a NUM semantic relation. If a selected sentence had no such triples, it was effectively discarded from further analysis. In sentences with NUM triples, the number itself (the discourse entity) was selected as the potential answer. Scores were differentially applied to these sentences so that those triples where the number modified a discourse entity equal to the key noun were given the highest number of points. TIME and NUM triples potentially satisfying year specifications were also examined to see whether a year restriction was met. In the absence of a clear sentence year specification, the document date was used.

3.5.4 Evaluation of Sentence and Short Answer Quality

After all triples of a sentence were examined, the quality of the sentences and short answers was further assessed. In general, for each question type, we assessed the sentence for the presence of the key noun, the key verb, and any adjective qualifiers of the key noun. The scores were increased significantly if these key items were present and decreased significantly if not. In the absence of a clear sentence year specification (for **who**, **what**, and **number** questions containing a year restriction), the document date was used. For certain question types, there were additional checks and possible changes to the short answers.

For **location** questions, where we accumulated a set of proper nouns found in the definition of the key noun, the score for a sentence was incremented for the presence of those words in the sentence. Proper nouns were also favored, and if two answers were found, a proper noun would replace a common noun; proper nouns also present as proper nouns in the Macquarie dictionary were given additional points. Similarly, if a sentence contained several prepositional phrases, answers from "in" phrases replaced those from "of" or "from" phrases. For questions in which the key verb was not "be", we tested the discourse entities of the sentence against the DIMAP-enhanced Macquarie dictionary to see whether they were derived from the key verb (e.g., "assassination" derived from "assassinate").

For **who** and **what** questions, when a sentence contained appositives and in which satisfactory short answers were not constructed, we examined the number of hits for all appositives. In general, we would construct a short answer from the modificand of the appositive with the greatest number of hits. However, if one appositive was nested inside another, and had the same number of hits, we would take the nested appositive. For these questions, we also gave preference to short answers that were capitalized; this distinguished short answers that were mixed in case.

For these two question types, we also performed an anaphora resolution if the short answer was a pronoun. In these cases, we worked backward from the current sentence until we found a possible proper noun referent. As we proceeded backwards, we also worked from the last triple of the each sentence. If we found a plausible referent, we used that discourse entity as the short answer and the sentence in which it occurred as the long answer, giving it the same score as the sentence in which we found the pronoun.

Also, if either of these two question types was a definition question, we added points for each discourse entity that was among the content words of the definition.

For **size** questions, we deprecated sentences in which we were unable to find a measurement word. We also looked for cases in which the discourse entities in several contiguous triples has not been properly combined (such as number containing commas and fractions), modifying the short answers in such cases.

After scores have been computed for all sentences submitted to this step, the sentences are sorted on decreasing score. Finally, the output is constructed in the desired format, with the 50-byte answer extracted from the original sentences retrieved from the documents.

4. TREC-10 Q&A Results

CL Research submitted 2 runs for the main task; the official scores for these runs are shown in Table 1. The score is the mean reciprocal rank of the best answer over all 492 questions that were included in the final judgments. The score of 0.120 for run clr01b1 means that, over all questions, the CL Research system provided a sentence with a correct answer at the 8th position. This compares to an average score of 0.235 among all submissions for the TREC-9 QA 250-byte answers (i.e., a correct answer slightly worse than the 4th position).

Table 1. CL Research Run Scores

Run	Doc. Num.	Type	Score	TREC Ave.
clr01b1	10	50-byte	0.120	0.235
clr01b2	20	50-byte	0.114	0.235

The CL Research runs differ in the number of documents of the top 50 documents provided by the generic search engine that were processed. As will be discussed below, the number of documents processed reflects a point of diminishing returns in finding answers from the top documents. Table 2 shows the number of questions for which answers were found at any rank for the 492 questions.

Table 2. Answers Found (492)

Run	Doc. Num.	Type	Num	Pct.
clr00b1	10	50-byte	94	0.191
clr00b2	20	50-byte	96	0.195

For the list task, the CL Research average accuracy was 0.13 and 0.12 for two runs compared to the average of 0.222. For the context questions, CL Research had mean reciprocal rank score of 0.178, 5th of the 7 submissions.

5. Analysis

As mentioned above, we only processed the top 20 documents provided by NIST. Table 3 clearly indicates that, after the first 10 documents, the amount of incremental improvement from processing more documents is quite small. This table indicates that the CL Research results might better be interpreted in terms of the questions that could possibly have been answered.

Table 3. Highest ranked top document containing strict answer string

Document Number	Number of Questions
1-10	311
11-20	26
21-30	13
31-40	15
41-50	5
None	122

Of the 122 questions having no answer in the top 50 documents, 49 have been judged as having no answers in the document collection. Adjusting the scores to include only questions that might have been answered (311 in the 10 document analysis and 337 in the 20 document analysis), the CL Research performance, shown in Table 4, is somewhat increased. For the 10-document case, the result is 0.217, compared to the average score of 0.235, while for the 20-document case, the adjusted result is down to 0.176.

Table 4. CL Research Adjusted Run Scores

Run	Doc. Num.	Type	Score	TREC Ave.
clr01b1	10	50-byte	0.217	0.235
clr01b2	20	50-byte	0.176	0.235

A significant malfunction occurred from a program bug affecting the 20-document runs, where only two answers were submitted for the majority of questions. Notwithstanding, our system performed less well when additional documents were analyzed. It was noted earlier that the number of semantic relation triples for the questions had declined from 4.5 in TREC-8 to 3.3 in TREC-9 and 3.15 in TREC-10. One of these triples contains a question element, so the decline in information content is about one-third. As a result, this year's questions, while being simpler to state, are actually more difficult to answer. This has meant that the likelihood of the retrieval system retrieving a relevant document much less. In particular, with the large number of definition questions (estimated at 165 of 492), retrieval based solely on the word to be defined is much less likely to obtain a document with the definition.

We examined our results using 250-byte answers as well. For the 10-document case, we obtained a score of 0.296 unadjusted and 0.465 adjusted. The difference in results indicates that we are generally narrowing down the candidate sentences, but having difficulty picking out the answer string.

For TREC-9, CL Research experimented with the Macquarie dictionary in support of answers to **location** questions. This strategy worked reasonably well in TREC-10, where we obtained an adjusted score of 0.319 for this type of question. However, it did not work for **what** and **who** definition questions. Part of this failure can be attributed to our mechanism for ranking, where we had not yet implemented an adequate test for the correctness of an answer. We have made some initial changes in our strategy that clearly lead to an improvement, but we have not yet been able to assess the overall effect of these changes.

We have not yet been able to complete our characterization of failures for TREC-10. In general, the problems lie in not being able to eliminate sentences that have a lot of hits with the discourse entities in the questions, giving too much weight to this aspect. The effect is that as we add further documents, sentences not containing the correct answer are given undue' weight, crowding out sentences that contain the answer. In addition, our strategy for evaluating phrases within a sentence suffers from the same difficulty, giving too much weight to the wrong discourse entities.

6. Anticipated Improvements

As indicated earlier, we are in the process of making many changes to our question-answering system and these were not completed in time for our submission.

We are in the process of extending our document processing to incorporate discourse analysis techniques, building on the discourse entities. These changes will characterize the discourse entities semantically, in addition to resolving anaphor and definite references. Discourse structure (the relation of segments to one another) will also be captured. This amounts to tagging a document with semantic classes, named-entity types, and discourse relations over sentence spans longer than noun phrases. A key component in these characterizations will be the integrated use of WordNet and the Macquarie dictionary and thesaurus.

At the same time, we have been modifying our question-answering strategies to home in on semantic types and syntactic structures more likely to provide the answers. Initial results with definition questions show considerable improvement over our TREC-10 results. The discourse analysis has proved useful in making modifications to these QA strategies. The reverse has also proved to be the case, namely, that the QA strategies inform the manner in which we perform the discourse analysis.

7. Summary

The CL Research system was reasonably successful in answering questions by selecting sentences from the documents in which the answers occur. The system generally indicates the viability of using relational triples (i.e., structural information in a sentence, consisting of discourse entities, semantic relations, and the governing words to which the entities are bound in the sentence) for question-answering. Post-hoc analysis of the results suggests several further improvements and the potential for investigating other avenues that make use of semantic networks and computational lexicology.

References

Fellbaum, C. (1998). *WordNet: An electronic lexical database.* Cambridge, Massachusetts: MIT Press.

Litkowski, K. C. (2000). Question-Answering Using Semantic Relation Triples. In: Voorhees, E. M. & Harman, D. K. (eds.) *The Eighth Text Retrieval Conference (TREC-8)*, NIST Special Publication 500-246. Gaithersburg, MD., 349-356.

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.

The Macquarie Dictionary (A. Delbridge, Ed.). (1997). Australia: The Macquarie Library Pty Ltd

Richardson, S. D. (1997). Determining similarity and inferring relations in a lexical knowledge base [Diss], New York, NY: The City University of New York.

Topic-Specific Optimization and Structuring

A Report on CLARIT TREC-2001 Experiments

David A. Evans, James Shanahan, Xiang Tong, Norbert Roma, Emilia Stoica,
Victor Sheftel, Jesse Montgomery, Jeffrey Bennett, Sumio Fujita*, Gregory Grefenstette

Clairvoyance Corporation, Pittsburgh, PA & *Justsystem Corporation, Tokushima, Japan

1. Introduction

The Clairvoyance team participated in the Filtering Track, submitting the maximum number of runs in each of the filtering categories: Adaptive, Batch, and Routing. We had two distinct goals this year: (1) to establish the generalizability of our approach to adaptive filtering and (2) to experiment with relatively more "radical" approaches to batch filtering using ensembles of filters. Our routing runs served principally to establish an internal basis for comparisons in performance to adaptive and batch efforts and are not discussed in this report.

2. Adaptive Filtering

In previous TREC work (TREC 7 & 8), we developed an approach to adaptive filtering that proved to be robust and reasonably effective, as evidenced by the relatively strong performance of our systems [1,2]. For TREC 2001 we sought to assess the generalizability of the approach, given especially the differences this year in (a) the amount and nature of the training data and (b) the inherently "classification"-oriented (vs. "query"-oriented) task. Indeed, additional differences, such as the large numbers of expected "hits" in the test set, contributed to the special character of this year's task.

The CLARIT Filtering system is based on core CLARIT retrieval technology. In brief, the CLARIT approach uses text structures such as noun phrases, sub-phrases, and morphologically normalized words, as features or terms to represent text (or passage) or topic (query) content. Terms, in turn, are weighted based on document and corpus statistics (such as IDF and TF), and additionally can have independent coefficients to adjust weights according to processing requirements (such as updates). Information objects are modeled as vectors in a high-dimensional vector space; the Euclidean inner product gives the distance (or closeness) measure (document score) in the space. The system also has a variety of thesaurus (term-extraction) algorithms; these are used to identify characterizing terms for a document or set of documents (e.g., the set of "relevant" documents associated with a topic).

In addition to core processing, our adaptive-filtering system has several parameters, including (i) the number and type of features used to create a topic profile, (ii) the score/threshold setting, (iii) the frequency of setting updates (driven by feedback), (iv) the selection and number of (new) features added at updates, (v) the resetting of score thresholds, and (vi) the number of documents retained over time as a basis for modeling the topic (historical reference statistics and aging).

For this year's TREC adaptive filtering task, we used the same system that was used for our TREC-8 experiments [2]. However, for threshold setting and updating we further experimented with our *beta-gamma* adaptive threshold-regulation method. The method selects a threshold by interpolating between an "optimal" threshold and "zero" threshold for a specified utility function. This method can be applied both to training or sample document sets, as well as to documents that have been returned and judged during actual filtering.

The optimal threshold is the threshold that yields the highest utility over the training or accumulated reference data. Operationally, this threshold is determined by using the topic profile as a query over the reference (judged) documents to score and rank them based on their features (terms). Additionally, based on the utility function for the filter, a cumulative utility score is calculated at each rank point in ascending order. Typically, the cumulative utility score at each rank point manifests a well-behaved trend: it ascends, reaches a peak value, and descends again, eventually turning negative (as the remaining documents are mostly non-relevant). The feature score on the document at the lowest rank point where the cumulative utility score reaches its maximum is taken as the optimal threshold. The zero threshold is determined by the score on the document at the highest rank point below the optimal threshold that has a cumulative utility of zero or less.

The two parameters, *beta* and *gamma,* are used to determine the feature score—between the optimal threshold and the zero threshold—that will be used as the actual threshold for the filter. Beta attempts to account for the inherent or systematic (i.e. sampling) bias in optimal threshold calculation. Gamma makes the thresholding algorithm sensitive to the number of documents processed. The inverse (1/gamma) expresses the number of documents needed to gain reasonable confidence in the value of the score threshold (apart from the bias already accounted for through beta). The parameters are

$$\theta = \alpha\theta_{zero} + (1-\alpha)\theta_{optimal}$$

$$\alpha = \beta + (1-\beta)e^{-n\gamma}$$

where

θ is the filter threshold

n is the number of positive examples

Figure 1. Beta-Gamma Threshold Regulation

determined empirically. See Figure 1 for the formulae that use beta and gamma to calculate a filter-threshold score value.

2.1. Pre-Test Experiments and Calibration

We chose to recalibrate all parameter settings by running the system again on the tasks for TREC-8 and TREC-9 Filtering—the latter task to better approximate the classification-oriented features of the Reuters data. (In particular, we did no adaptive-filtering calibrations on any Reuters data (1987 or 1996), given our desire to assess unbiased system performance.) Our results for these preliminary tasks were quite good (actually better than any of the reported results in TREC 8 and equal to the best results in TREC 9).

2.2. Test Configuration

Our approach to testing included the following elements.

- Preprocessing: All documents, including testing documents, training documents, and topic descriptions, were pre-indexed using all single nouns, single words occurring in noun phrases, and two-word noun phrases, as recognized by the CLARIT parser.
- IDF Statistics: The IDF statistics were collected from all the training data. We did not update initial term statistics in the process of filtering as our past experiments indicate that doing so does not seem to have as much impact on the overall filtering performance as other factors.
- Term Weighting and Scoring: We used the BM25 TF formula for TF-IDF weighting; the average document length was set to 1,000. We used dot product scoring for matching documents with profiles.
- Initial Profile Term Vector: An initial profile vector was built from the original topic description and trained using the two training examples for each topic by adding 20 terms to the original profiles with coefficient of 0.1. We used a delivery-ratio estimation method to set the initial score threshold and set both *gamma* and *beta* to 0.1 for use in processing test data.
- Term Vector Updating: We used Rocchio term vector learning, but only positive examples were used to expand the profile. A centroid vector accumulator was updated whenever a profile accepted a relevant document. The top *K* terms with the highest scores were selected from the centroid and added to the original profile with a uniform coefficient. The vector was updated when a specified number of documents had been delivered since the last update or when the profile had not been updated for a time interval measured by 3,000 documents in the test stream.
- Threshold Updating: We used the same beta-gamma threshold-regulation algorithm as in TREC-8. To emphasize recent documents, we discarded any documents in the cached set of scored documents for each profile that were older than 30,000 documents, provided the cached set did not fall below a minimum of 1,000 documents. The cached documents included both true- and false-positive examples. At any point when a false-positive document scored below a "reference threshold"—equal to half of the then-current real threshold for the profile—the document was discarded.

Run	Ts Added	Coeff.	Beta	Gamma	Interval
CL01afa	20	0.10	0.10	0.10	2
CL01afb	200	0.25	0.25	0.05	2
CL01afc	200	0.25	0.25	0.05	4
CL01afd	20	0.10	0.10	0.10	4

Table 1. Configurations for Adaptive Filtering Runs

	CLT10AFA	CLT10AFB	CLT10AFC	CLT10AFD
T10U	163.7	160.4	172.9	221.8
T10SU	0.054	0.051	0.05	0.051
T10F	0.081	0.075	0.07	0.078

Table 2. Official Adaptive Filtering Test Results

	CLT10AFA	CLT10AFB	CLT10AFC	CLT10AFD
Best	3	1	1	2
> Median	13	17	16	12
Median	10	9	10	14
< Median	56	55	55	54
Worst	2	2	2	2

Table 3. Comparative T10SU Results: Number of Topics Scoring at Various Ranks

For the official TREC-2001 submission, we used the best parameter settings we discovered in our preliminary experiments (on TREC-8 and TREC-9 data). In particular, we varied only two parameters—the number of terms added at each update and slight differences in threshold convergence rates—to create four different submissions, with configurations as given in Table 1, optimized for linear utility T10U.

2.3. Test Results

Table 2 gives official results for our submitted runs and Table 3 gives comparative results. Our four official runs have similar performance. Our results were good from the point of view of "conservative" filtering (and delivery of information); we achieved an average utility of 222 for our best run, with only 30 topics scoring slight negative utility (the average of these being –4.37 and the maximum –12). However, in the context of the TREC task and the Reuters data, this is poor-to-mediocre performance.

2.4. Observations

It seems clear in retrospect that the principal problem in the system was the setting of rather high thresholds (scores), resulting in the delivery of too few documents, especially in the first stages of filtering.

In our system, the initial threshold setting is determined, in part, by the expected "delivery ratio" or density of relevant documents expected in the stream of data to be processed. In particular, before any filtering can occur, a score threshold must be established based on the available information about the topic. Two example documents alone do not constitute a sufficiently representative sample of documents for effective beta-gamma regulation. Instead, we employ a reference collection (in this case, the Reuters 1996 training documents) as a target corpus. In practice, we use the topic profile (based on terms extracted from the topic description and the two example documents) as a query to score and rank the reference documents. Note that we examine none of the documents in the reference corpus in this process and neither make nor require any information about the relevance of individual documents. We merely use the documents of the collection as an empirical test of the scoring potential of the topic profile. After scoring, we identify the rank point that corresponds to the expected ratio of relevant documents for the collection. The score on that document is used as the initial score threshold for the filter. As a concrete example, if we project the delivery ratio to be 1-in-1,000 and we have 10,000 documents in the reference collection, we would use the score on the document at rank 10 as the initial score for the filter threshold.

Given that we calibrated on TREC-8 and TREC-9 tasks, where observed delivery ratios average approximately 1-in-10,000 (TREC-9 = 0.000173 and TREC-8 = 0.00019), we began the TREC-2001 task with default assumptions of delivery that were far out of line with the actual density of topics in the Reuters 1996 Test Collection. In fact, the average density of topics in Reuters is approximately 1-in-100 (0.0125), nearly two orders of magnitude greater than in the collections we have seen in previous TREC tasks.

This discrepancy between our initial expectations (and the only ones that we might legitimately make) and the actual topic density in Reuters is an immediate source of error in our processing. It might underscore one criticism of the Reuters collection—or at least the use of Reuters subject categories in that collection—as a test bed for adaptive filtering, namely, that such "topics" with such high densities are poor representatives of real-world adaptive filtering tasks.

This problem in delivery-ratio expectations can also be regarded as an indication of a flaw in the user model we (as a group) have adopted for TREC adaptive filtering. In that model, we assume that a simple utility function—balancing the value of true versus false positives, and possibly taking into account false negatives—can represent the target outcome of a process. It is clear, however, that some expectations of delivery are also critical and are very likely a part of any user's set of expectations on filter performance.

Note, it is possible to criticize a system that requires a delivery-ratio setting to perform well in contrast to one that does not. Any system that can perform well without such a setting is to be preferred to one that cannot—*ceteris paribus*, by Occam's Razor alone. However, it is not clear that any of the more successful adaptive filtering systems that participated in TREC 2001 experiments are such systems. In fact, these better systems seem to have modeled the delivery ratio quite accurately. One wonders how such a model might have been developed on the basis of the topic statement and two sample documents alone. Of course, it is possible that such systems were simply initialized with expectations of 1- or 2-in-100 documents as candidate density. If so, these were lucky choices, indeed. And, of course, if these were just good guesses, it still remains to determine how such good guessing might be ensured, in principal, in filtering over other streams/collections, such as the ones we saw in TREC-8 and TREC-9 tasks, or such as occur in real-world applications, where information about expected density of a topic in the possibly many data streams that are accessed is not available.

	CLT10F01	CLT10F02	CLT10F03	CLT10F04
Del-Ratio	0.025	0.025	0.025	0.025
Beta	0.15	0.15	0.15	0.15
Gamma	0.01	0.01	0.005	0.005
Update	9	9	9	9
Mu	1.0	0.9	1.0	0.9
T10U	1716.8	1678.9	1714.9	1679.6
T10SU	0.1003	0.0843	0.0983	0.0813
T10F	0.1980	0.2097	0.2057	0.2148

Table 4. Results of Post-TREC Adaptive-Filtering Experiments

2.5. Follow-Up Experiments

Recognizing that our system suffered from the inappropriate expectations of density we used, we decided to re-run the experiments with explicitly different delivery-ratio settings. In particular, we wanted to assess the inherent strength (or weakness) of the system without the artificial constraint imposed by inappropriate delivery-ratio assumptions.

In a set of follow-up experiments, we re-set a variety of parameters to accommodate the special conditions of Reuters topics. We used a delivery-ratio expectation of 2.5% (0.025) to model the relatively frequent occurrence of topics. This was designed to insure that we would commence filtering with a lower expected score threshold. But given the extraordinarily high ratios of relevant documents for many topics, we might well find the lowered thresholds to be still too high. We hypothesize that, when we expect high density of a topic in a stream, we should expect any small number of sample documents (e.g., 2) to be extremely under-representative of the topic and to create a high-score bias. This is because features (terms) extracted from such non-representative documents will emphasize the distinct characteristics of those documents and will tend to select and score highly only the small subset of similar documents that share their biases. In such cases, we should depress the lower-bound score further, at least until we have achieve a feedback sample of sufficient size to insure that topic-representation biases are minimized.

As a test of this hypothesis we used lowered beta and gamma values to retard the convergence on a stable, high (optimal) threshold score. (Note that a beta = 0 would essentially deliver any document that matched on any of the features in the profile.) We also delayed the profile updates until we had accumulated sufficient judgments to yield nine true positives (along with any false positives that also were delivered in the interval). And, finally, we introduced a new parameter, *mu*, to serve as a coefficient on the filter threshold. For 0 < *mu* < 1, this effectively further lowers the threshold to allow more documents to be delivered for judgment.

The results of these follow-up experiments, given in Table 4 for the Runs labeled CLT10F01–04, demonstrate immediate, dramatic improvements. Compared to the official runs (Table 2), the improvement in performance is nearly 100% for T10SU, 250%+ for T10F, and approaching 800% for T10U. Note that these results do not reflect the effect of different term selection (or numbers of terms selected), rather derive only from (1) assuming a more appropriate delivery ratio, (2) lowering the rate of convergence on an "optimal" utility point, (3) postponing updates, and (4) further reducing the threshold.

Still, the results are sub-optimal and not at the level of the best-performing systems. We suspect that several factors are interacting to limit performance, including the fact that our core process is geared to retrieval performance and not classification. Thus, we did not model negative or border cases explicitly in developing topic profiles. In addition, the Reuters topics are quite vague and in some cases diffuse, in the sense of having a variety of subtopics. We believe that such cases are best treated with complex filters, not simple ones, capable of modeling the topic structure directly. We offer more specific thoughts on this point in the following section, in our discussion of topic-specific optimization strategies in batch filtering.

3. Batch Filtering

Traditional information retrieval approaches to batch filtering have tended to represent a category or topic using a single or monolithic filter (model) that is extracted from positive examples of the category. However, both empirical and theoretical studies in other fields such as machine learning have shown that using multiple models or ensembles of models can lead to improved performance given some weak assumptions about the constituent models [3,4,5,6]. Hansen and Salamon [3] proved that, given an ensemble of models in which the error rate of each constituent model is better than random and where each constituent model makes errors completely independent of any other, the expected ensemble error decreases monotonically with the number of constituent models. As examples of these theoretical claims, empirical studies in the field of machine learning have shown that, when weak or unstable learning algorithms, such as C4.5, are used in conjunction with ensemble techniques, the performance of these approaches can be improved significantly [4,8].

The improved performance gained from using ensemble approaches can be attributed to avoiding risks that arise from using a single model. These risks can be statistical in nature, where more than one statistical solution exists (stability). They can be algorithmic in nature, e.g., with high risk of getting stuck in local minima models. They can be representational in nature, e.g., when the space of representable models is infinite. In addition, some concepts can be very diverse and can be more accurately modeled using multiple models. Though the use of ensemble models is a relatively new, active, and very promising field of research in machine learning, very little work in information retrieval has incorporated the notion of ensemble models.

Our TREC-2001 experiments were designed explicitly to explore some of the issues in the use of ensemble filters for batch filtering. The arguments for using complex (non-unitary) filters are intuitively compelling. We recognize (a) that no single term-selection method works uniformly well for all topics and (b) that some topics are best modeled as "dispersed"—not based on a single set of features, but possibly a family of distinct sub-features. This would seem to suggest that multiple representations (hence, multiple filters) are needed. Thus, we created an approach that optimized filters on a topic-by-topic basis according to feature extraction method and filter structure. In particular, in this heterogeneous approach, filters for each topic were unique: each topic's features were derived by one of five different feature extraction techniques and each was modeled by either (i) a single (monolithic) filter, or (ii) a family of four, parallel (multiplexed) filters, or (iii) a set of *n* (cascaded) filters sequenced so that each filter after the first considered only the fallout (below-threshold-scoring) documents of the preceding filter.

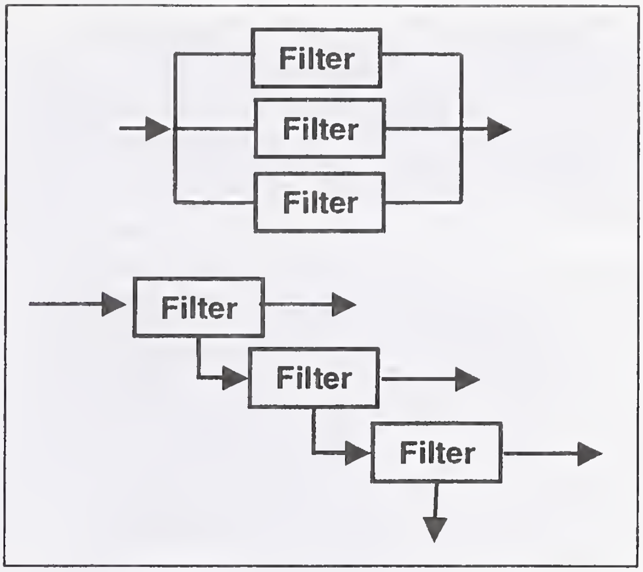


Figure 2. Schematic Representation of a Multiplex (Parallel) and Cascade (Sequential) Filter

3.1 General Description of Ensemble Batch Filtering

Ensemble filtering explores the general idea of constructing many weak or focused filters and combining these into a single highly accurate filter (using, for example, voting) in order to filter or classify an unlabeled document. Ensemble filters can be constructed and combined using various techniques that have been proposed and empirically demonstrated in the fields of machine learning and statistics. Construction approaches vary widely but can generally be placed into three broad categories: data-related methods (such as bagging and boosting); representation-based methods (such as constructive induction and alternative representations of the output space, such as error correcting output codes); and approaches that differ based upon the hypothesis search strategies employed. When it comes to aggregating the constituent filters of an ensemble, various strategies can be used, such as voting strategies as in multiplexing, a cascade (or waterfall) aggregation strategy, or aggregation strategies that are learned, as in stacked generalization [9].

For TREC 2001, we limited our exploration of ensemble filters to multiplex and cascade filters, illustrated schematically in Figure 2. Due to time and system limitations, we used simplified versions of the bagging and boosting algorithms, both of which generate component filters based upon different training data sampling procedures, to construct multiplex and cascade filters respectively.

A *multiplex filter* is a filter made up of constituent filters F_i, where the multiplex filter accepts the unlabeled document (and classifies it as positive) based on some interpretation of the independent scoring of each constituent filter F_i. In fact, there are many possible, alternative methods for interpreting the score of a set of filters, ranging from some simple combination of binary outcomes (e.g., the sum of the "votes" of each filter) to a weighted, possibly non-independent scoring based on the interaction of filters. In our experiments, we chose the simplest approach and accepted a document if the document was accepted by any one of the constituent filters.

On the other hand, a *cascade filter* is an ensemble filter that consists of an ordered list of filters $\{F_1, \ldots, F_n\}$, where each filter, F_i, consists of two outputs: one corresponding to the positive class and the other corresponding to the negative or fallout class. Each constituent filter F_i is linked to the fallout class of the filter F_{i-1}. An unlabeled document is processed by each filter F_i in left-to-right fashion. Should any filter accept the document, processing terminates and the unlabeled document is accepted by the ensemble filter. Otherwise, the subsequent filter F_{i+1} processes the unlabeled document in a similar fashion. This process repeats until either some constituent filter has accepted the document or no filter has.

$$Rocchio(t) = IDF(t) \ x \ \frac{\sum_{D \in DocSet} TF_D(t)}{R}$$

$$RocchioFQ(t) = IDF(t) \ x \ \frac{\sum_{D \in DocSet} F_D(t)}{R}$$

$$Prob1(t) = log(\frac{N-R+2}{N_t - R_t + 1} - 1) - log(\frac{R+1}{R_t} - 1)$$

$$Prob2(t) = log(R_t + 1) * Prob1(t)$$

Where,

$IDF(t) = 1 + log \frac{N}{N_t}$

$TF_d(t)$ = Frequency of term *t* or how many times the term appears in document *d*

N = Number of documents in the reference corpus

N_t = Number of documents in the reference corpus that contain term *t*

R = Number of relevant documents

R_t = Number of relevant documents containing term *t*

Figure 3. Term-Extraction Formulae

Method X Terms	CC	Roc	RocFQ	Prob1	Prob2
10	X	X	X	X	X
25	X	X	X	X	X
30	X				X
50	X	X	X	X	X
80	X	X	X	X	X
100	X	X	X	X	X
120	X	X	X	X	X
150	X	X	X	X	X
180	X				X
200	X	X	X	X	X
260	X				X
300	X	X	X	X	X
340	X				X
400	X	X	X	X	X
450	X				X
500	X				X

Table 5. Term Count and Extraction Method Combinations

In the case of ensemble filters, we used two different construction approaches for our TREC-2001 submitted runs (both of which are outlined below): one based upon the iterative modeling of fallout examples, which is a simplified version of boosting; the other based upon cross-validation, which is a simplified version of bagging. We used n-fold cross-validation [7] to choose the construction and aggregation method and make other representational decisions, such as which of several term-extraction methods and term counts to use.

3.1.1. Cross-Validation to Construct Monolithic Filters

For ease of presentation, prior to describing ensemble filter construction algorithms, we review how cross-validation was used to construct monolithic filters. Monolithic filters served as our baseline submission. The presentation is made more concrete by using the TREC-2001 filtering problem—the Reuters 1996 dataset.

For all submissions, the training corpus was divided into four folds. More specifically, the Reuters 1996 training corpus of twelve days was partitioned into four subsets denoted by *Q1, Q2, Q3,* and *Q4*, where each quarter consisted of a non-overlapping sequential sampling of a subset of the full training dataset.

Monolithic filters were constructed using either (a) the topic descriptions alone, which we subsequently refer to as "topic filters," or (b) the training corpus. To construct topic filters, we extracted filter terms from the topic descriptions and set thresholds using a beta-gamma optimization on three quarters of the data, while the unseen quarter was used as a blind test. This led to a utility measure for the test quarter. This experiment was repeated for each quarter, thereby generating four utility measures U_1, U_2, U_3, U_4. The average was taken of these four utility measures resulting in a utility measure, *AvgTopicU*, for the corresponding topic filter.

On the other hand, when constructing monolithic filters from training examples, we examined the results of various thesaurus extraction methods and corresponding term counts and chose an optimal filter representation based upon its cross validation performance. See Table 5 for a list of all combinations of thesaurus extraction methods and term counts that were examined for our TREC-2001 submissions. Note that the label "CC" in the Table stands for "CLARIT Classic," a proprietary term-extraction method. Our implementations of the methods we refer to as "Rocchio" ("Roc"), "RocchioFQ" ("RocFQ"), "Prob1," and "Prob2," are given in Figure 3.

In practice, given a training dataset that is partitioned into four folds, *Q1, Q2, Q3*, and *Q4* (for the purposes of our experiments), this optimization procedure translates into taking each combination of thesaurus extraction method *E* and number of terms *N* and performing the following steps:

For each topic *T*

1. Repeat steps 2 to 6 for all combinations *{E, N}* listed in Table 5, thereby generating an average utility for each combination.
2. Do thesaurus extraction on *Q1+Q2* with the *{E, N}* combination.
3. Optimize the threshold for *T* using beta-gamma threshold optimization over *Q3*.
4. Do a blind test on *Q4* generating a utility value U_4.
5. Repeat steps 1 to 4 for alternative combinations of *Q1, Q2, Q3, Q4*, insuring that each database subset is used as a blind test at most once. This leads to a utility value for each database subset of U_1, U_2, U_3, U_4.
6. Set average utility for this combination of *{E, N}* *AvgMonoU* to *Average*(U_1, U_2, U_3, U_4).
7. Select the combination *{E, N}* that provides the highest average utility as the optimal means of generating a monolithic filter for this topic *T*.

A utility measure for each fold of the dataset can be generated using various combinations of extraction folds and optimization folds, however, for our experiments, we limited our exploration to the following: *{Q1=3, Q2=4, Q3=2, Q4=1; Q1=1, Q2=3, Q3=4, Q4=2; Q1=2, Q2=4, Q3=1, Q4=3; Q1=1, Q2=2, Q3=3, Q4=4*}, where *1, 2, 3*, and *4* denote the folds in the training dataset and Q_i denote the variables used in the above algorithm. In deciding between modeling a topic using a monolithic filter or a topic filter, we choose the filter with the highest average utility scores on the four-fold datasets (i.e., *AvgMonoU* and *AvgTopicU*). Prior to running the selected filters on the eleven months of test data, the system retrains each filter using the entire twelve days of training, where the filter thresholds are set using the beta-gamma method on linear utility, T10U, over the entire training dataset.

3.1.2. Ensemble Filter Construction Algorithms

For our TREC-2001 submissions we developed one construction algorithm for each of the two ensemble-filter types used. Since the construction algorithm for multiplex filters is closely related to that for monolithic filter construction (described above), we begin by presenting multiplex filter construction. This algorithm is a simplified version of bagging, whereby each filter is constructed from a sampled subset of the training data based upon an n-fold partitioning of the data. This is in contrast to the more commonly used approach for bagging, where each filter is constructed from a randomly generated dataset. In this case each filter's training dataset is generated by randomly drawing, with replacement, a specified number of examples from the training dataset (typically equal to the size of the training data). We adapted the simpler strategy based on n-folds due to time and system limitations.

For our current experiments, each topic was modeled using a multiplex filter consisting of four component filters (unless there was insufficient training data for the topic), where each filter was constructed using steps 2 to 6 in the algorithm for constructing monolithic filters (above), i.e., one filter corresponding to each fold in the training data. Unlike the monolithic run (where the final monolithic filters were trained on the entire training dataset), the component filters in the multiplex filter were trained on two quarters of the training data, while the threshold was optimized using the beta-gamma method on the third quarter.

Figure 4 gives a screen shot of a multiplex filter that was constructed for topic 39 using the CLARIT AW Toolkit. It presents a multiplex filter consisting of four component filters (each depicted as a node) that were constructed using four different subsets of the training dataset. The folds that were used to generate these subsets are depicted as nodes in the top portion of the screen shot.

On the other hand, the construction algorithm for cascade filters in our TREC2001 submissions is a simplified version of boosting (a version of boosting based upon sampling). The focus of these methods is to produce a series of filters. The training set used for each filter in the series is chosen based on the performance of earlier filters in the series. In boosting, examples that are incorrectly classified by previous filters in the series are chosen more often to train subsequent filters than examples that were correctly classified. Thus boosting attempts to generate filters that are better able to predict examples for which the current ensemble's performance is poor.

For our experiments, we adapted a simplified, rather radical, approach to boosting, whereby each example that was correctly modeled using the current ensemble was not used in the construction of subsequent filters. As a result of this simplification, different stopping criteria were required to ensure termination of the algorithm. These are presented subsequently.

The approach to constructing a cascade filter for a topic *T* involves a number of steps and assumes as input three datasets *D1, D2*, and *D3*, which are respectively used for thesaurus extraction, threshold optimization, and blind testing. These datasets could correspond to the following folds in the training data: *Q1+Q2, Q3*, and *Q4* respectively. In this case the training dataset is split into four subsets/folds. The algorithm consists of two threads: the extraction thread and the threshold setting or optimization thread. Each thread results in the construction of its own cascade filter, namely, $C_{Extraction}$ and C_{Opt}. The algorithm is presented in stepwise fashion in Figure 6, where the left and right hand sides of the figure depict the extraction thread and optimization thread respectively. The algorithm is iterative in nature, whereby the first filter in the cascade, $C1_{Extraction}$, is constructed using the positive topic

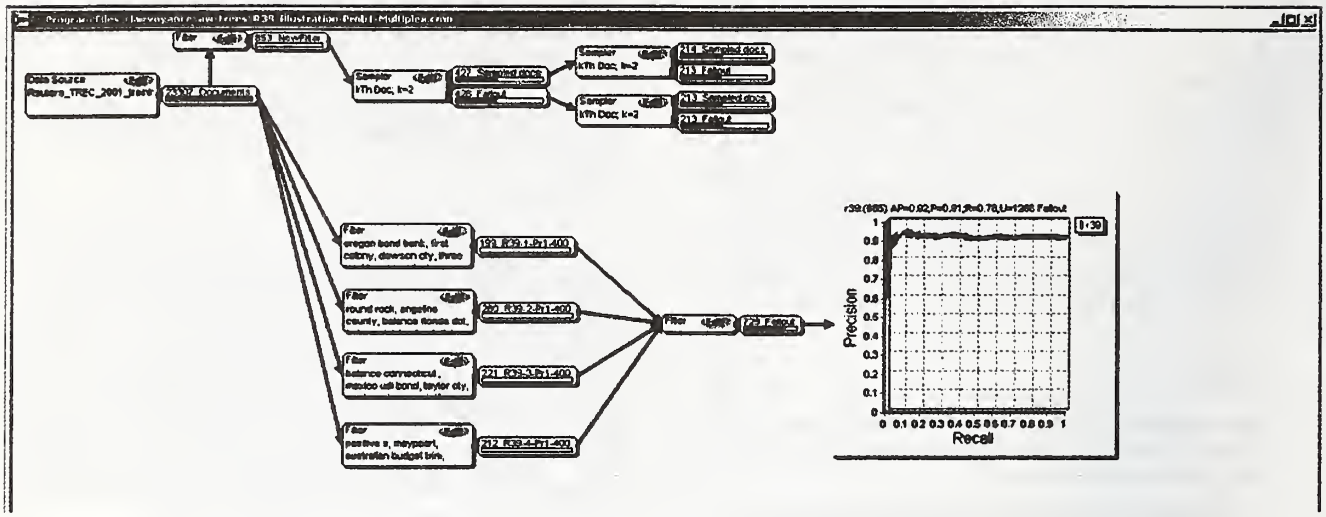


Figure 4. Example of a Multiplex Filter for Topic 39, with Performance Illustrated on the Training Corpus

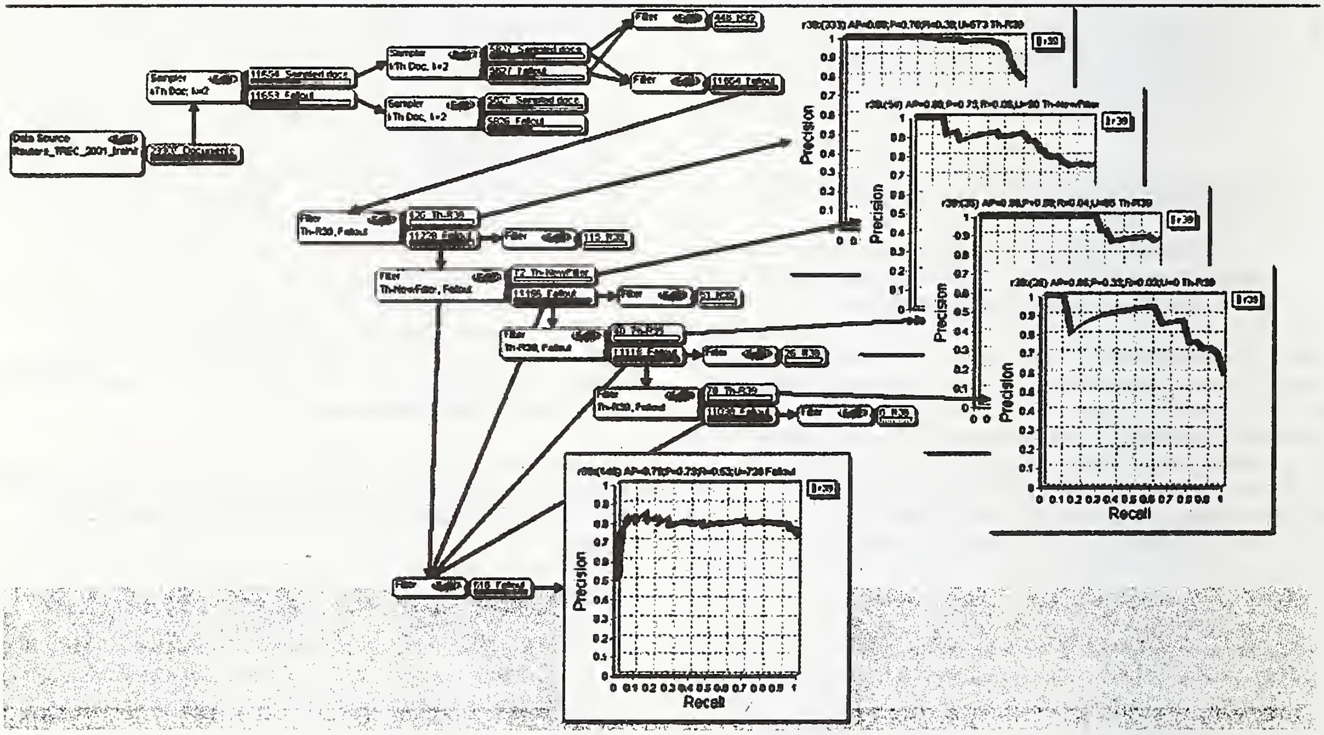
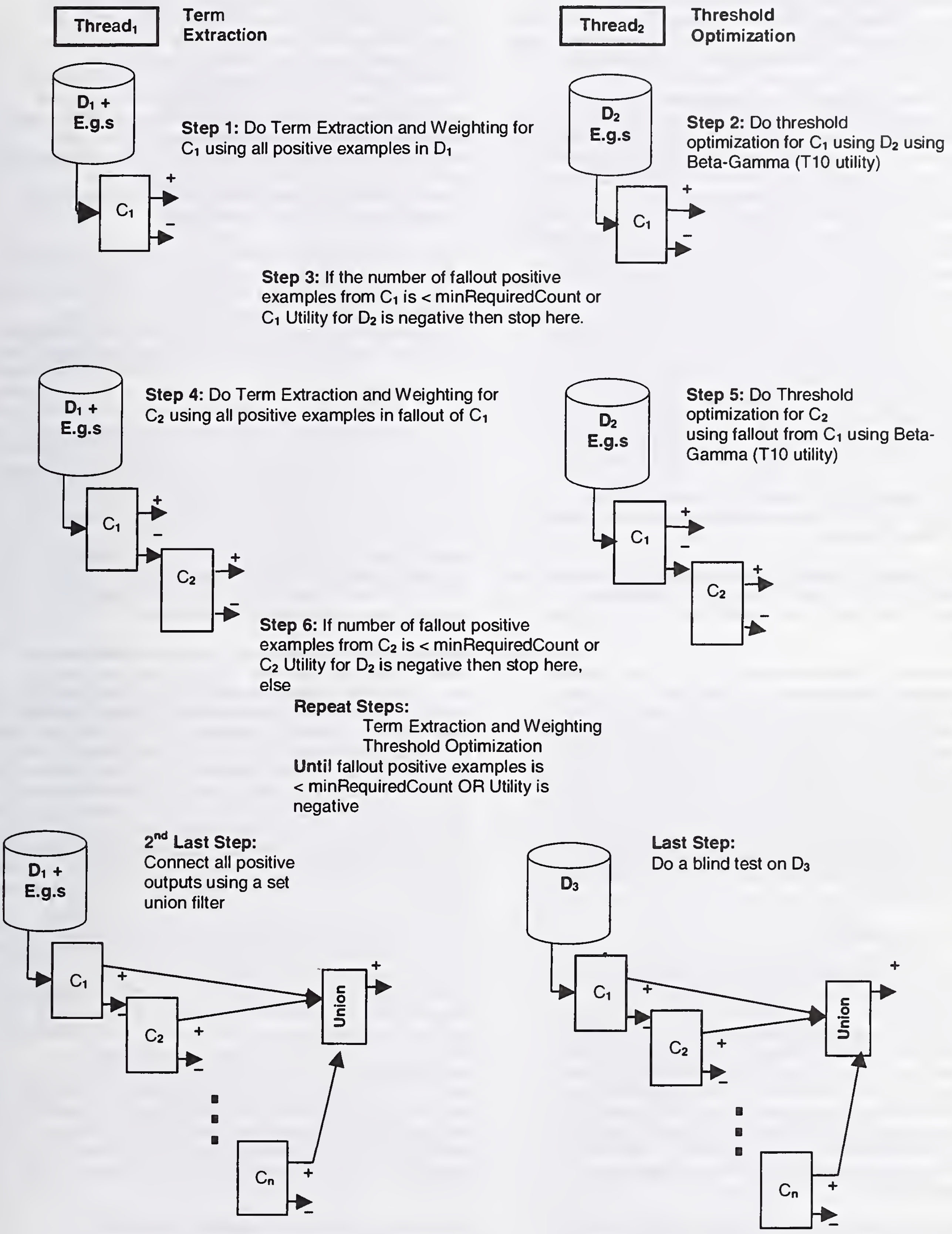


Figure 5. Example of a Cascade Filter for Topic 39, with Performance Illustrated on the Training Corpus



Step 1: Do Term Extraction and Weighting for C_1 using all positive examples in D_1

Step 2: Do threshold optimization for C_1 using D_2 using Beta-Gamma (T10 utility)

Step 3: If the number of fallout positive examples from C_1 is < minRequiredCount or C_1 Utility for D_2 is negative then stop here.

Step 4: Do Term Extraction and Weighting for C_2 using all positive examples in fallout of C_1

Step 5: Do Threshold optimization for C_2 using fallout from C_1 using Beta-Gamma (T10 utility)

Step 6: If number of fallout positive examples from C_2 is < minRequiredCount or C_2 Utility for D_2 is negative then stop here, else

```
Repeat Steps:
        Term Extraction and Weighting
        Threshold Optimization
Until fallout positive examples is
< minRequiredCount OR Utility is
negative
```

2nd Last Step: Connect all positive outputs using a set union filter

Last Step: Do a blind test on D_3

Figure 6. Schematic Representation of the Cascade-Filter Creation Procedure

examples in the extraction dataset *D1* (Step 1 in Figure 6). This cascade corresponds to the extraction cascade $C_{Extraction}$. In order to set the threshold for $C1_{Extraction}$, a second cascade filter (i.e., the optimization cascade) is constructed. The first constituent filter in this cascade is simply a copy of $C1_{Extraction}$ and is denoted as $C1_{Opt}$. To avoid clutter in Figure 6, the *Extraction* and *Opt* suffixes are dropped from the component filters names. The threshold for $C1_{Opt}$ is set using the beta-gamma method based upon the linear utility over the optimization dataset *D2* (Step 2 in Figure 6). The threshold of the $C1_{Extraction}$ filter is set to the optimized threshold of $C1_{Opt}$. Subsequently, the fallout documents from *C1* (i.e., positive examples from *D1* that are rejected by $C1_{Extraction}$) are used to construct the second filter $C2_{Extraction}$ in the cascade, provided various continuation conditions are met. These include: the number of documents in the fallout of $C1_{Extraction}$ is greater than a minimum number of documents required to construct a filter; the linear utility of the $C1_{Opt}$ over the optimization dataset is positive. The above steps of constituent filter extraction and threshold optimization (on the fallout of each preceding filter) are repeated as long as the continuation conditions are satisfied. Once any continuation conditions fail, all the positive outputs of the constituent filters of the extraction cascade $C_{Extraction}$ are connected to a union filter (2nd-Last Step in Figure 6). Subsequently this cascade filter is applied to the D3 dataset and a utility measure is obtained (Last Step in Figure 6).

Figure 5 presents a screen shot of the extraction cascade for topic 39 of the Reuters Corpus. Each filter in the cascade, in this figure, is associated with a precision-recall curve.

The above procedure is repeated such that each quarter of the training dataset serves as a blind test (D3) at most once. This results in a utility measure for each quarter. An average of these four values is then taken, resulting in an average utility, *AvgCascU*, for this cascade filter. This value serves as a comparison to other approaches used for modeling a topic. Should the cascade filter be chosen (based upon average utility) to model a topic, the cascade filter is first re-trained using three quarters of the data for extraction (*D1*), while the fourth quarter is used for threshold optimization (*D2*) and a blind test is carried out on the test dataset.

3.2. Test Configuration

We submitted two batch filtering runs: one where each topic was represented by a single or monolithic filter (*CLT10BFA*); the other, which we call the "rainbow run" (*CLT10BFB*), where each topic was represented by using one (the best) of either a topic, monolithic, multiplex, or cascade filter. Note that for both submission runs, beta was set to 0.1 and gamma was set to 0.05. The minimum number of documents required for filter construction in ensemble filters was set to five. In both cases, we chose the final representation for a topic based on which of the alternative choices yielded the highest average cross-validation utility.

In the construction of cascaded filters, the representation of each constituent filter was globally set to the optimally determined representation for monolithic filters. Since the average cross-validation utility for both monolithic and multiplex filters corresponds to the same value, a further evaluation criterion was necessary. To decide between these two approaches, we re-trained a corresponding monolithic filter using the entire twelve days of training. The beta-gamma method was used to optimize the threshold of the resulting monolithic filter. This re-trained filter was subsequently used for retrieval over the twelve days of training data and a corresponding global utility was calculated. Similarly, the extracted multiplex filter was used to filter documents from the twelve days of training and a corresponding global utility calculated. The approach with the highest global utility was chosen as the approach to model the associated topic.

As a benchmark for our ensemble approaches, we carried out a batch filtering experiment using our traditional information retrieval system in conjunction with an optimization strategy for identifying the term extraction method and term count similar to that outlined above. This experiment modeled each topic using a monolithic filter.

3.3. Test Results

Table 6 presents a summary of various batch filtering runs in terms of the linear utility (T10U) and normalized linear utility (T10SU). Row one of this table presents the median of all submitted runs (from all groups) for TREC-2001 batch filtering. The second and third rows summarize the results for our two submitted runs. Our official heterogeneous or rainbow submission had an average normalized utility of 0.152 and F-utility of 3371. The *CLT10BFC* run represents the result of our benchmark run, which was inadvertently not submitted.

In the context of this year's task, our Batch results are weak. In follow-up analyses, we determined that some of the performance shortfalls were due to processing errors under our control. For example, the test data had become corrupted in our translation of the NIST sources into our processing format; this led to our losing actual test documents, which naturally limited our results in some cases. In another more serious case, we inadvertently failed to reset a critical system parameter—one that sets an upper bound on the number of documents that are considered in any set of documents to be compared to a topic profile—when the system moved from training to testing data. In effect, we only considered about one-third of the test documents that should have been considered as candidates for matching each topic. This particular problem affected both our routing-based baseline run and our heterogeneous run. When we corrected these problems and re-ran over the correct version of the testing data, we saw immediate improvement in both runs. In particular the heterogeneous approach improved by about 60% (from 0.152 to 0.239 normalized utility), making it essentially indistinguishable from our routing-based results. The *CLT10BFA2*, *CLT10BFB2*, and *CLT10BFC2* runs correspond to re-runs of our official submissions and our benchmark run, respectively, where both the dataset errors and the critical retrieval parameter have been corrected. Here, our benchmark run yields a performance of 0.257.

The remainder of Table 6 relates to some post-TREC experiments that addressed various problems that occurred during the preparation of our final submissions.

Run Description		T10SU	T10U
Median for all Submitted Runs		0.256	N/A
Submitted Results	CLT10BFA	0.147	N/A
	CLT10BFB	0.152	3371
Post-TREC Runs	CLT10BFA2	0.237	5834
	CLT10BFB2	0.210	4925
	CLT10BFC	0.234	5453
	CLT10BFC2	0.257	5895
	GlobalCascade	0.220	5323
	LocalCascade	0.195	4882
	Multiplex	0.225	5665

Table 6. Results of Post-TREC Batch Experiments

These experiments were all conducted on the corrected dataset and with properly set retrieval parameters. *GlobalCascade* relates to a run where each topic is represented using a cascade filter where the extraction method and associated term count for each constituent filter in the cascade has a fixed setting. In particular, all constituent filters in the cascade are given the same settings as were determined to be optimal for the corresponding monolithic filter for the topic (in *CLT10BFA2*). *LocalCascade* denotes an experiment where each topic is represented using a cascade filter. In this case the extraction method and associated term count for each constituent filter in the cascade is optimized locally. *Multiplex* relates to an experiment where each filter is represented using a multiplex filter.

3.4. Observations

Our work in batch filtering this year represented an initial attempt at the construction of ensemble filters. Due to system limitations and time constraints, our experiments were accomplished using simplified versions of bagging and boosting algorithms for the construction of multiplex and cascade filters respectively. Even though the performance of the current system is only comparable to the TREC median, performance should improve with full implementations of bagging and boosting algorithms along with more comprehensive experimentation. Current work is addressing both of these issues.

Though our proposed approaches to ensemble filters model subtopic structure, albeit in a limited fashion, a more natural means of identifying and thereby representing topic structure can be achieved using clustering. This forms a very important part to our current work in this area.

Our analysis suggests that ensemble filters perform better than monolithic filters for certain classes of topics. To take advantage of the potential boost in performance that would come from topic structuring, any operational system would have to be able to predict whether a particular filtering task (topic) is best modeled via a monolithic or an ensemble filter. Our hypothesis is that, if there is sufficient training data (or relevance judgments in a running filter), then multiplex filters will outperform monolithic ones. In such cases, then, the task becomes choosing between multiplex and cascade approaches. Our hypothesis is that cascading is optimal when the topic is vague or diffuse. Thus, to make a principled decision about which approach to select, we need a means of diagnosing topic structure. We have begun work on the development of a simple method to accomplish this.

In our report at the TREC 2001 Meeting, we described retrospective results that simulated such an ideal choice—essentially, the best performing method for each topic as seen in the homogeneous runs represented by our baseline monolithic run (*CLT10BFB2*), our multiplex run (*Multiplex*), and our cascade run (*GlobalCascade*)—which we called "MadMax." A striking feature of the MadMax simulation run was the remarkable performance of cascade filters in the case of twelve topics, significantly exceeding the reported official TREC maximum results. *We feel obligated to report now that, in work since the time of the Meeting, we have not been able to duplicate the extraordinary performance of the cascade runs and now believe we had corrupted data in reporting those results.* In repeated experiments with cascade filters—albeit with the limited conditions we employed in our original runs—we have achieved individual topic performance that equals or exceeds the reported TREC maximum on six topics; but only one of these is significantly better than the maximum.

We continue to believe that successful, robust filtering (especially in distinction to classification) will require topic-specific optimizations, including topic structuring. We have only just begun to explore this problem. We will continue to develop topic-structuring techniques and apply them in future TREC experiments.

References

[1] 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.

[2] 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.

[3] Hansen L, and Salamon P. 1990. *Neural network ensembles*. IEEE Transactions on PAMI 12:993–1001

[4] Quinlan JR, Bagging, boosting, and C4.5. In *Proc.Fourteenth National Conference on Artificial Intelligence*, 1996.

[5] Schapire RE, The strength of weak learnability, *Machine Learning*, 5(2):197–227, 1990.

[6] Breiman, L, *Bagging Predictors*, Machine Learning, 24(2):123-140, 1996.

[7] Stone, M (1974), "*Cross-validatory choice and assessment of statistical predictions*", Journal of RSS B 36, 111–147.

[8] Schapire RE and Singer Y, BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[9] Wolpert, DH *Stacked generalization*. Neural Networks 5 (1992) 241–259.

TREC-10 Shot Boundary Detection Task: CLIPS System Description and Evaluation

Georges M. Quénot

CLIPS-IMAG, BP53, 38041 Grenoble Cedex 9, France
Georges.Quenot@imag.fr

Abstract

This paper presents the system used by CLIPS-IMAG to perform the Shot Boundary Detection (SBD) task of the Video track of the TREC-10 conference. Cut detection is performed by computing image difference after motion compensation. Dissolve detection is performed by the comparison of the norm over the whole image of the first and second temporal derivatives. An output filter is added in order to clean the data of both detector and to merge them into a consistent output. This system also has a special module for detecting photographic flashes and filtering them as erroneous "cuts". Results obtained for the TREC-10 evaluation are presented. The system appear to perform in a very good way for cut transitions and within the average for gradual transitions.

1 Introduction

Temporal video segmentation has very important applications in video document indexing and retrieval, in information or emission type filtering and in video document browsing among many others. It must be distinguished from spatial (extracting objects) and spatio-temporal (tracking objects) segmentations that will not be considered here (even though probably also useful for video document indexing). This work focuses solely on the segmentation of the image track of video documents. The segmentation process consists mainly in detecting "transition effects" between "homogeneous segments" (shots), the definition of which is rather application-dependent. Transition effects may be roughly classified into three categories:

- "cuts": sharp transitions between two consecutive images, the second image is completely or almost completely different from the first one,
- "dissolves": continuous transition between two continuous sequences by a progressive linear combination of them (this includes "fades in" and "fades out"),
- "others": all other type of transitions, including all possible special effects.

Several levels of difficulty arise within the global segmentation task:

- The most easy and low level is to find "cuts" and "dissolves" between almost static images. Specific filters can be quite easily designed for this task.
- More difficult is the detection of "dissolves" in the general case and other effects (such as wipes for instance). Higher level tools can also be developed for detecting such difficult to find transitions or as well these and the simple ones simultaneously [1].
- Finally, the highest level of difficulty is to find among the identified transitions or segments, which of them are significant at the semantic level (possibly hierarchically) in order to be able to structure the document [2].

The transition effect definition is not always straightforward and may depend upon the target applications. For instance, it has been decided in our case that cuts, even obvious, appearing inside "visual jingles" and stroboscopic effects should not be counted as actual cuts. All effects are counted only if they correspond to a transition for the whole image. Superimposed text, small images and logos appearance and disappearance are not counted as transition effects.

Many automated tools for the temporal segmentation of video streams have been already proposed. It is possible to find some papers that are providing state of the art of such methods [3] [4] [5]. In this paper, we describe the temporal video segmentation system used by CLIPS-IMAG to perform the Shot Boundary Detection (SBD) task of the Video track of the TREC-10 conference. This system was first developed at the LIMSI-CNRS laboratory and was then improved at the CLIPS-IMAG laboratory. It 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. This system also has a special module for detecting photographic flashes and filtering them as erroneous "cuts". It is globally organized according to a (software) dataflow approach and Figure 1 shows its architecture.

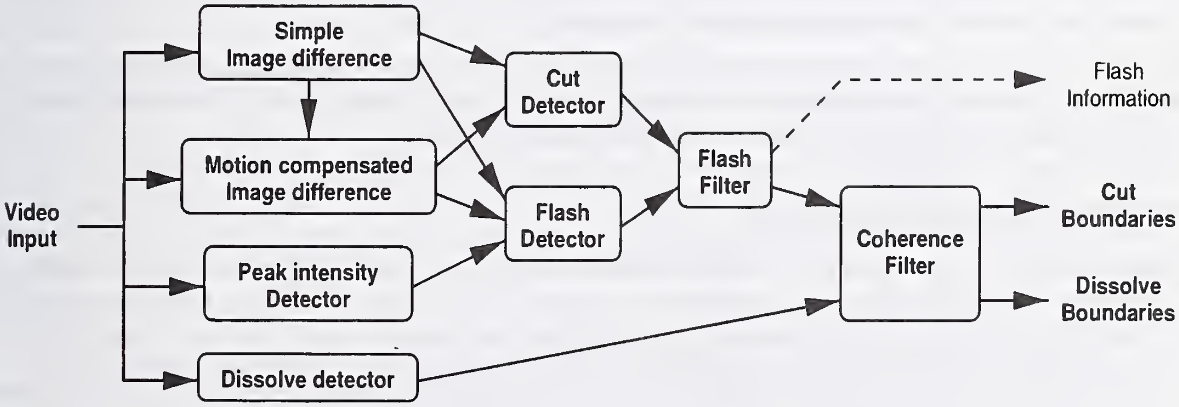


Figure 1: System architecture

The original version of this system was evaluated using the INA corpus and the standard protocol [6] (`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 SBD task partly reused this test protocol (with a different test corpus).

2 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 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 [7] 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.

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.

2.2 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.

3 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 (± 3 frames) of each frame and a filter is then applied (the effect must be detected or almost detected in several consecutive frames).

4 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.

The system is designed for having the capability to evolve by including within its dataflow architecture new feature detection modules and new decision modules. It may also output other data than segmentation information like detection of photographic flashes or other features.

5 Evaluation using the TREC test data

The results for two variants of the CLIPS system were submitted for the TREC SBD task. These variants differ only in the value of some control (threshold) parameters. They are laleled "CLIPS-1" and "CLIPS-2" (CL-1 and CL-2 in the tables) respectively. The first one corresponds to the original parameters of the system (which was tuned for French TV news segmentation). The second one was set with lower thresholds in order to try a configuration with a higher recall, possibly resulting also into a lower precision. The threshold parameters were changed only for the cut detection part. None of the threshold parameters were tuned using the part of the TREC-10 corpus, neither with the data used for system evaluation, nor with the data unused for system evaluation. Shot boundary detection was performed on all of the test data specified for the TREC-10 SBD task in from 5 to 10 times real time (using a Pentium III @ 800 MHz), depending upon the documents' content.

The results are presented on the basis of the final version of reference data and comparison software which give slightly different results from the draft version. Only the system which provided results for the whole test set are compared with our system. These include two systems from Fudan University, China (FU-1 and FU-2), two systems from IBM Almaden Research Center, USA (IBM-1 and IBM-2), one system from Imperial College - London, UK (ICKM), one system from Microsoft Research, China (MSSD), one system from Glasgow University, UK (MB_Frequency, MBF), and two systems from University of Amsterdam and TNO, the Netherlands (Media Mill, MM-1 and MM-2). Global deletion and insertion rates, recall and precision on all files (2061 cuts, 1108 gradual, 3169 total transitions for a total of 624267 frames in 42 video documents) are used as a synthetic data and are presented in table 1.

Cuts	**CL-1**	**CL-2**	FU-1	FU-2	IBM-1	IBM-2	ICKM	MBF	MSSD	MM-1	MM-2
Del.	**0.012**	**0.011**	0.030	0.030	0.021	0.035	0.065	0.184	0.072	0.053	0.091
Ins.	**0.105**	**0.293**	0.040	0.040	0.038	0.037	0.112	0.568	0.074	0.358	0.117
Rec.	**0.988**	**0.989**	0.970	0.970	0.979	0.965	0.935	0.816	0.928	0.947	0.909
Pre.	**0.904**	**0.771**	0.961	0.961	0.963	0.963	0.893	0.590	0.926	0.726	0.886
Grad.	**CL-1**	**CL-2**	FU-1	FU-2	IBM-1	IBM-2	ICKM	MBF	MSSD	MM-1	MM-2
Del.	**0.293**	**0.291**	0.379	0.402	0.268	0.230	0.360	0.963	0.306	0.222	0.554
Ins.	**0.566**	**0.565**	0.241	0.214	0.447	0.589	0.433	0.000	0.375	0.388	0.067
Rec.	**0.707**	**0.709**	0.621	0.597	0.732	0.770	0.640	0.037	0.694	0.778	0.446
Pre.	**0.555**	**0.555**	0.720	0.736	0.621	0.566	0.596	1.000	0.649	0.667	0.870
All	**CL-1**	**CL-2**	FU-1	FU-2	IBM-1	IBM-2	ICKM	MBF	MSSD	MM-1	MM-2
Del.	**0.110**	**0.109**	0.152	0.160	0.107	0.103	0.168	0.457	0.154	0.112	0.253
Ins.	**0.266**	**0.388**	0.110	0.101	0.181	0.230	0.224	0.369	0.179	0.368	0.100
Rec.	**0.890**	**0.891**	0.848	0.840	0.893	0.897	0.832	0.543	0.846	0.888	0.747
Pre.	**0.770**	**0.697**	0.885	0.893	0.832	0.796	0.788	0.595	0.825	0.707	0.882

Table 1: Global results for the SBD TREC-10 evaluation, deletion and insertion rates, precision and recall for "cut", gradual and all transitions.

Table 1 results shows that our attempt to increase the recall (or decrease the deletion rate) of the "cut" transitions by reducing the thresholds between the variants CL-1 and CL-2 of our system completely failed while the precision was severely decreased (or the insertion rate severely increased). Also, the ratio between insertions and deletions is of 9:1 for CL-1 and of 27:1 for CL-2, which is highly asymmetrical. The reason is probably that our system CL-1 was already a highly "recall oriented" system designed to minimize the deletion rate while keeping the insertion rate reasonable (this choice was justified by the hypothesis that over-segmentation can be identified and removed in further steps and may not be very penalizing in most applications while, once missed, transitions cannot easily be detected again and their miss may be penalizing for applications). However such a ratio was not expected (CL-1 was tuned for about a 5:1 ratio on French TV news) and neither was the absence of any improvement in the insertion rate (or recall). The results show that the transitions missed by our system cannot be recovered with the approach used whatever the threshold choice. However, for both variants, the deletion rate is very low (about 1 %). It is about twice lower than the one of the following best system (IBM-1 with about 2 %) and four times lower that the average of all systems.

For gradual transitions, our system shows roughly a 3:1 insertions to deletions ratio (table 1). However, our system is designed to detect only "dissolve" gradual transitions and the deletion rate relative to "dissolve" transitions alone might be lower and, therefore, the actual ratio higher. There is little difference between CL-1 and CL-2 systems since there is no change in the thresholds for the dissolve detector. The minor difference comes from indirect effects of differences in cut detection via the output filter. The performance of CL-2 appear to be slightly better for CL-2 but the overall performance is much better for CL-1.

The insertion and deletion rates for all systems appear to be much higher for gradual transitions than for cuts for all systems (a 10:1 ratio typically). Since cuts are only about twice as numerous as gradual transitions, the effect of errors on gradual transitions strongly dominates in the errors for all transitions (table 1).

Systems are hard to compare since the results include two independent measures: insertion and deletion rates (or precision and recall) and systems have an extremely variable insertions to deletions ratio: from 9:1 for CL-1 (excluding CL-2) for CL-1 down to 1:1 for MSSD (for cuts). However, deletion rates should be compared for an equivalent insertion rate or vice versa or, alternatively, they should be compared at a point for which both values are identical or in a pre-defined ratio. None of these performance indexes is significant without considering simultaneously the other independent variable. Currently, the TREC-10 SBD result data does not provide any single synthetic measure allowing to rank the systems.

Ruiloba et al. [6] proposed three different global indexes: the "error rate" which is the sum of insertion and deletion rates, "quality" which is equivalent to a weighted sum of them giving more importance to deletions than to insertions and "correction probability" which has the drawback of giving a lot more importance to deletions than to insertions, weighting them respectively with the total number of frames minus the number of transitions and the number of transitions alone. All of these measures have their bias and none was selected for TREC-10 SBD evaluation. Moreover, the one chosen would have to be known in advance so that the systems can be tuned appropriately (in terms of precision versus recall compromise) to it.

The best solution would have been that results be given for all systems for a wide range of insertions to deletions ratios (by varying internal threshold parameters) in order to produce a sound Recall × Precision curve. This would have allowed a more objective system performance comparison using for instance: precision at a given recall, recall at a given precision, or any of them for a fixed precision to recall ratio (or similar indexes using insertion and deletion rates instead of precision and recall). This was not possible because the test framework did not permit to provide a ranked list of detected transitions and allowed only two system output per institution.

We did, however, run our system with many different parameter sets (by varying only one global parameter, according to which all other vary simultaneously), we evaluate each run with the same software and reference data and were able to draw Recall × Precision and Deletion rate × Insertion rate diagrams which permitted to compare our system to all others. Both the cuts and global transition control parameters were varied here unlike in the two officially submitted runs into which ony the cuts control parameters were varied. Figures 2 and 3 shows on the same diagram the curve obtained by varying the CLIPS-IMAG system parameters and the points corresponding to all other systems. Figures 4 and 5 shows more detail results in the Deletion × Insertion plane.

From these data, it appears that when comparing the CLIPS system to the nine other systems (or the six other systems if we take only the best one for each institution that submitted two runs), it ranks:

- 2nd of ten (respectively 2nd of seven) for cuts,
- 5th of nine (respectively 3rd of six) for gradual transitions,
- 3rd of ten (respectively 2nd of seven) for all transitions,

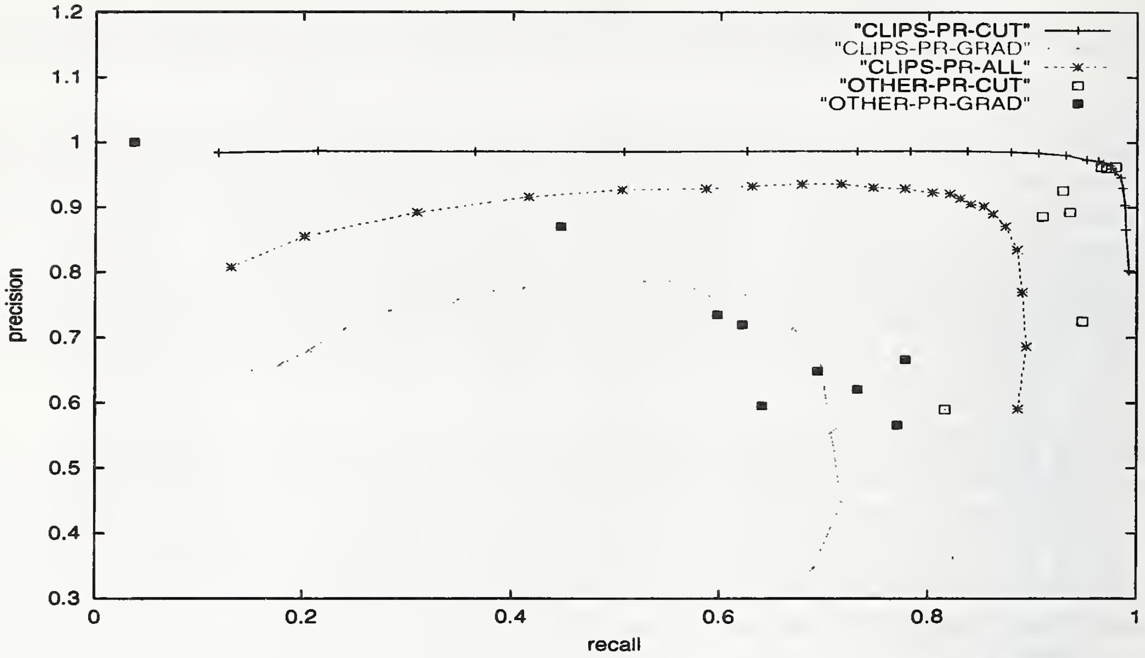


Figure 2: Global results: Recall × Precision

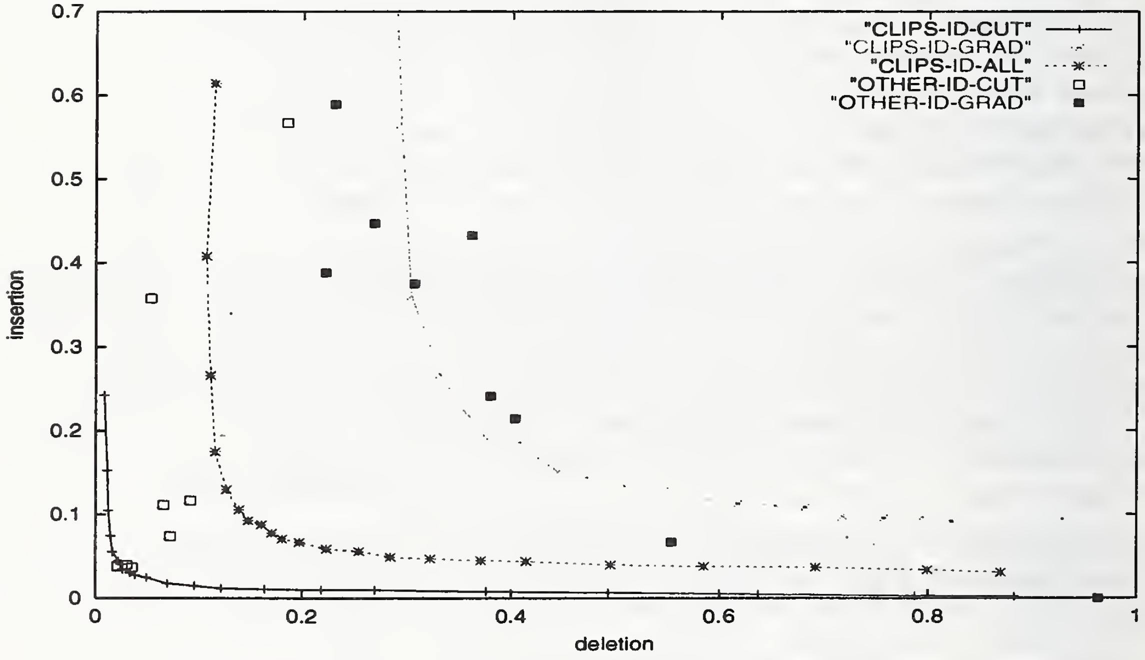


Figure 3: Global results: Deletion × Insertion

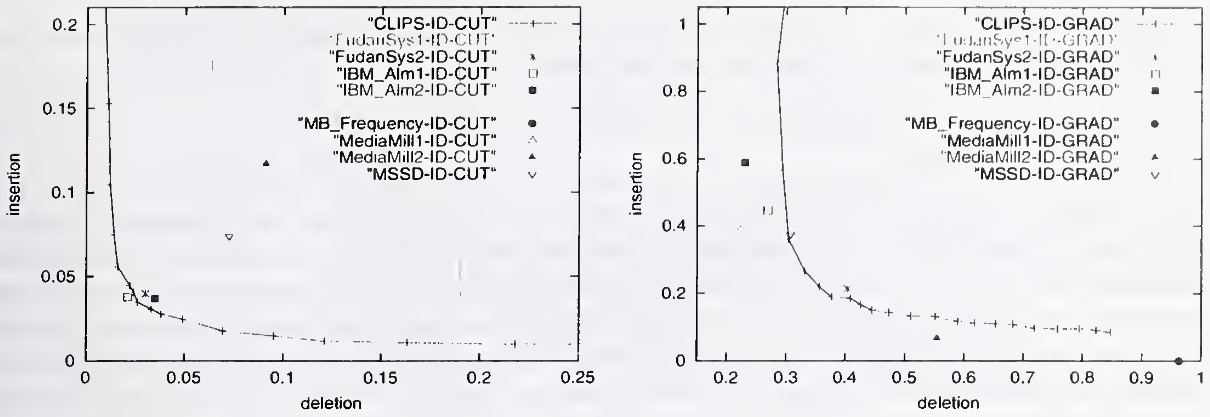


Figure 4: Global results for cuts (left) and gradual transitions (right) : Deletion × Insertion

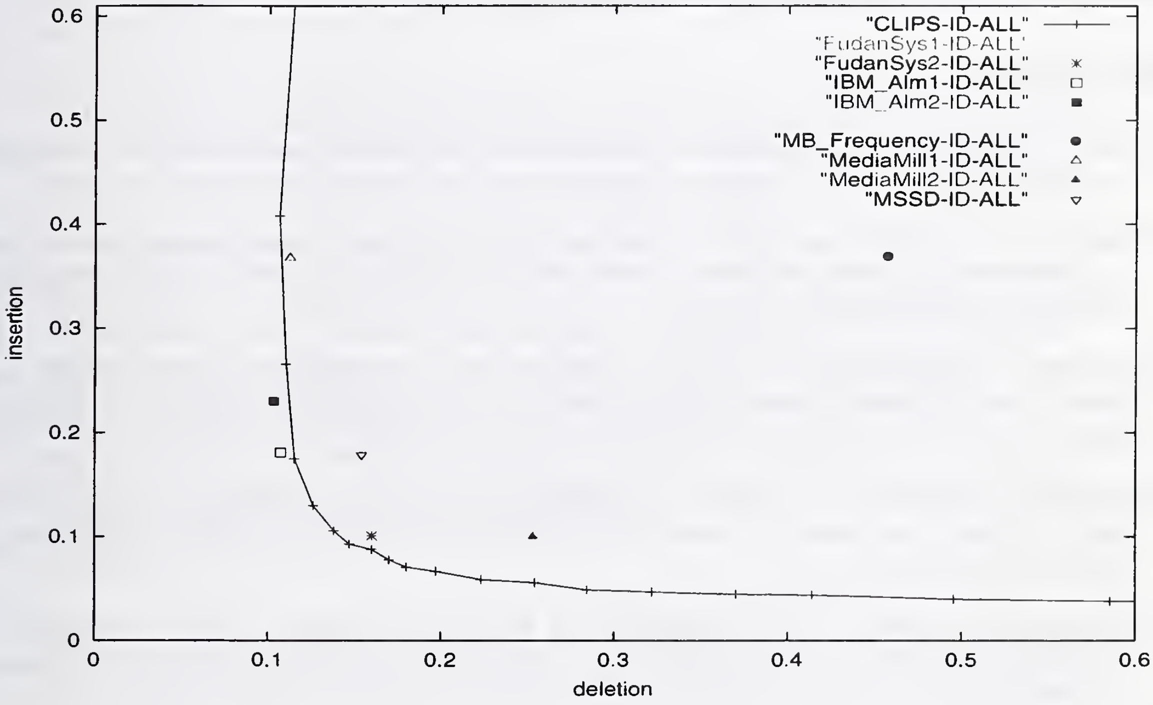


Figure 5: Global results for all transitions: Deletion × Insertion

The Recall × Precision and Deletion × Insertion curves do not appear to be monotonous as they usually are. This is because they are not obtained from a ranked list of detected transition but rather by modifying a set of parameters according to a single global control one. The combined effect of these various parameters, each controlling subsystems that interact with each other for adding or removing transition, explains these unusual results, possibly indicating a non optimal dependence of the several parameters from the global control one. For extreme values of the global control parameter, there is a loss both on recall and precision simultaneously, possibly indicating also unrealistic conditions of operation for the system.

6 Conclusion

This paper has presented the system used by CLIPS-IMAG to perform the Shot Boundary Detection (SBD) task of the Video track of the TREC-10 conference. It implements cut detection using image difference after motion compensation and dissolve detection by the comparison of the norm over the whole image of the first and second temporal derivatives. It also incorporate an output filter to clean the data of both detector and to merge them into a consistent output, and a special module for detecting photographic flashes and filtering them as erroneous "cuts". Shot boundary detection was performed on all of the test data specified for the task in from 5 to 10 times real time, depending upon the documents content. The CLIPS system appear to perform in a very good way for cut transitions and in the average for gradual transitions.

References

[1] Lin,Y., Kankanhalli, M., Chua, T.-S.,: Temporal multi-resolution analysis for video segmentation, In *ACM Multimedia Conference*, 1999.

[2] Aigrain, P., Joly, P., Longueville, V.,: Medium-Knowledge-Based Macro-Segmentation of Video into Sequences, In *Intelligent Multimedia Information Retrieval, Ed. Mark MAYBURY, AAAI Press, MIT Press*, pp 159–173, 1997

[3] Aigrain, P., Zhang, H.J., Petkovic, D.,: Content-based representation and retrieval of visual media : a state-of-the-art review, In *Multimedia Tools and Applications*, 3(3):179-202, November 1996

[4] Lupatini, G., Saraceno, C., Leonardi, R.: Scene break detection: a comparison, In *Proc. VIIIth Intern. Workshop on Research Issues in Data Engineering: Continuous-Media Databases and Applications*, 34-41, Feb. 98.

[5] Boreczky, J. S., Rowe, L. A.: Comparison of video shot boundary detection technique. In *IS&T/SPIE Conference on Electronic Imaging Technology and Science*, San Jose, USA, February 1996.

[6] 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.

[7] 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.

TREC10 Web and Interactive Tracks at CSIRO

Nick Craswell   *David Hawking*   *Ross Wilkinson*   *Mingfang Wu*

Technologies for Electronic Documents
Division of Mathematical and Information Sciences
CSIRO, Australia
{Nick.Craswell; David.Hawking; Ross.Wilkinson; Mingfang.Wu}@csiro.au

1. Overview

For the 2001 round of TREC, the TED group of CSIRO participated and completed runs in two tracks: web and interactive.

Our primary goals in the Web track participation were two-fold: A) to confirm our earlier finding [1] that anchor text is useful in a homepage finding task, and B) to provide an interactive search engine style interface to searching the WT10g data. In addition, three title-only runs were submitted, comparing two different implementations of stemming to unstemmed processing of the raw query. None of these runs used pseudo relevance feedback.

In the interactive track, our investigation was focused on whether there exists any correlation between delivery (searching/presentation) mechanisms and searching tasks. Our experiment involved three delivery mechanisms and two types of searching tasks. The three delivery mechanisms are: a ranked list interface, a clustering interface, and an integrated interface with ranked list, clustering structure, and Expert Links. The two searching tasks are searching for an individual document and searching for a set of documents. Our experiment result shows that subjects usually used only one delivery mechanism regardless of the searching task. No delivery mechanism was found to be superior for any particular task, the only difference was the time used to complete a search, that favored the ranked list interface.

2. Web Track

For the topic relevance runs, an index was built in a similar way to TREC-9. The PADRE99 retrieval system was used. Stemming and stopword elimination were not applied and sequences of letters and or digits were considered as indexable words. Words occurring in titles, metadata fields (if any), URL strings and in referring anchor text were distinguished in the index both from each other and from the normal document text. In order to keep the (compressed) index file under the 2gB filesize limit, only the first 3500 words in each document were indexed. Indexing took just under 2 hours elapsed time on a Dell Latitude C600 laptop with an 850 MHz processor and 512MB of RAM.

When processing queries, the Okapi BM25 relevance function used in TREC-9 was employed. Query-time stemming was employed in title-only automatic runs csiro0awa1 amd csiro0awa3 but not in csiro0awa2.

Performance of the search-engine style interactive interface to WT10g searching was quite acceptable despite the computation involved in extracting documents for display. Hawking and

Craswell took half of the topics each and interacted with the interface for an average of 5-6 minutes per topic while attempting to develop a set of good queries. Each query was saved for later batch processing and submission.

On the homepage finding task, run csiro0awh1 used exactly the same index and query processing machinery as did run csiro0awa2 in the topic relevance task. Results in the homepage finding task were much better than in the topic relevance task.

Homepage finding run csiro0awh2 corresponded to the machinery used in runs described in [1]. A perl script was used to extract anchor text from all the WT10g documents and collect it in pseudo documents named after the target URL. Later, pseudo documents corresponding to documents outside WT10g were removed and a PADRE99 index was built. The unstemmed queries were then processed against this unstemmed index using straightforward Okapi BM25 scoring(taking no account of URLs, or document structure.)

3. Interactive Track

3.1. Introduction

People search information for various purposes/tasks, and information retrieval systems are targeting their searching technologies to specific tasks. For example, some search mechanisms are good at content finding, and some others are good at homepage finding or online service finding[2,3]. An interesting question is whether a user can recognize the special merits of a search mechanism and take advantage of them for his/her searching tasks accordingly.

In this experiment, we conducted a user study in an attempt to gain a better understanding of users' mental model of searching mechanisms and users' searching tasks. Particularly, we investigated the following three research questions:

- If a user has a set of available searching mechanisms and a set of searching tasks, would the user be able to select a suitable mechanism that is optimal for that searching task?
- If a user has a set of available searching mechanisms and is aware of the advantages of each mechanism for certain types of searching tasks, would the user select the one that is optimal for that searching task?
- Can we improve a user's searching performance by guiding the user to use a suitable searching mechanism for a specific task?

3.2. Experimental setting

3.2.1. Topic

We selected eight searching topics from the TREC-10 interactive track. The eight search topics are:

1. Tell me three categories of people who should or should not get a flu shot and why.
2. Find a website likely to contain reliable information on the effect of second-hand smoke.
3. Find three articles that a high school student could use in writing a report on the Titanic.
4. Find three different information resources that may be useful to a high school student in writing a biography of Sr. Ernest Shackleton.
5. Find a website where I can find material on global warming.
6. I want to visit Antarctica. Find a website with information on organized tours/trips there.

7. Identify three interesting places to visit in Perth.
8. Find two websites that will let me buy a teddy bear online.

The above eight searching topics are of two types:

- Type I topic: Searching for a single document (- the information need can be satisfied by a single web document), the Topics 1, 2, 5 and 6 are of this type.
- Type II topic: Searching for a collection of documents (- the information need can be satisfied by a set of web documents), the Topics 3, 4, 7 and 8 are of this type. (It turns out that Topic 7 can also be satisfied by a document.)

3.2.2. Searching mechanism

In this experiment, we used Teoma (http://www.teoma.com) search engine for backend information retrieving. We chose it because the three types of search results returned from Teoma meet our requirements on two selected types of tasks.

Teoma provides the following three searching mechanisms:

- **Web page search (ranked list)**
  This searching mechanism is similar to other web search engines, which return the retrieved documents as a ranked list.

- **Experts' links**
  When a user wants to collect information about a certain topic, the user may not be the only one in this world who is interested in this topic. Very likely, someone else may already build his/her own portal for the topic and make that information available on the internet. If the user can get this portal directly, he/she will save time by avoiding searching/selecting the information piece by piece as from the ranked list.

- **Web pages by topic (clusters)**
  With this mechanism, the top ranked documents are grouped into topic/theme related clusters based on their topic keywords. For example, if a user wants to collect information on "global warming", the top retrieved documents will be clustered dynamically into the categories like "Institute", "Science", "Climate Change, Warming Climate" etc. The user can either drill down to a cluster to get information about a certain topic in depth, or browse a few clusters to collect information in width. This clustering structure can guide the user to collect the needed information purposely and avoid selecting/viewing the duplicated documents.

Intuitively, we think the web page search mechanism is suitable for the single document finding task, while Experts' links and clustering mechanisms are suitable for the information collection task.

Our experiment focused on users' mental model of their searching tasks and assigned searching/presentation mechanisms, instead of on the query formulation/reformulation, we chose to let subjects to perform their searching tasks by using predetermined (canned) queries. Therefore all subjects of the same searching topic got the same set of retrieved documents. We expected this would reduce the effect of query variation.

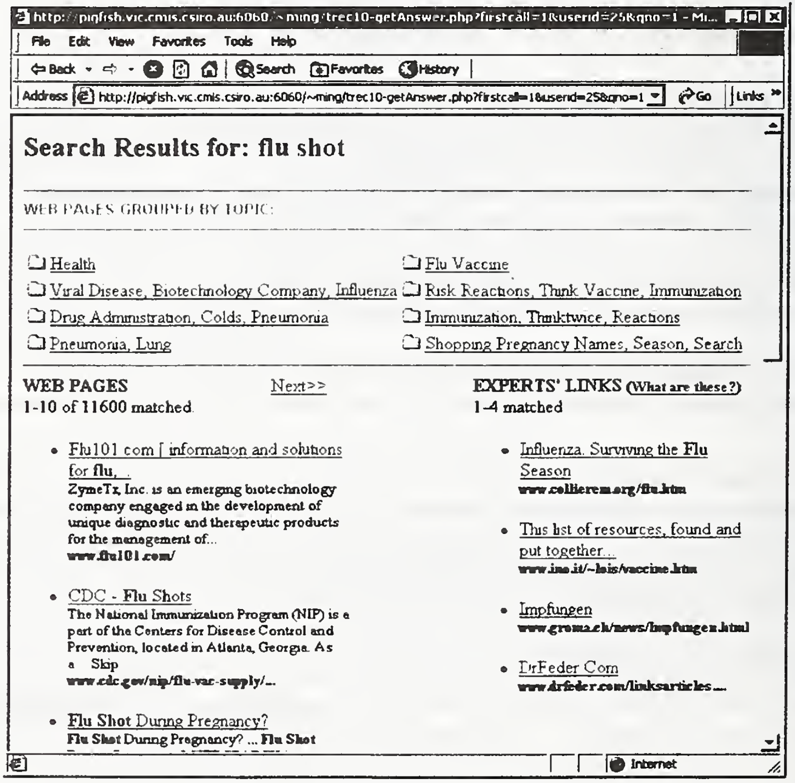
3.2.3. Experimental design

We recruited 24 subjects and divided them evenly into three groups. The experimental designs for each group are as following:

- Group 1
  Subjects were told only about the characteristics of each searching mechanism (as introduced in Teoma's help page). The aim was to observe whether subjects can recognize their task difference and choose a suitable searching mechanism accordingly.
- Group 2
  Subjects were told about the advantages of each searching mechanism and how they are related to the type of tasks, but subjects were still free to choose any mechanism for the search. The aim was to observe whether subjects in this group would select a suitable searching mechanism for a specific task when they clearly recognize the difference in searching topics and searching mechanisms.

In Groups 1 & 2, each searching topic is rotated in each searching position. The interface is similar to that from Teoma (see Figure 1). We developed an interface on the top of Teoma to keep just three necessary searching mechanisms, so that subjects would not know which search engine we are using and therefore concentrate on these three testing mechanisms.

- Group 3
  For this group of subjects, we developed two interfaces: one interface for supporting the web page search only (see Figure 2), and the other for supporting the clustering only (see Figure 3). The searching topics were blocked according to their types. The Latin-square experiment design was used here – each subject used two interfaces to search a block of four topics according to a predetermined order. With this experimental design, we try to compare whether a subject's searching performance would be improved by using a suitable interface (that we think) for that task.


http://pigfish.vic.cmis.csiro.au:6060/~ming/trec10-getAnswer.php?firstcall=1&userid=25&qno=1 - Mi...
File
Edit
View
Favorites
Tools
Help
Back
Search
Favorites
History
Address
http://pigfish.vic.cmis.csiro.au:6060/~ming/trec10-getAnswer.php?firstcall=1&userid=25&qno=1
Go
Links
Internet

Search Results for: flu shot

WEB PAGES GROUPED BY TOPIC:

Health
Viral Disease, Biotechnology Company, Influenza
Drug Administration, Colds, Pneumonia
Pneumonia, Lung
Flu Vaccine
Risk Reactions, Think Vaccine, Immunization
Immunization, Thinktwice, Reactions
Shopping Pregnancy Names, Season, Search

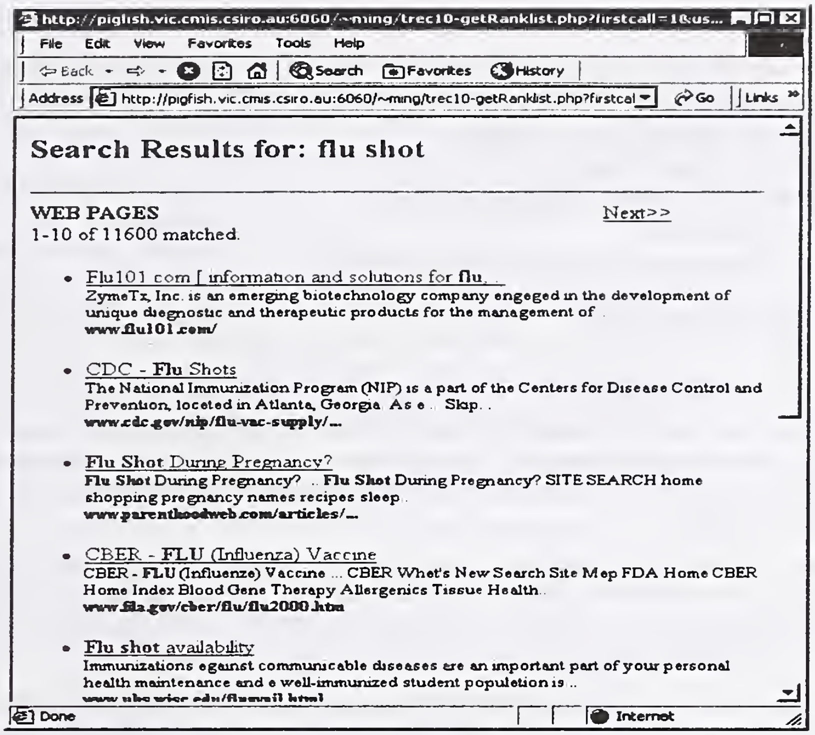
WEB PAGES Next>>
1-10 of 11600 matched

- Flu101 com [information and solutions for flu, .
  ZymeTx, Inc. is an emerging biotechnology company engaged in the development of unique diagnostic and therapeutic products for the management of...
  www.flu101.com/
- CDC - Flu Shots
  The National Immunization Program (NIP) is a part of the Centers for Disease Control and Prevention, located in Atlanta, Georgia. As a Skip
  www.cdc.gov/nip/flu-vac-supply/...
- Flu Shot During Pregnancy?
  Flu Shot During Pregnancy? ... Flu Shot

EXPERTS' LINKS (What are these?)
1-4 matched

- Influenza. Surviving the Flu Season
  www.collierem.org/flu.htm
- This list of resources, found and put together...
  www.ino.it/~lois/vaccine.htm
- Impfungen
  www.groma.ch/news/impfungen.html
- DrFeder Com
  www.drfeder.com/linksarticles....

Figure 1. The integrated interface


http://pigfish.vic.cmis.csiro.au:6060/~ming/trec10-getRanklist.php?firstcall=1&us...
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://pigfish.vic.cmis.csiro.au:6060/~ming/trec10-getRanklist.php?firstcal
Go Links
Done
Internet

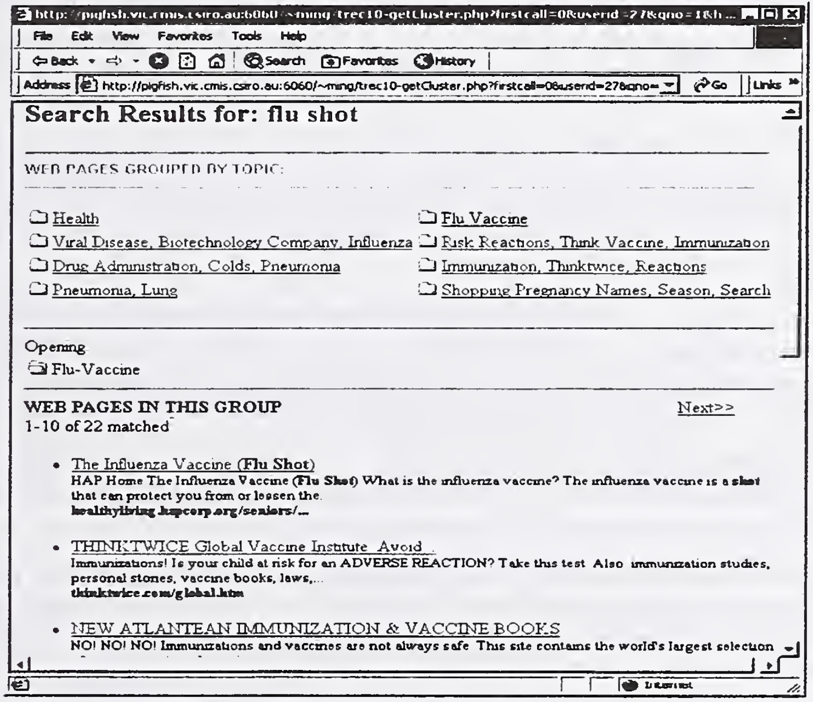
Search Results for: flu shot

WEB PAGES Next>>

1-10 of 11600 matched.

- Flu101.com [information and solutions for **flu**...
  ZymeTx, Inc. is an emerging biotechnology company engeged in the development of unique diagnostic and therapeutic products for the management of ...
  www.flu101.com/
- CDC - **Flu** Shots
  The National Immunization Program (NIP) is a part of the Centers for Disease Control and Prevention, located in Atlanta, Georgia. As a ... Skip...
  www.cdc.gov/nip/flu-vac-supply/...
- **Flu Shot** During Pregnancy?
  Flu Shot During Pregnancy? ... **Flu Shot** During Pregnancy? SITE SEARCH home shopping pregnancy names recipes sleep...
  www.parenthoodweb.com/articles/...
- CBER - **FLU** (Influenza) Vaccine
  CBER - **FLU** (Influenze) Vaccine ... CBER What's New Search Site Map FDA Home CBER Home Index Blood Gene Therapy Allergenics Tissue Health...
  www.fda.gov/cber/flu/flu2000.htm
- **Flu shot** availability
  Immunizations against communicable diseases are an important part of your personal health maintenance and a well-immunized student population is ...

Figure 2. The ranked list interface


http://pigfish.vic.cmis.csiro.au:6060/~ming/trec10-getCluster.php?firstcall=0&userid=27&qno=1&h...
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://pigfish.vic.cmis.csiro.au:6060/~ming/trec10-getCluster.php?firstcall=0&userid=27&qno=
Go Links
Internet

Search Results for: flu shot

WEB PAGES GROUPED BY TOPIC:

Health	Flu Vaccine
Viral Disease, Biotechnology Company, Influenza	Risk Reactions, Think Vaccine, Immunization
Drug Administration, Colds, Pneumonia	Immunization, Thinktwice, Reactions
Pneumonia, Lung	Shopping Pregnancy Names, Season, Search

Opening
Flu-Vaccine

WEB PAGES IN THIS GROUP Next>>

1-10 of 22 matched

- The Influenza Vaccine (**Flu Shot**)
  HAP Home The Influenza Vaccine (**Flu Shot**) What is the influenza vaccine? The influenza vaccine is a **shot** that can protect you from or lessen the...
  healthyliving.hapcorp.org/seniors/...
- THINKTWICE Global Vaccine Institute. Avoid ...
  Immunizations! Is your child at risk for an ADVERSE REACTION? Take this test. Also: immunization studies, personal stories, vaccine books, laws,...
  thinktwice.com/global.htm
- NEW ATLANTEAN IMMUNIZATION & VACCINE BOOKS
  NO! NO! NO! Immunizations and vaccines are not always safe. This site contains the world's largest selection

Figure 3: The clustering Interface

3.2.4. Experimental procedure

All experiments followed the following procedure:

1. Pre-search questionnaire
2. Training session
3. A search session (maximal 8 min)
4. Post-search questionnaire
   Repeat 3 & 4 until all topics are searched.
5. Exit questionnaire
   (The whole procedure took about 1.5 hours.)

3.3. Experimental result

3.3.1. Search Performance

All subjects successfully searched all topics within assigned time period. It seems that this year's searching topics are relatively easy. For each topic, the needed information can usually be found from the top 5 retrieved documents, though more within-a-site browsing/searching is needed for Topics 6, 7, and 8.

We can see from Table 1 that there is no significant difference between each group in terms the number of document read, either within the same type of topics or across all eight topics. (The mean across all eight topics for each group is: M(Group1) = 4.1, M(Group2) = 4.1, M(Group3-Clus) = 3.9, and M(Group3-List) = 4.1.)

Table 1. The number of documents read by each group

		Type I topic					Type II topic				
Topic		1	2	5	6	Mean	3	4	7	8	Mean
Group1		5.5	3	3	4	3.9	5.4	4.6	3.4	4.4	4.5
Group2		5.8	3.0	2.2	3.6	3.7	4.9	4.7	3.9	4.8	4.6
Group3	Clus	5.9	3.0	3.0	3.3	3.8	4.8	4.7	3.1	3.7	4.1
	List	5.9	3.3	2.3	3.3	3.7	4.9	4.7	4.0	4.3	4.5

Table 2 shows the time taken to finish each search session. Overall, subjects from Group3 used the least time by using the ranked list interface, this is probably because subjects of this interface got less distraction, they concentrated on one interface, and the quality of this list is very high. In this table, significant difference only exists between the Group1 and the list interface of the Group3.

Table 2. Time to finish each search session (in second)

		Type I topic					Type II topic				
Topic		1	2	5	6	Mean	3	4	7	8	Mean
Group1		230.3	184.9	119.6	232.4	191.8	303.1	306.0	146.5	356.1	277.9
Group2		131.4	132.5	116.9	230.5	152.8	315.6	228.6	243.3	208.0	248.9
Group3	Clus	248.0	162.5	154.5	196.0	190.1	349.8	253.0	192.3	271.5	266.7
	List	93.0	125.0	91.3	178.5	122.0	245.0	248.3	125.5	334.3	238.3

To answer our research question 3, we compare subject's performance with the Group 3 by using either clustering interface or the ranked list interface, we do not see any evidence to show that the subjects of the clustering interface would finish their searching sessions faster than other interfaces, and take less effort.

3.3.2. Search Behavior

During the experiment, we observed the following searching behaviors:

- There were generally two browsing strategies used by Group1 and Group2 subjects. One type of subjects read the search result page from top to bottom, and picked up a possibly relevant document to read along the way. Another type of subjects read and browsed the search result first, then selected the (possibly most) relevant documents to read. The distribution of each type is relatively even (52 : 76)[1]. This may imply that to the first type of subjects, the ranking is important, while to the second type of subjects, the site summary is more important.
- To answer our research question 1 and 2, we went through the recorded screen actions of subjects from Group1 and Group2. Generally, subjects of Group1 and Group2 checked the ranked list first, when they could not find satisfactory document(s), then they would switch to clusters or expert links. Only for a small number of sessions (17 in Group 1 vs. 24 in Group 2) from each group, a searching session was started by using clustering or expert links. Subjects from Group 2 used more clustering organization and expert links (63 sessions in Group 2 vs. 47 sessions in Group 1). This may indicate that if the subjects understand more about the clustering organization and expert links, they would use these two mechanisms more. A further experiment with more subjects is needed to verify this claim.
- The interaction with the searching system is mixed up with the usability of a website. For example, for tourism and shopping topics (topics 6, 7 and 8), the needed information is usually not brought on the first page. A within-a-site browsing is needed. The searching success depends on the design of the site, subject's searching habit and luck. For example, for Topic 8, ten subjects (from Group 1 & 2) read the first ranked document, only half of them found the needed information; the quickest one took 40 seconds while the slowest one took 3 minutes. For another example, the first ranked document of the topic 6 has nearly 10 screens of text, the first screen has a lot of links for browsing, and the needed information is on the fourth screen. If subjects scroll a lot and read the whole text (or up to the fourth screen) of the page, they would be able to find the needed information easily. However some subjects just read the first screen, they then followed whatever links on the first screen that they were interested in, as a result, they usually got lost within this site.
- In our post-experimental questionnaire/interview, we asked subjects to describe their daily searching habits. Generally, subjects recognized that they search for various purposes, but they usually stick to one search engine. They switch to other search engines only when they can not get satisfactory results. Only two subjects claimed that they choose search engines depending on what they are searching. For example, they would select Google to search someone's homepage, NorthernLight for research documents, and Yahoo for shopping stuff. (It is not clear if an interface is designed to encourage users to switch from one service to another, will more users do so?)

3.3.3. Subjects' feedback

Subjects from Group 1 and Group 2 gave similar comments on each mechanism. Here are some typical examples what they like about each mechanisms and what they dislike each mechanisms.

Pros:

Cluster

- Make it easier to find information related to the search topic

[1] Group 1 and Group 2 together had 128 searching sessions (8 topics x 16 subjects) in total.

- Easy to find specific information
- If you can find a useful topic, then you get a good list of useful relevant sites/links
- When the groups are accurate they are very useful for finding multiple sites with similar content.
- It gave me the opportunity to learn more about the topic.

Web pages

- Highlight the match words
- More detail description of each link, so it helps to search quickly.
- Perhaps contains the "most relevant sites, but also contains many irrelevant sites, but if you can sift them out, you usually get good results.

Expert Links

- Credible, often lots of links
- Have a page with many good, specific links

Cons:

Cluster

- Can be hard to find a completely relevant topic

Web pages

- Sometimes requires experience to be able to look at the briefs/summaries and quickly find the useful relevant ones
- Many pages, have to scroll more.

Expert links

- Show only the address, it is better to show a short description of the page.
- Often disorganized, and occasionally just too much information, overwhelming
- Sometimes too specific, not general enough

Most subjects said the searching topics are close to their daily search but these selected topics are relatively easy. One subject commented: "Overall, most searches were fairly easy and quick."; and another subject wrote: "I hope I didn't finish it too quickly! It might seem like I was rushing, but it was just that I found results quickly."

3.4. Conclusion

Based on the experimental results, we can't conclude that users would select a particular mechanism for a certain type of search tasks, neither we can assume that user's search behavior is task driven at this stage. Further analysis and research are needed.

4. References

1. Nick Craswell, David Hawking and Stephen Roberson. Effctive site finding using link anchor information. Proceedings of ACM SIGIR 2001, September 2001, pp.250-257

2. David Hawking, Nick Craswell and Kathleen Griffiths. *Which search engine is best at finding online services?*. WWW-10 Poster Proceedings, 2001.

3. Nick Craswell, David Hawking and Kathleen Griffiths. *Which search engine is best at finding airline site home pages?*. CSIRO Mathematical and Information Sciences TR01/45, 2001.

Lazy Users and Automatic Video Retrieval Tools in (the) Lowlands

The Lowlands Team
CWI[1], TNO[2], University of Amsterdam[3], University of Twente[4]
The Netherlands

This work was funded (in part) by the ICES/KIS MIA project and the Dutch Telematics Institute project DRUID. The following people have contributed to these results (appearing in alphabetical order): Jan Baan[2], Alex van Ballegooij[1], Jan Mark Geusenbroek[3], Jurgen den Hartog[2], Djoerd Hiemstra[4], Johan List[1], Thijs Westerveld[4], Ioannis Patras[3], Stephan Raaijmakers[2], Cees Snoek[3], Leon Todoran[3], Jeroen Vendrig[3], Arjen P. de Vries[1] and Marcel Worring[3].

1 Introduction

This paper describes our participation in the TREC Video Retrieval evaluation. Our approach uses two complementary automatic approaches (the first based on visual content, the other on transcripts), to be refined in an interactive setting. The experiments focused on revealing relationships between (1) different modalities, (2) the amount of human processing, and (3) the quality of the results.

We submitted five runs, summarized in Table 1. Run 1 is based on the query text and the visual content of the video. The query text is analyzed to choose the best detectors, e.g. for faces, names, specific camera techniques, dialogs, or natural scenes. Query by example based on detector specific features (e.g. number of faces, invariant color histograms) yields the final ranking result.

To assess the additional value of speech content, we experimented with a transcript generated using speech recognition (made available by CMU). We queried the transcribed collection with the topic text combined with the transcripts of video examples. Despite of the error-prone recognition process, the transcripts often provide useful information about the video scenes. Run 2 combines the ranked output of the speech transcripts with (visual-only) run 1 in an attempt to improve its results; run 3 is the obligatory transcript-only run.

Run	Description
1	Detector-based, automatic
2	Combined 1–3, automatic
3	Transcript-based, automatic
4	Query articulation, interactive
5	Combined 1–4, interactive, by a lazy user

Table 1: Summary of runs

Run 4 models a user working with the output of an automatic visual run, choosing the best answer-set from a number of options, or attempting to improve its quality by helping the system; for example, finding moon-landers by entering knowledge that the sky on the moon is black or locating the Starwars scene by pointing out that the robot has golden skin.

Finally, run 5 combines all information available in our system: from detectors, to speech transcript, to the human-in-the-loop. Depending on the evaluation measures used, this leads to slightly better or slightly worse results than using these methods in isolation, caused by laziness expressed in the model for selecting the combination strategy.

2 Detector-based Processing

The main research question addressed in run 1 was how to make query processing fully automatic. This includes devising mechanisms that bridge in an automatic way the semantic gap [13] between (1) the user's information need as specified on the one hand by the topic text description and ou the other hand by the video and image examples and (2) the low level features that can be extracted from the video. We propose a unifying approach in which a wide range of detectors and features are combined in a way that is specified by semantic analysis of the topic description. Section 2.1 describes the system's architecture and Section 2.2 the specific detectors and features used.

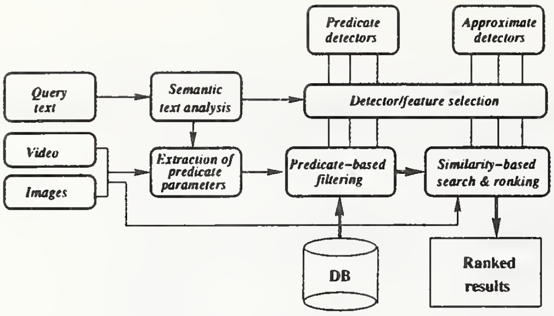

Predicate detectors
Approximate detectors
Query text
Semantic text analysis
Detector/feature selection
Video
Images
Extraction of predicate parameters
Predicate-based filtering
Similarity-based search & ranking
DB
Ranked results

Figure 1: Architecture for automatic system.

2.1 System's architecture

A great challenge in automatic retrieval of multimedia material is to determine which aspect of the information carried in the audiovisual stream is relevant for the topic in question. The aspects of information that we restrict to are determined by the specific detectors that our systems employs. Examples are color-based detectors, face detectors or modules that detect the camera technique or the presence of monologues.

In order to select the relevant detectors we associate them with concepts that exist in the 'is-a' hierarchy of the Wordnet dictionary. For example, the face detectors are associated with the concept *'person, individual, human'*. In order to determine if the specific detector is to be used for a topic, we analyze its text[1] in two steps. In the first step, a syntactic analysis discards the words that are not nouns, verbs or adjectives. In the second step, we feed the remaining words to the Wordnet dictionary and detect if the concepts that are associated with the detectors that we have at our disposal are present in the 'is-a' hierarchy of the most common meaning of the words in question. Such an approach makes good associations for most of our detectors. However, it exhibits its limitations in the case that the query word has also other meanings. For example the most common meaning of the word "pan" is *cooking utensil, cookware"*. Such ambiguities are resolved in our current system by maintaining an additional set of keywords for the camera motion detector.

Once the appropriate set of detectors are selected we proceed to the retrieval of the relevant video clips. In order to do so we need to make a distinction between two different kinds of detectors[13]:

- detectors for *exact* queries that yield a yes/no answer depending if a set of predicates is satisfied (e.g. does the camera exhibit a zoom-in?). The face detector, the monologue detector, and the camera technique detector fall in this category;
- detectors for *approximate* queries that yield a measure that expresses how similar is the examined video clip with an example video clip. In this category fall the module for color-based retrieval.

The selected detectors of the first category are used to filter-out irrelevant material. Then, a query-by-example based search on the (selected) detectors of the second category produces the final ranked results. In case that the analysis of the topic description determines that no detector of the second category should be selected, the ranking is based on the shot length.

Let us finally note that some of the detectors of the first category learn some of their parameters from the examples provided in the topic. Such a detector is the face detector which learns from the query example how many persons should appear in a video clip so that it is characterized as relevant.

2.2 Detectors

Another goal in the evaluation was to assess the quality of the detectors discussed in this Section. The results of run 1, in the cases that the right detector was chosen, indicate the techniques perform with fairly high precision.

2.2.1 Camera technique detection

To detect the camera technique used in a shot, we use a method based on spatiotemporal slices of the original video to detect whether the apparent motion is due to known camera activities such as pan and tilt, or the scene is static [9]. In the former case, we estimate the percentage of the apparent motion that is due to camera's pan, tilt and zoom (e.g. 60% zoom, 5% tilt and 35% pan). Clips to which the dominant apparent motion is not caused by camera operations are characterized as "unknown".

The detector of the camera technique was used for topics 44, 48 and 74 in which the keywords 'zoom' and 'pan' appear. The system categorized successfully apparent motions that are due to pure camera operations (90% precision for topic 44 and 100% precision for query 74), but failed for topic 48 in which the zooming-in is not due to change in camera's focal-length. The reason for the latter is that the apparent motion field depends on the distance between camera and scene.

[1] We analyzed only the first sentence of the topic description.

2.2.2 Face detector

An off-the-shelf face detector (Rowley[12]) is used in order to detect how many faces are present in the video clip in question. The result is compared with the number of faces that were detected in the image example. We use five categories of numbers of faces: 'no-face", '1-face', '2-faces', '3-faces', 'many-faces'. The face detector is associated with the general concepts *"person, individual, human"* and *"people"* for the Wordnet hierarchy. It works well for topics requesting humans appearing in (near) frontal view (e.g. 100% precision for topic 41) but, naturally, is not relevant otherwise (e.g. water-skier in topic 31).

2.2.3 Caption retrieval

For finding given names in the visual content, three steps are taken:

- text segmentation;
- OCR;
- fuzzy string matching.

For text segmentation of video frames we use a dual approach. The first approach is a color segmentation method [20], to reduce the number of colors, while preserving the characters. The second approach is intensity based, using the fact captions are superimposed. OCR is done by ScanSoft's TextBridge SDK 4.5 library [16]. Finally, string matching is done using k-differences approximate string matching (see e.g. [1]).

The detector worked well in retrieving video based on the text that appears as caption. It has been applied for 24 topics that contain capitalized text (e.g. 'House' and 'Congress' in topic 30) with around 10% and 20% false positives and false negatives respectively. However, the retrieved video (even if it contained the query text as a caption) did not always match with the user's intention (e.g. the result for topic 30 is a shot of a text document). Therefore, we have used the results of such a detector only when the topic consists of a text description only (i.e. no media example is available). Only in that case the shots that are retrieved based on this detector are used to initiate a color-based query.

2.2.4 Monologue detection

The method for monologue detection [15] first uses a camera distance heuristic based on Rowley's face detector [12]. Only shots showing faces appearing in front of the camera within a certain distance are processed. In a post-processing stage all those shots are checked upon using three constraints:

- shot should contain speech;
- shot should have a static or unknown camera technique;
- shot should have a minimum length.

When all constraints are met, a shot is classified as a monologue. Subsequently, the selected shots are ranked based on their length: the longer the shot the higher the likelihood of it being a true monologue.

This detector has been used for topics 40, 63 and 64 with a very good performance (near 100% precision). The performance is lower for topic 64 (60% precision), because satisfying the information need (*male* interviewees) requires to distinguish between sexes, a predicate not anticipated in our current system.

2.2.5 Detectors based on color invariant features

Ranking of the shots remaining after filtering using predicate detectors, was accomplished by implementing a query by image example paradigm. For each keyframe a robust estimate of the color content of each keyframe is computed by converting the keyframe to the Gaussian color model as described in [4]. The Gaussian color model is robust against spatial compression noise, achieved by the Gaussian smoothing involved. Further, the Gaussian color model is an opponent color representation, for which the channels are largely uncorrelated. Hence, the color histograms can be constructed as three separate one-dimensional histograms. The keyframes were stored in a database, together with their color histogram information. Matching of example keyframe against the database targets is efficiently performed by histogram intersection between each of the three (one-dimensional) histograms. Matching time was within a second, ensuring system response to be adequate for interactive retrieval purposes.

3 Probabilistic Multimedia Retrieval

This section introduces our probabilistic approach to information retrieval, an approach that unifies models of discrete signals (i.e. text) and models of continuous signals (i.e. images) into one common framework. We usually take for text retrieval an approach based on statistical language models [6, 7, 10, 3], which uses a mixture of discrete probability measures. For image retrieval, we experimented with a probabilistic model that uses a mixture of continuous probability measures [18].

The basic model can in principal be used for any type of documents and queries, but for now we assume our documents are shots from a video. In a probabilistic setting, ranking the shots in decreasing order of relevance amounts to ranking the shots by the probability $P(Shot_i|Q)$ given that query. Using Bayes' rule we can rewrite this to:

$$\begin{aligned} P(Shot_i|Q) &= \frac{P(Q|Shot_i)P(Shot_i)}{P(Q)} \\ &\propto P(Q|Shot_i)P(Shot_i) \end{aligned}$$

In the above, the right-hand side will produce the same ranking as the left-hand side. In absence of a query, we assume that each shot is equally likely of being retrieved, i.e. $P(Shot_i)$ = constant. Therefore, in a probabilistic model for video retrieval shots are ranked by their probability of having generated the query. If a query consists of several independent parts (e.g. a textual Qt and visual part Qv), then the probability function can be easily expressed as the joint probability of the different parts. Assuming independence between the textual part and the visual part of the query leads to:

$$P(Q|Shot_i) = P(Qt|Shot_i)P(Qv|Shot_i) \quad (1)$$

3.1 Text retrieval: the use of speech transcripts

For text retrieval, our main concern was adapting our standard language model system to the retrieval of shots. More specifically, we were interested in an approach to information retrieval that explicitly models the familiar 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 modeling complex representations of the data [6]. We propose to rank shots by a probability function that is a linear combination of a simple probability measure of the shot, of its corresponding scene, and of the corresponding video (we ignore frames, because in practice words in transcribed speech are not associated with a particular frame).

Assuming independence between query terms:

$$\begin{aligned} P(Qt_1, \cdots, Qt_n|Shot) = \\ \prod_{j=1}^{n} (\pi_1 P(Qt_j) + \pi_2 P(Qt_j|Video) + \\ \pi_3 P(Qt_j|Scene) + \pi_4 P(Qt_j|Shot)) \end{aligned}$$

In the formula, $Qt_1, \cdots, Qt_n$ is a textual query of length n, $\pi_1, \cdots, \pi_4$ are the probabilities of each representation, and e.g. $P(Qt_j|Shot)$ is the probability of occurrence of the term Qt_j in the shot: if the shot contains 10 terms in total and the query term in question occurs 2 times then this probability would be simply $2/10 = 0.2$. $P(Qt_j)$ is the probability of occurrence of the term Qt_j in the collection.

The main idea behind this approach is that a good shot is one that contains the query terms; one that is part of a scene that has more occurrences of the query terms; and one that is part of a video that has even 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 the shot just before it begins or just after it finishes. 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.

3.2 Image retrieval: retrieving the key frames of shots

For the visual part, we cut the key frames of each shot into blocks of 8 by 8 pixels. On these blocks we perform the Discrete Cosine Transform (DCT), which is used in the JPEG compression standard. We use the first 10 DCT-coefficients from each color channel[2] to describe the block. If an image consists of n blocks, we have n feature vectors describing the image (each vector consisting of 30 DCT coefficients). Now the probability that a particular feature vector (Qv_j) from our query is drawn from a particular shot $(Shot_i)$ can be described by a Gaussian Mixture Model [18]. Each shot in the collection is then described by a mixture of C Gaussians.[3] The probability that the a query (Qv) was drawn from $Shot_i$ is simply the joint probability for all feature vectors from Qv. We assume independence between the feature vectors

$$\begin{aligned} P(Qv_1, \quad \ldots \quad, Qv_n|Shot_i) = \\ \prod_{j=1}^{n} \sum_{c=1}^{C} \pi_{i,c} \mathcal{G}(Qv_j, \mu_{i,c}, \Sigma_{i,c}) \quad (2) \end{aligned}$$

where $\pi_{i,c}$ is the probability of class c from $Shot_i$ and $\mathcal{G}(Qv_j, \mu_{i,c}, \Sigma_{i,c})$ is the Gaussian density (or normal density) for class c from shot i with mean vector μ_i and co-variance matrix Σ_i. If m is the number of

[2] We work in the YCbCr color space.

[3] We used a mixture of 8 Gaussians.

DCT features representing a shot, the Gaussian is defined as:

$$\mathcal{G}(x,\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^m|\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} \qquad (3)$$

For each of the shots in the collection we estimated the probability, mean and co-variance for each of the Gaussians in the model using the Expectation Maximization algorithm [11] on the feature vectors from the shots.

At this stage, equation 2 could be used to rank shots given a query, however, its computational complexity is rather high. Therefore, instead of this Feature Likelihood (the likelihood of drawing all query features from a shot model) we computed the Random Sample Likelihood introduced by Vasconcelos [18]. The Random Sample Likelihood is defined as the likelihood that a random sample from the query model was drawn from the shot model, which comes down to building a model for your query image(s) and comparing that model to the documents models to rank our shots.

3.3 Experimental setup

For the textual descriptions of the video shots, we used speech transcripts kindly provided by Carnegie Mellon University. Words that occurred within a transition between two shots were put within the previous shot. We did not have a division of the video into scenes, nor did we build a scene detector. Instead, scenes were simply defined as overlapping windows of three consecutive shots. Because we did not have material available to tune the model, the values of the parameters were determined on a ad-hoc basis. Instead of implementing the model as described, we took a more straightforward approach of doubling artificially the terms in the middle shots to obtain pseudo-documents, and ranked those using the 'standard' model with parameter $\lambda = 0.15$ (see [6]). For the queries, we took both the words from the textual description of the topics and the words occurring in the video examples' time frame, if these were provided.

Run 2 combines automatically the results of run 1 and run 3. It is produced by applying the ranking strategy determined by query analysis to the results of the speech transcript run, using the latter as a filter; unless query analysis decides the transcripts would be irrelevant. Transcripts are ignored if the video is not expected to contain query words, which is the case of predicate detectors like camera motion techniques and monologues.

Run	R@100	P@100
Text-based (run 3)	0.133	0.007
Detector-based (run 1)	0.101	0.003
Image-based (unofficial)	0.065	0.003
Combined (run 2)	0.085	0.005
Combined (unofficial)	0.079	0.005

Table 2: Recall @ 100 and precision @ 100 for probabilistic runs

The results of run 2 did not improve upon run 3, which may be attributed to the ad-hoc approach of combining methods. This motivated additional experiments with a pure probabilistic approach. We evaluated this alternative on the known item search task in an unofficial run. Table 2 compares these unofficial results with our submitted runs. A returned fragment is regarded relevant if the intersection between the fragment and a known item contains at least one third of the fragment and one third of the known item.

Unfortunately, the unofficial combined run is not better than run 2. The difference between measured performance of the unofficial image-based run and run 1 may have influenced this result. Although it is too early to draw strong conclusions from our experiments, another plausible explanation is that the assumption of independence between the textual and visual part is not a valid one.

4 Interactive Experiments

Our interactive topic set consisted – by mistake – of only 30 topics, of which we 'solved' 9, and could not produce any answer for 2.[4] This Section presents mostly positive highlights of our work on the interactive topics for the Video Collection. Note that our interactive users do not identify the correct answers in the retrieved result sets, so precision is not expected to be 100% (see also Section 5).

A quick investigation of behavior of 'standard' image and video analysis techniques on the interactive topics proved our suspicion that purely automatic systems cannot be expected to perform well on most topics: a result of the 'difficult' queries (not just 'sunset' and 'tropical fish') and the low quality of the video data itself. Thus, we focused on the research question how users could improve upon naive

[4] The slightly smaller topic set used was the result of missing a crucial message on the mailing list.

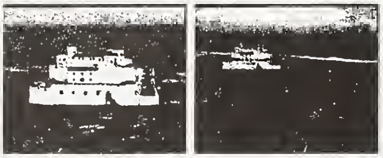


Figure 2: Topic 33, White fort, example(left) and known-item(right) keyframes.

query-by-example methods to express their information needs in a more successful manner.

The retrieval system used for this task is developed on top of Monet, a main-memory database system. It uses a variety of features that are all based on the distribution of color in the keyframes of the shots. Details on the particular features used are provided in a forth-coming technical report [17]. Note that, even though we participated in the interactive topics, the lack of a proper user interface in our current implementation implies that system interaction consisted mostly of writing scripts in Monet's query language.

4.1 Color-based Retrieval Techniques

The results of topics 33 (White fort) and 54 (Glenn Canyon dam) clearly demonstrate that popular color-based retrieval techniques can indeed be successful, *as long as the query example is derived from the same source as the target objects.* Figure 2 shows the keyframes representing the example and known item for topic 33; any color-based technique worked out well for this query. Topic 54 was solved using a spatial color histogram retrieval method, implicitly enforcing locality such as blue sky on top, brown rocks on the sides and white water and concrete dam in the center.[5]

Topic 53 (Perseus) is an example where we were lucky: the example image provided happens to look surprisingly much like the Perseus footage in the data-set, and spatial color histogram retrieval retrieves a large number of Perseus clips.

Topic 24 (R. Lynn Bondurant) provides an interesting lesson about the balance between recall and precision using content-based retrieval techniques. Although it is relatively easy to find some other shots showing Dr. Bondurant – those where he sits in the same room wearing the same suit – finding *all* shots is a completely different question.

The other topics confirm our intuition that we should not expect too much from 'traditional' content-based retrieval techniques. Although more advanced features based on texture and shape possibly could help in solving more topics directly, we doubt whether a *significant* improvement over these results would be achieved. If available however, *domain-specific* detectors (such as the face detectors deployed in run 1) can provide good performance for specific tasks.

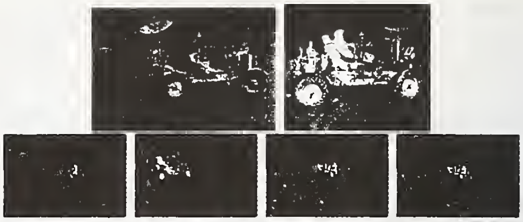


Figure 3: Topic 19, Lunar rover, examples (images on top) and the keyframes of the correct answers.

4.2 Query Articulation

As an alternative approach, we propose to put more emphasis on the quality of the queries expressing the underlying information need. We aim for the interactive refinement from initial, broad multi-modal examples into relatively precise search requests, in a process we have termed *query articulation* [2]. In essence, articulating a query corresponds to constructing a query-specific detector on-the-fly.

The idea of query articulation is best demonstrated through the idea of a 'color-set'. Users define color-sets interactively by selecting regions from the example images, possibly extending the implied color-set by adding similar colors. Unlike the binary sets introduced in VisualSEEK [14], we essentially re-quantize the color space in a smaller number of colors, by collapsing the individual elements of a color-set onto a single new color.

Topic 19: Lunar Rover

Topic 19 (Lunar Rover) provides 2 example images showing the lunar rover. The visual differences between the (grayish) sample images and (bluish) known-items (shown in Figure 3) explain why color-based retrieval techniques are not successful on this topic. Query articulation allows users to circumvent this problem, by making explicit their own world knowledge: in scenes on the moon, the sky is black. This can be expressed in terms of the system using two simple filters based on color-sets:

- 'Black Sky': The filter is realized by selecting those keyframes for which the top 25% of the

[5] Obviously, nothing guaranteed the dams found are indeed Glenn Canyon dams...

Figure 4: The 'dark' color-set as defined for topic 19, Lunar rover.

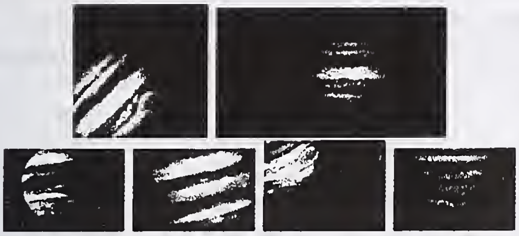


Figure 5: Topic 8, Jupiter, example (on top) and some correct answers keyframes.

image is at least 95% dark (a color-set shown in Figure 4).

- 'Non-black Bottom': making sure that no completely dark images are retrieved, (a large number of outer-space shots are present in the dataset) this second filter selects only those keyframes that do not have a black bottom as there should be lunar surface with the lunar rover visible. The filter is realized by selecting those keyframes for which the lower half of the image is less than 80% dark.

Together, these filters effectively reduce the total data-set of approximately 7000 keyframes to only 26, containing three of the four known items. Recall is improved using a follow-up query, ranking the images with a 'Black Sky' using the spatial color histogram method on a seed image drawn from the previous phase. This second step returns the four known items in the top-10.

Topic 8: Jupiter

The Jupiter topic is another example that benefits significantly from query articulation. At a first thought, this query may seem to be easy to solve, as planets have a typical appearance (a colored circle surrounded by black) and Jupiter should be easily recognized. But, examining the example images shown in Figure 5, it is apparent that colors in different photos of Jupiter can differ significantly.

An important characteristic of Jupiter is the distinguishable orange and white lines crossing its surface. Articulating this through color content, we decided to put emphasis on the orange content, the white content, and their interrelationships, expressed as filters on color-set *correlograms* [?]. Computing correlograms from the color-sets shown in Figure 6 produces 9-dimensional feature vectors, one dimension for each possible transition. To ensure that the results are not dominated by the auto-correlation coefficients, the resulting vectors are weighted using the inverse of their corresponding coefficients in the query images. The derived query finally finds some of the known-items, but recall remains low.

Another way to emphasize the striped appearance of Jupiter is to detect the actual presence of (horizontal) lines in images and rank the keyframes based on that presence. This was implemented by means of DCT-coefficients, classifying each DCT-matrix in the luminance channel of a keyframe into texture-classes. We used the classes 'horizontal-line', 'vertical-line', 'blank' and 'other'. The cheap method of ranking by simple statistics on these texture-classes proved only slightly worse than the previous (elaborate and expensive) method based on correlograms.

Although a combination of both results did not retrieve any additional answers, a minor improvement is obtained through a subsequent search, seeded with a retrieved shot found before.

Topic 25: Starwars

Finding the Starwars scene became a matter of honor, since we submitted the topic ourselves – perhaps a bit over-enthusiastically. After several unfruitful attempts using color histograms and color-sets, we decided to articulate the query by modeling the golden appearance of one of the robots, C3PO. This idea might work well, as we do not expect to find *many* golden objects in the data-set.

The appearance of gold does not simply correspond to the occurrence of a range of colors; its most distinguishing characteristic derives from the fact it is a *shiny* material, implying the presence of small, sharp highlights. We implemented two stages of boolean filters to capture these properties, followed by a custom ranking procedure.

The first filter selects only those images that

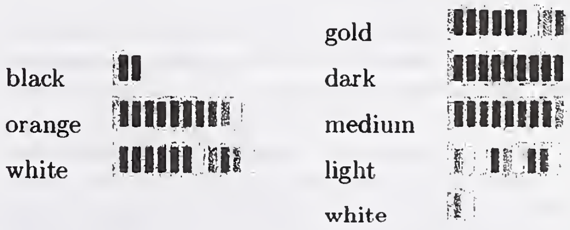


Figure 6: Color-sets used in the Jupiter (left) and the Starwars (right) topics.

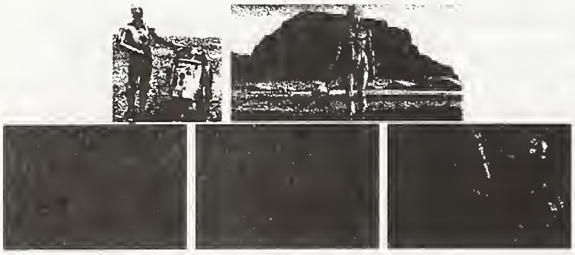


Figure 7: Topic 25, Starwars, examples(left 2 images) and the correct answers keyframes.

have sufficient amount of golden content. It checks whether images have at least 20% 'golden' pixels, using the gold color-set defined in Figure 6. Secondly, a set of filters reduces the data-set by selecting those images that contain the color(set)s shown, representing the appearance of gold in different lighting conditions, in a way expected for shiny metallic surfaces: a bit of white, some light-gold, a lot of medium-gold, and some dark-gold. Although the precise percentages to be selected are difficult to choose correctly, we believe the underlying idea is valid, as we modeled expected levels of gold-content for a shiny-gold robot.

The resulting subset is then ranked using another characteristic of shiny surfaces: the expected spatial relations between those color-sets (white highlights surrounded by light-gold spots, surrounded by medium-gold surfaces, which in turn are surrounded with dark-golden edges). We expressed this property using color correlograms, ranking the relevant transitions.

Using this elaborate approach, we managed to retrieve one of the correct answers, but no higher than position 30. We retrieve many 'golden' images with elements satisfying our limited definition of shininess (most of them not 'metallic'), but the properties of metal surfaces must be modeled more realistically to get more convincing results.

Topic 32: Helicopter

The helicopter topic provides three audio examples, and we experimented with the audio analogon of query articulation in an attempt to find scenes with helicopters. We hoped to specify the characteristics of a helicopter sound as a combination of two filters: (1) a repetitive pattern using periodicity of the audio spectrum, and (2) a concentration of energy in the lower frequencies, using spectral centroid and bandwidth features. Details of the techniques we tried can be found in [17].

Unfortunately, the helicopter sound in the known-item can only be noticed in the background, and some characteristics of the speech voice-over overlap with the idea of the second filter. It turns out the combination of filters can detect sounds corresponding to vehicles and airplanes, but we have not managed to tune the filters such that it singles out helicopters only.

4.3 Reflection

The highlighted known-item searches illustrate the idea underlying the process of query articulation, and demonstrate how query articulation may improve the results of multimedia retrieval dramatically. Without the elicitation of such relatively exact queries, none of these topics could be solved using our limited feature models. The query articulation process studied for topics 25 and 32 (and even for topic 8) suffered however from the risk of overemphasizing precision, sacrificing overall recall. Especially if the features available in the system do not correspond closely to the particular characteristics of the desired result set, the current system does not provide sufficient support to assess suitability of candidate strategies. But, also if appropriate features are available, the resulting query may 'overlook' other possibilities; for example, our strategy would not find the lunar rover if appearing in a lunar crater or in a hangar on earth (so there is no visible black sky).

5 Lazy Users

In our interactive experiments, we assumed a 'lazy user' model: users investing only limited effort to express their information need. Our users view 20 result summaries at a time, after which they choose whether to look at more results from the current strategy, or formulate a new strategy. They are not expected to investigate more than 100 result summaries in total. Lazy users identify result sets instead of correct answers, so our interactive results are not 100% precision.

The combination strategies used to construct run 5 consisted of:

- choose the run that looks best;
- concatenate or interleave top-N from various runs;
- continue with an automatic, seeded search strategy.

For example, the strategy for topic 24 (Lynn Bondurant) used a seeded search based on run 3, which was interleaved with the results of run 4. Surprisingly, the run with speech transcripts only turns out better than the combined run, although not on all topics. It has proven difficult to combine results of multiple input runs effectively. While lack of time did also play a role (the combination strategies were not tried very systematically), the results for topics 54 and 59 demonstrate that a lazy user can, based on a visual impression of a result set, inadvertently decide to discard the better results (in both cases, run 3 was better but run 4 was chosen as best answer). Tool support for such a combination process seems a promising and worthwhile research direction.

6 Discussion

A major goal of having a video retrieval task at TREC-10 was to research a meta-question: investigate (experimentally, through a 'dry-run') *how* video retrieval systems should be evaluated. Working on the task, we identified three concerns with the current setup of the evaluation:

- the inhomogeneity of the topics;
- the low quality of the data;
- the evaluation measures used.

Candidate participants all contributed a small number of multimedia topics, the union of which formed the topic set. Partly as a result of the different angles from which the problem of video retrieval can be approached, the resulting topic set is very inhomogeneous. The topic text may describe the information need concisely, but can also provide a detailed elucidation; topics can test particular detectors, or request very high-level information; and some topic definitions are plainly confusing, like 'sailboat on the beach' which uses a yacht on the sea as image example[6]. Thus, each subtask consisted of a mix of (at least) three distinct classes of topics: detector-testers, precise known-item topics, and generic searches. This inhomogeneity causes two problems: it complicates query analysis for automatic systems, and makes comparison between runs difficult (a single good detector can easily dominate an overall score like average precision).

The low quality of the video data provided another unexpected challenge. It makes some topics more complex than they seemed at first sight (like 'Jupiter'). Also, the results obtained with the technique discussed in Section 2.2.5 are much lower than the application of the same paradigm on for example the Corel photo gallery. In fact, we observed that in many cases the color distributions to a large extent are a better indication of the similarity in age of the data than of the true video content. Of course, this can also be viewed as a feature of this data set rather than a concern. Experiments discussed by Hampapur in [5] showed as well how techniques behaving nicely on homogeneous, high quality data sets are of little value when applied to finding illegal copies of video footage on the web (recorded and digitized with widely varying equipment).

The third concern, about the evaluation measures, is based on two slightly distinct observations. First, our lazy user model returns shots as answers for known-item queries, but these are often shorter than 1/3 of the scenes that should be found. The chosen evaluation metric for known-item topics thus deems our answers not relevant, while this could be considered open for discussion: a user could easily rewind to the start of the scene.

Second, an experimental setup that solves the interactive topics by handpicking correct answers should probably result into 100% precision answer sets. First of all, this indicates that precision is not the right measure to evaluate the results of the interactive task. Lower scores on precision only indicate inter-assessor disagreement (viewing the user as just another assessor), instead of the precision of the result set. Another example of this phenomenon can be found in the judgments for topic 59 on runs 4 and 5, where identical results were judged differently.[7] The significant difference in measured performance indicate that the current topics and relevance judgments should probably not be used as ground truth data for laboratory experiments.

As a concluding remark, it is not so clear how realistic the task is. First of all, no participant seemed to know how to create 'doable' topics for the BBC data, while those video clips are drawn from a real video archive. Also, it seems unlikely that a user with state-of-the-art video retrieval tools could have beaten a naive user who simply scrolls through the relatively small set of keyframes. A larger collection would give video retrieval systems a fairer chance, but the engineering problems (and cost) arising might discourage participation in the task.

7 Conclusions

In spite of the issues raised in the discussion, we believe the TREC video evaluation is a strong initiative

[6] Shame on us – we contributed this topic ourselves.

[7] This may also have been a case of *intra*-assessor disagreement.

that was much needed to advance the field of multimedia retrieval, and it has already pointed us to a range of problems that we may never have thought of without participation.

Our evaluation demonstrates the importance of combining various techniques to analyze the multiple modalities. The optimal technique depends always on the query; both visual and speech can prove to be the key determining factor, while user interaction is crucial in most cases. The final experiment attempted to deploy all available information, and it seems worthwhile to investigate in research into better techniques to support choosing a good combination of approaches. In some cases, this choice can already be made automatically, as demonstrated in run 1; but, in cases like the known-item searches discussed for run 4, user interaction is still required to decide upon a good strategy.

Our (admittedly poor) results identify many issues for future research: new and improved detectors (better suited for low-quality data), better combination strategies, and more intelligent use of the user's knowledge. The integration of supervised and unsupervised techniques for query formulation form a particular research challenge.

Acknowledgments

Many thanks go to Alex Hauptman of Carnegie Mellon University for providing the output of the CMU large-vocabulary speech recognition system.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern information retrieval.* Addison-Wesley, Wokingham, UK, 1999. 3

[2] P. Bosch, A. van Ballegooij, A.P. de Vries, and M.L. Kers ten. Exact matching in image databases. In *Proceedings of the 2001 IEEE International Conference on Multi media and Expo (ICME2001)*, pages 513–516, Tokyo, Japan, August 22-25 2001. 6

[3] Arjen P. de Vries. The Mirror DBMS at TREC-9. In Voorhees and Harman [19], pages 171–177. 3

[4] J. M. Geusebroek, R. van den Boomgaard, A. W. M. Smeulders, and H. Geerts. Color invariance. *IEEE Trans. Pattern Anal. Machine Intell.*, to appear, November, 2001. 3

[5] A. Hampapur and R. Bolle. Comparison of distance measures for video copy detection. In *Proceedings of the 2001 IEEE International Conference on Multi media and Expo (ICME2001)*, Tokyo, Japan, August 22–25 2001. 9

[6] Djoerd Hiemstra. *Using language models for information retrieval.* PhD thesis, Centre for Telematics and Information Technology, University of Twente, 2001. 3, 4, 5

[7] Djoerd Hiemstra and Wessel Kraaij. Twenty-One at TREC-7: Ad-hoc and cross-language track. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Seventh Text Retrieval Conference TREC-7*, number 500–242 in NIST Special publications, pages 227–238, 1999. 3

[8] J. Huang, S.R. Kumar, M. Mitra, W. Zhu, and R. Zahib. Spatial Color Indexing and Applications. *International journal of Computer Vision*, 35(3):245–268, 1999. 7

[9] Philippe Joly and Hae-Kwan Kim. Efficient automatic analysis of camera work and microsegmentation of video using spatiotemporal images. *Signal Processing: Image Communication*, 8(4):295–307, 1996. 2

[10] Wessel Kraaij and Thijs Westerveld. TNO/UT at TREC-9: How different are web documents? In Voorhees and Harman [19], pages 665–671. 3

[11] N.M. Laird, A.P. Dempster, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, series B*, 39(1):1–38, 1977. 5

[12] H.A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1), 1998. 3

[13] A.W.M. Smeulders, S. Santini M. Worring, A. Gupta, and R. Jain. Content based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, Dec. 2000. 1, 2

[14] J.R. Smith and S.-F. Chang. VisualSEEk: a fully automated content-based image query system. In *ACM Multimedia 96*, Boston, MA, 1996. 6

[15] C.G.M. Snoek. Camera distance classification: Indexing video shots based on visual features. Master's thesis, Universiteit van Amsterdam, October 2000. 3

[16] TextBridge SDK 4.5. http://www.scansoft.com. 3

[17] Alex van Ballegooij, Johan List, and Arjen P. de Vries. Participating in Video-TREC with Monet. Technical report, CWI, 2001. 6, 8

[18] N. Vasconcelos and A. Lippman. Embedded mixture modelling for efficient probabilistic content-based indexing and retrieval. In *Multimedia Storage and Archiving Systems III*, volume 3527 of *Proceedings of the SPIE*, pages 134–143, 1998. 3, 4, 5

[19] E.M. Voorhees and D.K. Harman, editors. *Proceedings of the Ninth Text Retrieval Conference (TREC-9)*, number 500–249 in NIST Special publications, 2001. 10

[20] M. Worring and L. Todoran. Segmentation of color documents by line oriented clustering using spatial information. In *International Conference on Document Analysis and Recognition ICDAR'99*, pages 67–70, Bangalore, India, 1999. 3

Dublin City University Video Track Experiments for TREC 2001

P. Browne, C. Gurrin, H. Lee, K. Mc Donald, S. Sav, A. F. Smeaton and J. Ye

Centre for Digital Video Processing
Dublin City University
Ireland
Alan.Smeaton@compapp.dcu.ie

Abstract

Dublin City University participated in the interactive search task and Shot Boundary Detection task[*] *of the TREC Video Track. In the interactive search task experiment thirty people used three different digital video browsers to find video segments matching the given topics. Each user was under a time constraint of six minutes for each topic assigned to them. The purpose of this experiment was to compare video browsers and so a method was developed for combining independent users' results for a topic into one set of results. Collated results based on thirty users are available herein though individual users' and browsers' results are currently unavailable for comparison. Our purpose in participating in this TREC track was to create the ground truth within the TREC framework, which will allow us to do direct browser performance comparisons.*

1. Introduction

This year Dublin City University took part in the video track where we submitted runs for both the interactive search task and the automatic shot boundary task. Firstly we will present our experiments and results for the interactive task in section 2. In section 3 we will briefly discuss our Shot Boundary Detection experiments. Although we have an interest in the Shot Boundary Detection task our primary focus was on participating in the interactive task. We have submitted some Shot Boundary Detection results and are continuing our research into the area and look forward to being in a position to participate more fully in the Shot Boundary Detection task next year, should it run.

2. Interactive Search Task

For the interactive search task we undertook to evaluate 3 different interfaces for browsing digital video. In the following sections we will give an overview of the system that was created in order to evaluate the browsers, then the evaluation procedure for experiments, followed by how experimental data was collated from the different users and finally we will present our initial analysis of the results.

2.1 System Description

For the interactive search task, we used 3 of the 8 video keyframe browsers available in the Físchlár system. The Físchlár system is web-based and allows users to record, browse, and watch television programmes online [O'Connor *et al.* 01]. For this experiment we created a separate interface (Section 2.1.2) from Físchlár. The 3 browsers were chosen for the large differences among them in the way they presented the keyframes (representative frame from each video shot), while at the same time all of them had the same dynamic and immediate response style of interaction. The test users used all these browsers to locate the video clip results for interactive queries.

[*] Work on the Shot Boundary Detection task was done in collaboration with the Multimedia & Vision Research Lab in Queen Mary, University of London, U.K.

2.1.1 The Three Browser Interfaces

The underlying Físchlár system processed the TREC video set to automatically extract keyframes for each video and the 3 browsers present these sets of keyframes in 3 different ways. The 3 browsers are the Timeline browser, Slide Show browser, and Hierarchical browser. These are described here briefly.

The *Timeline* browser (see Figure 1) presents keyframes in a linear fashion. When a user selects one of the videos, the first 24 keyframes are displayed on the screen, with that part of the video indicated on the timeline at the top. Keyframes are miniaturised so that many of them can fit on the screen while still the content of each is recognisable. The timeline is broken into segments, each representing the appropriate time when the 24 keyframes below appear in the video. The timeline is mouse-over activated so when the user moves the mouse cursor over any of the timeline segments, the keyframes immediately change to display that part of the video on the screen. This way, one sweep of the mouse through the timeline bar from left to right can quickly present the whole keyframe set of a video. When a mouse cursor is on the timeline, a small ToolTip pops up to indicate the time of that segment.

Figure 1: Timeline browser

The *Slide Show* browser (see Figure 2) presents keyframes one by one, in a temporal fashion. When a user selects one of the videos, keyframes from the video will be displayed one by one, automatically flipping from one keyframe to the next as in a slide show. The size of the keyframes is larger than in the Timeline browser, as it shows only one keyframe on the screen at a time. The user has control over the keyframe flipping – she[*] may pause the slide show, flip through manually by clicking the forward and backward buttons, or leave it to do the slide show by itself. Also the user can put the mouse cursor over the small timeline below the keyframe, and drag the current point quickly back and forth, similar to the Timeline bar. When the mouse cursor is on the timeline, a box pops up displaying a small keyframe (smaller than the miniaturised keyframes in the Timeline browser) representing the cursor's point in the timeline along with the time of that point.

Figure 2: Slide Show browser

The *Hierarchical* browser (see Figure 3) presents keyframes in a 4-level, hierarchical fashion. When a user selects one of the videos, 6 keyframes that are representative of the video are displayed on the screen (the top 6 keyframes in Figure 3). These 6 keyframes are selected throughout the chosen video's content, representing a rough overview of the video. When the user moves the mouse cursor over any of these 6 keyframes, another 6 keyframes within the segment represented by that keyframe are displayed just below the top 6 keyframes, showing more details of that segment. The user can again move the mouse to this second level of keyframes to show 6 more detailed keyframes below it.

Figure 3: Hierarchical browser

This way, the user can quickly move the mouse cursor over any keyframe displayed on the screen, hierarchically drilling up and down the keyframe set. This particular style of keyframe browsing has earlier been mentioned elsewhere [Mills *et al.* 92] [Zhang *et al.* 95].

[*] There are, of course, male users as well.

2.1.2 The Evaluation Suite

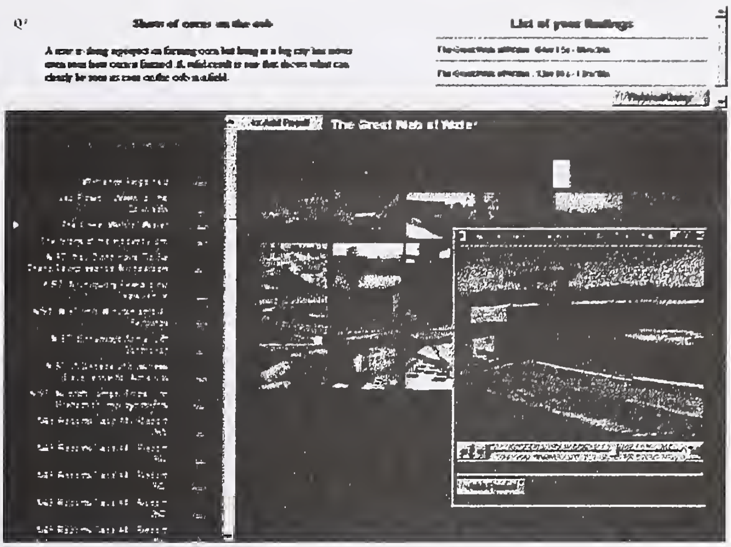


Figure 4: Evaluation suite

We used a specially designed, automated web-based evaluation suite for the interactive testing, which integrated all the test users' tasks to be conducted. Test users were presented with a user-interface that presents each of the queries, one of the 3 browsers to do the search task, and an input panel to indicate the findings. Figure 4 shows a screen shot from the user-interface of the evaluation suite. When the user clicks the start button (not shown in Figure 4), the first query window pops up with its description and example images and videos (if examples are provided by the query). The user can close the query description window to do the search task, but the description part remains on the top left of the screen (in Figure 4) so that she can refer to it while doing the task. The bottom left side of the screen shows the list of video clips. The user can click on a video title in this list to see the keyframes of that video on the right side, with one of the pre-determined browsers. For any of the 3 browsers that are used for a task, clicking on any of the displayed keyframes will pop up a small player window that starts playing the video from the clicked keyframe onwards. When the user browses and finds a part of the video which she thinks satisfies the query, she clicks the 'Add Result' button either on the browser screen or on the player window, which pops up an input panel window where the user indicates start and end time of the video result. The indicated finding will be added on the list at the top right side of the screen for the user to see, and as she finds and adds more results, this list will grow. Once a result is added it cannot be edited or deleted. The user will continue the search task until the experimenter asks her to stop (in 6 minutes - see Procedure section below)..

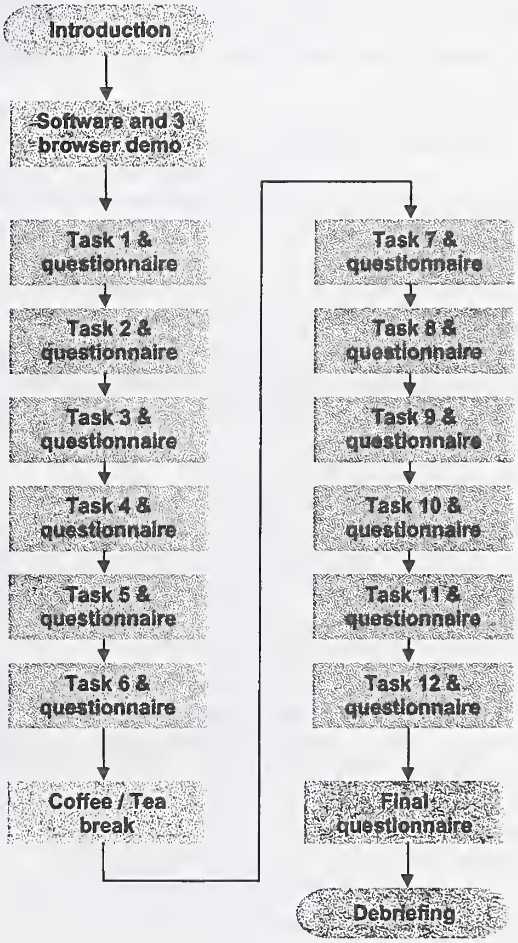


Figure 5: Session procedure

Table 1: Test user demographics

by gender		by status	
Male Female	18 12	Undergraduates Postgraduates Staff	10 19 1
Total	30	Total	30

2.2 Evaluation Procedure for Experiments

The test users volunteered within the School of Computer Applications and the School of Electronic Engineering, and were asked to come to a computer lab where the testing was to be held.

2.2.1 Evaluation Environment

In each session 5-7 test users participated, depending on their time availability, and a total of 6 sessions were conducted on different days. The total number of participants was 36 people, from which 6 people's results were discarded due to a network congestion problem that disrupted the tasks of those people. The demographics of the test users are in Table 1. They were all experienced users of the Microsoft Windows operating environment.

Each session took about 3 hours, but the exact time varied due to test users who came late, their questions during the introduction and debriefing stage. Test users sat in front of their assigned PC with the web browser displaying the first screen of the specially designed web-based suite for this evaluation. Test users filled in a very short demographic questionnaire asking gender, course/year, and familiarity with Físchlár browsers. The question of familiarity with the browsers was asked because the system had a wide availability within the campus and many students have been using its browsers on a daily basis, which would make differences in their task performance. Then the experimenter briefly gave an introduction, thanking them for their participation, telling them what the purpose of the testing was, how long it would take, and how stressful it would be. They were informed that they could leave at any point if they felt unhappy, frustrated or too stressed, and that the information they provided would be confidential and only used for research purposes.

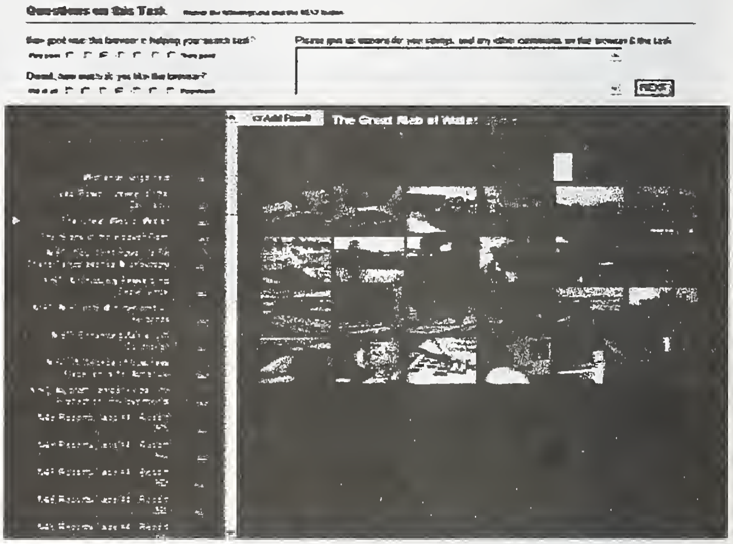


Figure 6: Post-task questionnaire

After the introduction, the experimenter and 2 - 3 more assistants (who stayed and helped during the sessions) gave a 10 - 15 minute demonstration by gathering 3 - 4 test users in a group. They showed them how to interact with the evaluation software (i.e., how to start, how to use 3 different browsers, how to add the findings into the result list, how to proceed, etc.), and replied to the questions the individual test users had.

After the demonstration, the experimenter asked the test users to start the first task and started the stopwatch. Each task lasted 6 minutes, during which time the test users read the assigned query with example images and/or video clips and tried to find the segments matching the query and add results. The query and the video browser they used were automatically assigned (see Section 2.2.2). They were asked to find as many answer segments as possible, for there can be more than one answer for a query. After 6 minutes of the task, the experimenter asked them to stop the task and click the "Finished query" button, which brings them to a short questionnaire asking them to indicate how good the browser was in completing the task, how much they liked the browser, and then an open question about the browser and the task (see Figure 6). While filling in this questionnaire the browser was available on the bottom of the screen so that the users can still try the browser. During this period the experimenter and the assistants stayed away

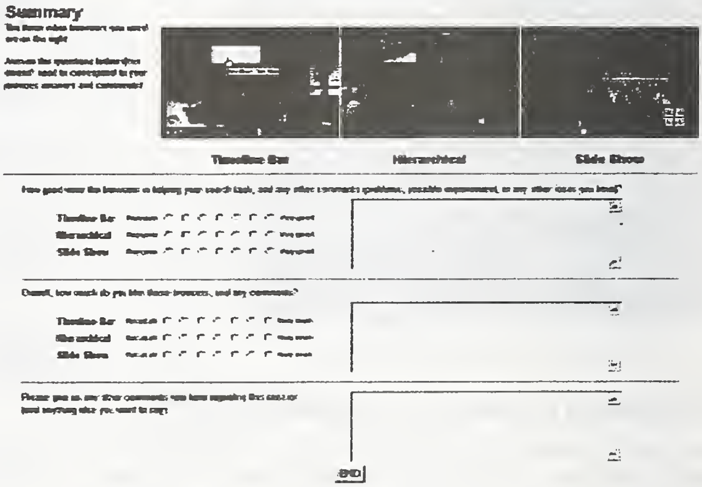


Figure 7: Final questionnaire

from the test users so that they could feel free to give ratings and comments without being conscious of others. After finishing the questionnaire, the users were asked to click the "NEXT" button to continue to the second task, which would again take 6 minutes.

Test users did a total of 12 tasks (i.e. 12 different queries) in this fashion, and at the end of 6th task there was a 10 - 15 minute break to provide them with rest so that their performance would not be influenced by their tiredness too much in later tasks. During the break they were provided with refreshments (coffee/tea with biscuits), and asked to feel free to go out of the lab or chat with other test users or with the experimenter and assistants while having refreshment.

At the end of 12th task (last task), the screen displayed a summary of 3 browsers with the final questionnaire (see Figure 7 above). This asks them to rate the browsers by their task completion efficiency and their subjective preference, with plenty of space for any comments and suggestions on any aspect of the browsers and the testing on the whole.

Finally, the test users clicked the "END" button at the bottom, which displayed a simple thanking message indicating the end of the session. Test users were asked to plug off the headphones or earphones they used and bring it with them.

Considering the long duration of each session of 3 hours or so, the middle break and the highly interactive and novel video browsers seemed to keep the users active and engaged throughout. There was no monetary payment for their participation, but the interaction with interesting video browsers, free refreshment and free headphones or earphones seemed to be relatively effective.

2.2.2 Avoiding Bias in the Experiments

There was concern that without careful planning the integrity of our experiments could be compromised by the problem of user bias. There are a number of reasons why this could be the case:

- A user may become too familiar with the browser interfaces as more topics were processed. Since the users were operating under time constraints it is feasible that a user may be quicker at finding relevant clips after using a particular browser on a number of previous occasions.
- A user may become too familiar with the dataset and remember where some video clips had been seen before. We had to avoid the possibility of all users processing topics in some standard order, which would aid the expected user performance on latter topics.

To avoid this problem we had to avoid presenting the users with a number of topics in a uniform order. We did not wish to limit the number of topics that a user processed and therefore we designed the experiment to facilitate each topic being processed (in random order) by twelve different users from among the total of 30 users. Each user processed twelve topics resulting in a total of 360 user results. Before running the experiments, we drew up the following constraints:

- That no user processes the same topic twice, regardless of the interface used.
- That each user processes twelve topics using the three browsers, where each browser was used four times, and the browsers were presented in random order. In this way, the results provided to the system will not be skewed by one user gaining more experience with one browser over the others.
- That each topic must be processed by the same number of users (12) and each interface used for a topic must also be processed by the same number of users (4).
- That, like the browsers, the order in which the topics are processed by each user must be random. In this way we hoped to avoid any issues arising out of the fact that many users could process a particular topic later on in the experiment when these users would be more familiar with the interfaces and dataset.

The following table shows a summary of the experiment variables.

Table 2: Experiment variables for program to generate topic and browser sequence for users

Number of users partaking in the experiments	30
Number of topics	30
Number of interfaces to evaluate	3
Number of interface evaluations (each) per topic	4
Total number of user evaluations per topic	12
Total number of evaluations carried out	360
Mean position of a topic (in order it was presented to the users) for evaluation. This value is in the range (0...11) representing (first...last)	5.5
Standard deviation of topic positions about the mean	0.8

2.3 Collating Experimental Data

Due to the lack of verification on user results as they entered them in the interface, it was necessary to correct invalid results before we could combine and rank the results for each topic. The following rules were applied in the given order to remove and correct invalid results:

- A result with *both* its start time and end time specified as "0m0s" (the default) was removed. In this case a user did not specify which part of a selected video is relevant to the topic.
- A result was removed if its start time was out of range for the specified video. It is unreasonable to do any modification of the result as the start time is invalid.
- A result with the start time specified as "0m0s" was changed so that its start time was "0m1s". Some users who wished to specify the beginning of a clip selected 0m0s but this was not allowed by the NIST checker program.
- A result with the end time specified as "0m0s" was changed to have an end time of the start time plus 1s. The results specified by users sometimes did not include an end time resulting in "0m0s" (the default) being set as the end time.
- One second was added to the end time of a result if the value of its start time and end time were equal. Some users probably considered that only one particular frame of the whole video clip was enough to meet the requirements of a specific topic. However, the NIST checker program asserts that a valid result must not have equal start and end times.
- If a result's end time was out of range or earlier than its start time, the end time of the result was modified by giving it a value of its start time plus 1 second. In this situation, it was assumed that a user specified a valid start time but erred in specifying the end time.

Having corrected or removed the invalid results, we then combined the results from each user as the TREC framework allowed us to submit only two full runs. The reader should remember that our purpose in taking part in this TREC track was to have our own relevance "clips" assessed for relevance and thus allowing us to explore our browser vs. browser comparison experiments *after* the ground truth becomes known. Because each "run" was the combination of inputs from several users, before submitting our own results to TREC we needed to remove duplicate and overlapping results. When results from the same user overlapped we replace them by a single result which overlapped the set of clips. When the overlapping results were not from the same user we took the earliest start time that is valid for the majority of overlapping results. The end time was chosen by taking the latest end time that is also valid for the majority of the overlapping results. For an overlap of only two results this simplifies to replacing them with their intersection.

Example 1:

```
<item seqNum="1" src="nad58.mpg" start="1m10s" stop="1m26s"/>
<item seqNum="2" src="nad58.mpg" start="1m12s" stop="1m34s"/>
<item seqNum="3" src="nad58.mpg" start="1m15s" stop="1m29s"/>
```

Result:

```
<item seqNum="1" src="nad58.mpg" start="1m12s" stop="1m29s"/>
```

Example 2:

```
<item seqNum="4" src="anni010.mpg" start="1m4s" stop="1m25s"/>
<item seqNum="5" src="anni010.mpg" start="1m20s" stop="3m52s"/>
```

Result:

```
<item seqNum="4" src="anni010.mpg" start="1m20s" stop="1m25s"/>
```

Finally the query results for each topic were ranked based on the number of duplicates that produced each result. The greater the number of duplicates the higher the ranking assigned to a result. For example: a result A was derived from three duplicate results and a result B from four duplicates, then result B will be allocated a higher precedence. The result items within these precedence levels were further ranked by the number of results for the video in the source results of the topic: the more results from a source video, the higher its ranking.

2.4 Interactive Search Task Results and Analysis

The precision and recall figures for our interactive search results are available in Table 3. The results for the search topics are separated into two groups, *general topics* and *known item topics*. Topics from these two groups were evaluated differently. The *general topics* are general statements of information need. The *known item topics* on the other hand are a more specific information need and their correct results, the known items, were specified during topic creation. The *general topics* have higher evaluation cost and these are evaluated using human assessors who look at each result and give a judgement of whether it is relevant or non-relevant. Currently, precision is available for these topics but recall figures for this type of search are unavailable. For the known item topics it is possible to evaluate them using an automatic method. So far, an overlap measure is used to automatically classify the results for these known item topics as relevant or non-relevant. The overlap measure has two control parameters, *minimum known item coverage* and *minimum result item coverage* that determine whether a result item is considered to match a known item. A result item is considered to match a known item if and only if the following two conditions hold:

1. The ratio of the length of the intersection of the result item and the known item to the length of the known item is at least as large as the *minimum known item coverage.*
2. The ratio of the length of the intersection of the result item and the known item to the length of the result item is at least as large as the *minimum result item coverage.*

Table 3: General Search and Known Item Precision and Recall

Search Type	**Minimum Known Item Coverage**	**Minimum Result Item Coverage**	**Precision**	**Recall**
General Search	N/A	N/A	0.84	N/A
Known Item ('low' recall, 'low' precision)	0.333	0.333	0.298	0.419
Known Item ('low' recall, 'high' precision)	0.333	0.666	0.236	0.356
Known Item ('high' recall, 'low' precision)	0.666	0.333	0.226	0.300
Known Item ('high' recall, 'high' precision)	0.666	0.666	0.164	0.237

The known item recall and precision has been calculated at four different combinations of the control parameters (Table 3). The performance of our general topic results (P=0.84) is far higher than for our known item topic results (P=0.298) even at the most generous of control parameters 0.333, 0.333. We believe that this is a result of the different evaluation methods used for the topic types instead of some fundamental difference between the topics in the two types. A lower control parameter level of say 0.2 or 0.1 may result in the overlap measure giving results that are closer to those the human assessors were giving for the topics. Further exploration of the result of the overlap measure at different control parameter values is required and this will be done by us, *post-TREC.*

The search types *known item search* and *general search* are not as distinctive as their labels and different evaluation methods may suggest. In some cases a topic could be either a known item or a general search depending on whether the submitting group indicated the results when submitting the topic. For example Topic 24: "Find all pictures of R. Lynn Bondurant" is classified as a general search. But Topic 22: "Find pictures of Harry Hertz..." and Topic 23: "Find images of Lou Gossett Jr." are classified as known item searches not general searches. Also some of the currently classified known item searches could also be classified as general. For example, Topic 30: "Scenes with buildings similar to dome-shaped House of Congress" could be a general search not a known item search. The topics that are classified into search types general and not general (known item) is more a result of whether we specified queries with known results or not, rather than of some fundamental difference.

Some of the loss of precision by our users can be put down to the lack of fine-grained frame-level editing facilities for their search results. Once a user added a search result it was set in stone and could not be refined or removed. Furthermore, a user specified the start and stop boundaries on a video clip to the nearest second, whereas the TREC evaluation is much more fine-grained than that and this obviously resulted in some irrelevant videos being included in our submissions. The precision loss for our results may also be attributed to the different interpretation of relevance by the assessors or topic creators and by our test users as our users sometimes were more willing to accept a clip as relevant. This may just be the nature of the experiment in that we did not stress enough to the users that they should be completely sure of relevance to the topic before adding it as a result. It is also probable that our users thought that having some result was better than none so, they may have knowingly added results that were nearly correct.

Interface errors and interpretation errors may account for the loss of precision in the general search results but it cannot account for the far lower results in the known item search results. In fact, for 5 of the known item queries no correct results were found using the overlap measure at the parameter level 0.333, 0.333 – that is the precision and recall value was 0 (Topic 1, 27, 28, 29, 33). On inspection of our results it is clear that our users found correct results but they did not enter the start time and end time strictly enough and did not segment adjacent results into separate results. For example, for topic 1 our system 1st result (10 seconds long) overlaps 3 of the known item results but under the overlap measure even at parameter values 0.333, 0.333 none of the 3 known-items are attributed as matched (.5, 3, 2 seconds intersections respectively). Since the current overlap measure does not take into account other known items overlapping the results item when calculating result item coverage ratio it is unforgiving when a user has specified a single result that contains temporally close known-items. Some of the known items are less than a second apart (0.5s in Topic 1). The test users without prior explicit direction may interpret known-items in adjacent shots as one result.

Even without the multiple known items in a result issue the precision at which our users specified the results was to seconds. Often users specified results with more than a couple of seconds padding before and after the known item. For a known item of small duration the results item coverage would be very small and therefore it may not be found relevant at result item overlap coverage of 0.333 or greater. In Topic 1 alone two known items are 0.5s long and another is 0.6s long. If our test users had the facilities to segment and refine their results to more exact timing than to seconds the results would be considerably improved. Of course, we should also have stressed to the users during the experiment that they should only add the maximum relevant continuous camera shot segment that matches the topic.

Our results show that our known item results are slightly more sensitive to known item coverage than to result item coverage, but the difference is not great. The majority overlap measure degrades into intersection for situations where there are only two overlapping results (Section 2.3). If the overlapping results are both covering two different but temporally close known items, then the intersection will be covering neither. This may account for why our experiment is more sensitive to known item coverage. Perhaps, the overlapping measure should be changed to start point calculated as the mid point of start points and end point calculated as the mid point of end points of the two overlapping results in this special case of only two overlapping results. But even still this would result in a single result that probably only partially covers the two known items. And with the current metric even at the 0.333, 0.333 parameter level no known item may be matched. It may be even better to replace the majority voting method for overlap to one of average start time and average end time. These are all issues which we need to address in the near future.

3. Shot Boundary Detection

The Shot Boundary Detection system we evaluated in the video track makes use of the compression features of MPEG encoded video. This was done in order to achieve performance with a minimum of processing requirements as the decoding of MPEG video sequences is relatively time consuming and in areas where speed is an issue, compressed domain processing can offer high accuracy in a fraction of real time.

The MPEG video standard [LeGall *et al.* 96] achieves compression by exploiting the data redundancy present in video sequences. As Shot Boundary Detection is a temporal segmentation of a video sequence, the temporal compression features of the MPEG video standard can be used to help the Shot Boundary Detection task. The macroblock is a primary element in the MPEG standard and has a determinant role in the temporal compression of the video. The following considerations have been used for our Shot Boundary Detection:

- In general for most B frames a video encoder will trend to favour bi-directional predicted macroblocks in order to achieve better compression.

- B frames tend to use preponderant prediction from the nearest reference frame. Thus in a *group of pictures* with RBBR encoding pattern (where R denotes a reference frame and B a B frame), first B frame would have dominant prediction from first R frame and second B frame from second R frame. That led to a bigger number of forward predicted macroblocks than backward predicted macroblocks in first B frame and opposite for second B frame.

If a macroblock distribution in the currently evaluated frame sequence does not comply with the considerations above it is highly probable that a shot boundary has occurred and the changing of the dominant reference frame would designate the exact position of a shot boundary. An increase of the intra-coded macroblocks in the P frames may indicate a possible gradual transition.

It is evident that the files from Shot Boundary Detection test collection for the video track come from a few distinct encoding sources and it likely that we will have similar content and video coding effects within files provided from the same source. Therefore, because of resources available to us at the time, it was not considered possible to carry out an evaluation over the entire collection. For evaluation we selected a set of five representative files, one from each source, regarding as representative the file with the longest playing time within each source or the file which apparently contains more or more complex, shot transitions. Furthermore, as our Shot Boundary Detection system is based on video compressed domain features, characteristics of the video encoders such as image size, frame encoding pattern and the encoded macroblock types, were considered in order to select various encoding parameters. Our official results are presented below in Table 4 for all transition types.

Table 4: Shot Boundary Detection performance figures for 5 files from the dataset

Files \ Metrics	Reference transitions	Deletion rate DR	Insertion rate IR	Precision Pr	Recall Re
`ahf1.mpg`	107	0.158	0.074	0.919	0.850
`anni009.mpg`	103	0.708	0.009	0.967	0.291
`bor03.mpg`	237	0.573	0.050	0.893	0.426
`ldoi874a_s1_02_004.mpg`	7	0.00	0.142	0.875	1.00
`nad28.mpg`	298	0.221	0.050	0.939	0.778

Weighted column mean	0.386	0.049	0.925	0.613

Our Shot Boundary Detection work will be evaluated on the full dataset at a later time and our results above are not directly comparable to the results of other TREC track participants. However, the most interesting part of the work we report above is the computation time taken; running on a 733 MHz Pentium III PC with 256 MB RAM running Red Hat 7.0 Linux, the 76 min 39s of video took 4 minutes and 2 seconds of computation time, about 5% of real time.

4. Conclusions

Our results for general search topics and known item topics show two very different results for precision and recall even though both sets of topics were performed under the same environmental conditions. Indeed, our users were unaware of the query types when performing the search tasks for each topic. Adjusting the metric used in the known-item search to account for our particular "flexibility" in determining the start and end of clips identified by users, will be necessary and will be done post-TREC. We will also experiment more with different ways of collating independently gathered results into one non-overlapping ranked results list. The current method may not be one of the best but without further experimentation this is but guesswork. Further study of our results for individual users and of the browsers will be conducted.

References

[LeGall *et al.* 96] LeGall, D., Mitchell, J.L., Pennbaker, W.B. and Fogg, C.E. *MPEG video compression standard.* Chapman & Hall, New York, USA, 1996.

[Mills *et al.* 92] Mills, M., Cohen, J. and Wong, Y-Y. A magnifier tool for video data. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '92)*, Monterey, CA, 3-7 May, 1992, 93-98.

[O'Connor *et al.* 01] O'Connor, N., Marlow, S., Murphy, N., Smeaton, A.F., Browne, P., Deasy, S., Lee, H. and Mc Donald, K. Físchlár: an on-line system for indexing and browsing of broadcast television content. *Proceedings of the 26th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2001),* Salt Lake City, UT, 7-11 May, 2001.

[Zhang *et al.* 95] Zhang, H., Low, C., Smoliar, S. and Wu, J. Video parsing, retrieval and browsing: an integrated and content-based solution. *Proceedings of 3rd ACM International Conference on Multimedia (MM '95)*, San Francisco, CA, 5-9 November, 1995, 503-512.

Word proximity QA system

Philip Rennert
EC Wise, Inc. Machine Learning Center
408 Saybrooke View Dr., Gaithersburg, MD 20877
phil.rennert@ioip.com

Abstract

This is a question answering system with very little NLP, based on question-category-dependent selection and weighting of search terms, selecting answer strings centered around words most commonly found near search terms. Its performance was medium, with a 0.2 MRR. Certain category strategies may be of interest to other QAers, and are described.

1.0 Question categories

To save bandwidth, I won't repeat the description of the QA task or the box-and-line description of the system; see almost any paper here. Here are the question categories, with performance:

Category	N	Average run MRR	My MRR
???	20	0.151	0.0625
ABBREVFOR	1	0.037	0
ABBREVIATION	5	0.214	0.8
CANNED	11	0.296	0.523
CAPITAL	5	0.512	0.7
FIRST TO	14	0.173	0.107
HOW MANY	4	0.174	0.5
HOW MANY AREA	0		
HOW MANY DIST	8	0.300	0.619
HOW MANY DOSAGE	0		
HOW MANY MASS	2	0.177	0
HOW MANY MONEY	0		
HOW MANY SPEED	3	0.150	0.333
HOW MANY STATED	5	0.244	0.4
HOW MANY TEMP	0		
HOW MANY TIME	5	0.217	0.067
INVENTION	6	0.280	0.367
PLACE WHERE GEO	21	0.216	0.099
PLACE WHERE ORIG	0		
PLACE WHERE PHYS	0		
POPULATION	8	0.298	0.604
STAR OF	0		
SUPERLATIVES	20	0.219	0.2
WHAT EAT	1	0.560	1.0
WHAT IS DESCR	154	0.172	0.134
WHAT IS NAMED	114	0.242	0.110
WHAT PAPER	2	0.217	0.25
WHAT SHOW	1	0.198	0
WHAT SPORT	1	0.291	0.5
WHEN BORN	6	0.264	0.25
WHEN DID	30	0.302	0.262
WHERE BORN	0		
WHERE IS	16	0.445	0.349
WHO DID	22	0.392	0.447
WHO IS	3	0.338	0.167
WHY	4	0.154	0

Average MRR: 0.234
Number of questions: 492

Question categories were defined from the training set of previous TRECs; some categories didn't occur in TREC 10.

2.0 Category strategies

The basic approach was to remove stopwords from the question and use all remaining words as search terms, though some were upweighted in some categories. The documents were viewed as an ordered list of words; occurrence of a search term awarded points to all words within a certain radius, typically five words. Points depended on the search term weight, and slightly on the distance. Stopwords and most punctuation were usually removed. Points were summed over all retrieved documents; answer strings were centered on the highest scoring words.
This was the default strategy, used in categories ???, WHAT IS DESCRIBED (a "What is" question with the object described but not named), and FIRST TO (Who was the first to...), and it wasn't very successful.
In certain categories, predefined lists of search terms or answer patterns produced better performance; these are described below.

ABBREVIATION - Look for the letters of the acronym in sequence at the head of words in sequence. Usually one letter per word (e.g., laser), but can be more (hazmat).

CANNED - Preloaded lists of US state and president information, and winners of World Series and Superbowls, downloaded from the Web. It's backwards to already know the answer and seek it in the test corpus; I included this category only because it's valid in a real QA system.

HOW MANY - Answer must be a number followed by a unit. Elaborate Perl regexp to identify a number; preloaded lists of units for various physical quantities (mass, distance, speed,...); map from question words to appropriate quantity. For some questions, unit is stated in question (e.g., How many dogs pull a sled in the Iditarod?; unit is "dogs"). Seek number-unit string near search terms.

INVENTION - Answer is a proper name, so will be Initcapped. Seek search terms, word which stems to "invent" or "patent", and Initcap string close together.

POPULATION - Answer is a number; seek search terms, "population" or "people", and number close together. As a tiebreak, pick the larger number; usually news stories describe a population and a subset.

WHAT IS NAMED (or Definition) - Unfortunate excess of these questions in TREC 10 has been described elsewhere. Postfit strategy, after TREC 10 judging: if word has a WordNet gloss, extract the first two nouns before the semicolon and add to search terms (if more than one sense, do it for all). This improved MRR to .38 in this category. Another backwards already-know-the-answer category; a mismatch with the news story corpus.

WHEN DID - Answer is a date; Perl regexp to define dates; seek them near search terms. If question starts "When is...", append date from story timeline, to cover holidays and such.

WHERE IS - Preloaded list of proximity terms used with Initcap places, like "near", "neighboring", "border", etc.

WHO DID - Answer is a proper name, so Initcapped. Stem action verbs in question (who built, who killed, ...). In counting most common Initcaps near search terms, combine subset terms (e.g., "Fred Jones" and "Jones").

WHO IS - Preloaded list of occupations/activities to make this person famous (writer, leader, winner, etc.); seek the occupation most commonly found near the name. Stem the occupation words (writer = written = writing, etc.).

3.0 Document retrieval strategy comparison

I wrote my own search engine (mergesort-based), to be able to retrieve documents based on my search terms and weights. I submitted two runs, one with only documents from the PRISE top 50, one with those plus the top 50 from my strategy, interleaved to make a list of 100. The results were statistically no different. I conclude the PRISE top 50 document set is good enough to support the answer-finding strategies used here.

4.0 What's next?

With the advance to exact answers in future TRECs, answer patterns will become essential. I believe question parsing and question term expansion to generate them will also become essential, and proximity most-common strategies will not be effective. Some of the answer patterns used here may remain useful.

FUB at TREC-10 Web Track: A probabilistic framework for topic relevance term weighting

Gianni Amati * Claudio Carpineto
Giovanni Romano
Fondazione Ugo Bordoni, Roma, Italy

1 Introduction

The main investigation of our participation in the WEB track of TREC-10 concerns the effectiveness of a novel probabilistic framework [1] for generating term weighting models of topic relevance retrieval. This approach endeavours to determine the weight of a word within a document in a purely theoretic way as a combination of different probability distributions, with the goal of reducing as much as possible the number of parameters which must be learned and tuned from relevance assessments on training test collections.

The framework is based on discounting the probability of terms in the whole collection, modeled as deviation from randomness, with a notion of information gain related to the probabilty of terms in single documents. The framework is made up of three components: the "information content" component relative to the entire data collection, the "information gain normalization factor" component relative to a subset of the data collection (the elite set of the observed term), and the "term frequency normalization function" component relative to the document length and to other collection statistics. Each component is obtained by computing a suitable probability density function.

One advantage of the framework is that we may easily compare and test the behaviour of different basic models of Information Retrieval under the same experimental conditions and normalization factors and functions. At the same time, we may test and compare different term frequency normalization factors and functions.

In addition to testing the effectiveness of the term weighting framework, we were interested in evaluating the utility of query expansion on the WT10g collection. We used information theoretic query expansion and focused on careful paremeter selection.

In our experiments, we did not use link information, partly because of tight scheduling

*The author has been carrying out his work also at the Computing Science Department, University of Glasgow, Scotland

- the WT10g collection was made available to us as late as late May 2001 - and partly because it has been shown at TREC-9 that it is not beneficial to topic relevance retrieval.

In the rest of the paper, we first introduce the term weighting framework. Then we describe how collection indexing was performed, which probability functions were used in the experiments to instantiate the term weighting framework, and which parameters were chosen for query expansion. A discussion of the main results concludes the paper.

2 The term weighting framework

The framework presented here can be found in [1].

The fundamental weighting Formula is the product of two *information content* functions:

$$w = \mathit{Inf}_1 \cdot \mathit{Inf}_2 \tag{1}$$

Both Inf_1 and Inf_2 are decreasing functions of two probabilities P_1 and P_2, respectively. The function Inf_1 is related to the whole document collection D, whilst Inf_2 to the elite set E_t (this notion roots back to Harter's work [5]) of the term t, namely the set of all documents in which the term t occurs.

P_1 is obtained as follows. We assume that words which bring little information are randomly distributed on the whole set of documents. By contrast, informative words diverge from the randomic behaviour and therefore they receive little probability according to a suitable model of randomness for Information Retrieval. This is the *the inverse document frequency "component"* of our model, in the sense that similar to the standard IR models based on the idf measure, the informative words have a small probability to occur within a document. We provide different basic models which defines such a notion of *randomness in the context of Information Retrieval.* A model of randomness is derived by a suitable interpretation of the probabilistic urn models of Types I and II [4] into the context of Information Retrieval. Basically, a model of Type I is a model where balls (tokens) are randomly extracted from an urn, whilst in Type II models balls are randomly extracted from an urn belonging to a collection of urns (documents). Among type I models there is the Poisson model, the Bose-Einstein statistics, the Geometric distribution, whilst a type II model is the inverse document frequency model. Therefore, the frequency of a word within a document with the lowest probability P_1 as predicted by such models of randomness or, equivalently, the words whose probability is less expected by the chosen model of urns, are "highly informative" words.

$$\mathit{Inf}_1 = -\log P_1 \quad \texttt{large for informative words in the collection}$$

If we observe the elite set E_t of the term, then we may derive a second conditional probability P_2 of term occurrence within a document in its elite set. The information content of a highly informative term, as obtained by means of Inf_1, will be tuned according to its elite set.

This weight tuning process corresponds to *the information gain "component"* of our model. *We will take as weighting formula only a fraction Inf_2 of Inf_1.* This fraction of the information content corresponds to the "gain" associated to the decision of accepting the term as an informative descriptor of the document. We assume, as in decision theory, that this information gain and thus inf_2 is inversely related to its odds P_2.

Differently from P_1, which is in general very small, P_2 should be in general close to certainty, especially when tf is large. If we observe many occurrences of the term t, then the observed term should have a very high probability P_2 of being a descriptor of the document. We assume that P_2 is the conditional probability of observing, within an *arbitrary document* of the elite set, $tf + 1$ occurrences of a given word in the hypothesis that one has already observed tf occurrences. The higher the term frequency tf, the higher the conditional P_2. Since the gain is inversely related to its odds:[1]

$$Inf_2 = 1 - P_2 \qquad \texttt{rate of the information content gained with } t$$

The weight of a term in a document is thus a function of two probabilities P_1 and P_2 which are related by the following relation:

$$w \;= Inf_1 \cdot Inf_2 \;= (-\log_2 P_1) \cdot (1 - P_2) \tag{2}$$

The term weight w of Formula 2 can be seen as a function of 4 random variables:

$$w = w(F, tf, n, N)$$

where
tf is the within document term frequency
N is the size of the collection
n is the size of the elite set E_t of the term,
F is the term frequency in its elite set

However, the size of tf depends on the document length: we have to derive the expected term frequency in a document when the document is compared to a fixed length (typically the average document length). We should determine what is the distribution that the tokens of a term follow in the documents of a collection at different document lengths. Once this distribution is obtained, the normalized term frequency tfn is used in the Formula 2 instead of the non-normalized tf.

One formula we have formally derived and successfully tested on previous TREC collections is:

$$tfn = \; tf \cdot \log_2\left(1 + \frac{c \cdot avg\_l}{l}\right) \quad \texttt{(with } c \geq 1\texttt{)} \tag{3}$$

[1] We also used an alternative monotone decreasing function, namely $Inf_2 = -\log_2 P_2$. Experimentally, this decreasing function seems to be a little less effective than $Inf_2 = 1 - P_2$. Also, the function $1 - P_2$ will be easily generalized below to the increment rate of two Bernoulli's trials, whilst a similar generalization with $Inf_2 = -\log_2 P_2$ is problematic.

where avg_l and l are the average length of the document collection and the length of the observed document respectively.

Our term weight w of Formula 2 will be thus a function of 6 random variables:

$$w = w(F, tfn, n, N) = w(F, tf, n, N, l, avg\_l)$$

where l is the document length
avg_l is the length mean

We postpone the discussion about the probability functions used to instantiate this framework and the choice of parameter c to Section 4.2. We first describe, in the next section, how collection indexing was performed.

3 Test collection indexing

Text segmentation. Our system first identified the individual terms occurring in the test collection, ignoring punctuation and case. The whole body of each document was indexed except for HTML tags, which were removed from documents. *Pure single keyword indexing was performed, and link information was not used.*

Document pruning. As we had very limited storage capabilities, we performed some document pruning. We removed very long or short documents as well as documents which were deemed to be nontextual or nonenglish textual. Specifically, we pruned the documents with more than 10,000 words (2,897) or less than 10 words (57,031); also, we removed the documents that contained more than 50% of unrecognized English word (86,146), according to a large morphological lexicon for English (Karp et al, 1992). In all, we removed 118.087 documents (this is not the exact sum of the three categories due to document overlap). The price we paid for this computational gain is that some relevant documents were lost. More exactly, we removed 162 out of 3363 relevant documents (4.81%). Thus, it should be emphasized that our actual performance retrieval was probably lower than the performance that we would have obtained by considering the whole set of documents.

Word pruning. Incorrect words affect collection statistics and query expansion. In order to reduce the inherent web word noise, we removed very rare, ill-formed or exceedingly long words. Specifically, the words contained in no more than 10 documents, which were apparently exclusively mispelled words, were dropped from the document descriptions. The words containing more than three consecutive equal characters or longer than 20 characters were also deleted. In this way, the number of distint words in the collection decreased dramatically, from 1,602,447 (after steps 1 and 2) to only 293,484.

Stop wording and word stemming. As we were primarily interested in early precision, we used a very limited stop list and did not peform word stemming at all.

The system has been implemented in ESL, a Lisp-like language that is automatically translated into ANSI C and then compiled by gcc compiler. The system indexes two gigabytes of documents per hour and allows sub-seconds searches on a 550 MHz Pentium III with 256 megabytes of RAM running Linux.

4 Term weighting models

The term-weighting framework described above was instantiated to a number of models using the Bose-Eistein statistics and the inverse document frequency (expected and non) combined with the weight normalization factor Inf_2 and frequency normalization function tfn. We first describe the basic models and then the 2 normalization factors L and B for Inf_2.

Bose-Einstein statistics

The operational model of the Bose-Einstein statistics is constructed by approximating the factorials by Stirling's formula. The model B_E (B_E stands for Bose-Einstein) is:

$$Inf_1(tf) = \log_2 \frac{(N+F-tf-2)!F!(N-1)}{(F-tf)!(N+F-1)!} \tag{4}$$

Let $\lambda = \frac{F}{N}$ be the mean of the frequency of the term t in the collection D, then the Bose-Einstein probability that a term occurs tf times in a document can be approximated by $P_1(tf) = \left(\frac{1}{1+\lambda}\right) \cdot \left(\frac{\lambda}{1+\lambda}\right)^{tf}$. The right hand side is known as the *geometric distribution* with probability $p = \dfrac{1}{1+\lambda}$. Hence:

$$Inf_1(tf) = -\log_2 \left(\frac{1}{1+\lambda}\right) - tf \cdot \log_2 \left(\frac{\lambda}{1+\lambda}\right) \tag{5}$$

The approximations of Equation 4 by Stirling's formula and by Equation 5 were indistinguishable in the experiments, therefore Equation 5 is preferred to Equation 4 for its simplicity.

The inverse document frequency model $I(n)$

We use a standard tf-idf probability distribution. The probability $P_1(tf)$ is obtained by first computing the probability of choosing a document containing the given term at random and then computing the probability of having tf occurrences of the same term in a document:

$$Inf_1(tf) = tf \cdot \log_2 \frac{N+1}{n+0.5} \tag{6}$$

The inverse expected document frequency model $I(n_{exp})$

A different model can be obtained by Bernoulli's law. Let n_{exp} the expected number of documents containing the term under the assumption that there are F tokens in the collection. Then

$$n_{exp} = N \cdot Prob(tf \neq 0) = N \cdot (1 - B(N, F, 0)) = N \cdot (1 - \left(\frac{N-1}{N}\right)^{F})$$

The third basic model is the tf-Expected_idf model $I(n_{exp})$:

$$Inf(tf) = tf \cdot \log_2 \frac{N+1}{n_{exp} + 0.5} \tag{7}$$

4.1 Term frequency normalizations: the probability P_2

We assume that the probability that an observed term contributes to select a relevant document is high if the probability of counting one more token of the same term in a relevant document is similarly high. This probability approaches 1 for high values of tf.

Laplace's normalization L

The first model of $P_2(tf)$ is obtained by the conditional probability $p(tf + 1|tf, d)$ of Laplace's Law of Succession: $P_2(tf) = \frac{tf}{tf+1}$

The normalization L (for Laplace) is:

$$Inf_2 = \frac{1}{tf+1} \tag{8}$$

Bernoulli's normalization B

To obtain an alternative estimate of P_2 with Bernoulli's trials we use the following urn model. Let $B(n, F, tf)$ be

$$B(n, F, tf) \quad = \binom{F}{tf} p^{tf} q^{F-tf}$$

where $p = \frac{1}{n}$ and $q = \frac{n-1}{n}$.

We add a new token of the term to the collection, thus having $F + 1$ tokens instead of F. We then compute the probability $B(n, F + 1, tf + 1)$ that this new token falls into the observed document, thus having a within document term frequency $tf + 1$ instead tf. The process $B(n, F + 1, tf + 1)$ computes the probability of obtaining one more token of the term t in the document d out of all n documents in which t occurs when a new token is added to the elite set. The ratio

$$\frac{B(n, F+1, tf+1)}{B(n, F, tf)} = \frac{F+1}{n \cdot (tf+1)}$$

of the new probability $B(n, F+1, tf+1)$ to the previous one $B(n, F, tf)$ tells us whether the probability of encountering a new occurrence by chance is increased or diminished.

Instead of using P_2 we normalize with the probability increment rate

$$\mathit{IncrementRate} = 1 - \frac{B(n, F+1, tf+1)}{B(n, F, tf)}$$

that is the normalization B is:

$$\mathit{Inf}_2(tf) = \frac{F+1}{n \cdot (tf+1)} \tag{9}$$

4.2 The parameter c for the baseline models

Independently from the model used, namely independently from the probability distributions P_1 and P_2 chosen, in TREC-9 and TREC-10 the best matching value for c was 7. The parameter c seems to be proportional to the size of the collection and inversely proportional to the size of the indexing vocabulary. A similar observation held also for the TREC-1 to TREC-8 collections.

We conjecture that the parameter c is connected to the Zipfian law which relates the size of vocabulary to the size of the collection. This relationships which is not linear affects the size of term frequency in the collection and thus the term frequency in the document.

5 Query expansion

For TREC-9, the results about the use of query expansion were not as good as with previous TRECs. Several groups reported that expansion did not improve or even hurt retrieval performance [6]. As groups participating in TREC-9 web track had little opportunity for parameter tuning and the WT10g collection is very different from the previous collections, these result may have been influenced by poor choice of query expansion parameters.

We encountered a similar problem with our own information theoretic-based expansion method [3, 2]. The weight of a term of the expanded query q^* of the original query q is obtained as follows:

$$\mathit{weight}(t \in q^*) = (\alpha \cdot tfq_n + \beta \cdot tfn_{KL}) \cdot \mathit{Inf}_1 \cdot \mathit{Inf}_2$$

where

- tfq_n is the normalized term frequency within the original query q (i.e. $\frac{tfq}{max_{t \in q} tfq}$)
- tfn_{KL} is a term frequency in the expanded query induced by using a normalized Kullback-Leibler measure

  $$tfn_{KL} = \frac{tf_{KL}}{\max\limits_{t \in q^*} tf_{KL}} \qquad (10)$$

  $$tf_{KL} = P_R(t) \cdot log \frac{P_R(t)}{P_C(t)}$$

  where $P_X(t)$, with $X = R, C$, is the probability of occurrence of term t in the set of documents X (estimated by the relative frequency of the term in X), R indicates the pseudo-relevant set, C indicates the whole collection.
- $\alpha = 1$, $\beta = 0.2$
- $|R| = 3$ with the number of terms of the expanded query equal to 10.
- Inf_1 and inf_2 as defined in Relation 1

This method was used with good results on TREC-8; however, when we ran it with the TREC-8 parameters against the TREC-9 collection, the retrieval performance was badly affected, whether using the new weighting functions discussed above or the Okapi formula. Thus, we focused on better selection of the values used for query expansion parameters for the WT10g document set, by performing parameter tuning on the TREC-9 test collection. We considered three parameters, namely the number of pseudo relevant documents, the number of expansion terms and the ratio between α and β in Rocchio's formula.

One of the most striking characteristics of the WT10g collection is that the quality of baseline retrieval is lower than that obtained for past TREC collections. In an attempt to reduce the chance to select terms from mostly nonrelevant documents we chose fewer pseudo-relevant documents than typically used for query expansion. We set the number of pseudo-relevant documents at 3. In order to compensate for the lower quality of the terms used for expansion, we also adjusted the values of α and β. Since the original query should become more important as the quality of the expansion terms and their weights diminishes, we set the ratio between α and β to 5 (i.e., $\alpha = 1$, $\beta = 0.2$) and reduced the number of terms used for query expansion to 10. In this way, it should be easier for the expanded query to keep the focus on the original topic, even in the presence of bad term suggestions.

Finally, it should be noted that the removal of bad words performed at indexing time (see discussion above) may have considerably reduced the number of typographical errors in documents, which was pointed out as one of the causes for poor query expansion.

6 Runs at TREC 10

We submitted 4 runs, 2 of them with our query expansion technique.

Runs fub01ne and fub01ne2: $I(n_{exp})L$

The baseline model fub01ne for Inf_1 is $I(n_{exp})$ and the normalization formula for Inf_2 is Laplace's law L namely the term weight is:

$$w = \frac{1}{tfn+1} \cdot \left(tfn \cdot \log_2 \frac{N+1}{n_{exp}+0.5} \right) \tag{11}$$

tfn is defined as in Equation 3 with $c = 7$. Run fub01ne was performed without query expansion, whilst run fub01ne2 with.

Run fub01be2: $B_E L$. This was the best performing run at TREC-10. The baseline model fub01be for Inf_1 is B_E and the normalization formula for Inf_2 is Laplace's law L namely the term weight is:

$$w = \frac{1}{tfn+1} \cdot \left(-\log_2 \left(\frac{1}{1+\lambda} \right) - tfn \cdot \log_2 \left(\frac{\lambda}{1+\lambda} \right) \right) \tag{12}$$

tfn is defined as in Equation 3 with $c = 7$. The automatic query expansion was performed.

Run fub01idf: $I(n)B$

The baseline model fub01idf for Inf_1 is $I(n)$ and the normalization formula for Inf_2 is Bernoulli's rate B namely the term weight is:

$$w = \frac{F+1}{n(tfn+1)} \cdot tfn \log_2 \frac{N+1}{n+0.5} \tag{13}$$

tfn is defined as in Equation 3 with $c = 7$. The automatic query expansion was not performed.

7 Results and conclusions

In Table 1 we show the retrieval performance of all possible models that can be generated by the term weighting framework using the probability functions introduced above, without and with query expansion.

The main conclusions that can be drawn from the experimental results are the following.

- On the whole, the term weighting framework was effective, with very good absolute and comparative retrieval performance (run **fub01be2** achieved the best performance of all official submissions in the title-only, automatic topic relevance task),

Method	Official run	AvPrec	Prec-at-10	Prec-at-20	Prec-at-30
Model performance without query expansion					
$B_E L$		0.1788	0.3180	0.2730	0.2413
$I(n)L$		0.1725	0.3180	0.2740	0.2353
$I(n_{exp})L$	fub01ne	0.1790	0.3240	0.2720	0.2440
$B_E B$		0.1881	0.3280	0.2980	0.2487
$I(n)B$	fub01idf	0.1900	0.3360	0.2880	0.2580
$I(n_{exp})B$		0.1902	0.3340	0.2860	0.2580
Model performance with query expansion					
$B_E L$	fub01be2	0.2225	0.3440	0.2860	0.2513
$I(n)L$		0.1973	0.3200	0.2730	0.2380
$I(n_{exp})L$	fub01ne2	0.1962	0.3280	0.2760	0.2507
$B_E B$		0.2152	0.3400	0.2870	0.2527
$I(n)B$		0.2052	0.3380	0.2970	0.2680
$I(n_{exp})B$		0.2041	0.3360	0.2990	0.2660

Table 1: Comparison of performance of models and normalization factors.

although noteworthy differences in performance were observed depending on which combination of probabilistic distributions and normalization techniques was used.

- Query expansion with the chosen parameters improved performance for almost all term weighting models and evaluation measures, with more tangible benefits for average precision.

More work is necessary to investigate the relative strengths and weaknesses of each model as well as to study the relationships to other term weighting approaches. Moreover, further experiments should be performed to control the effect on performance of a wider range of factors, including word stemming, document pruning, and word pruning.

References

[1] Gianni Amati and Cornelis Joost van Rijsbergen. Probabilistic models of information retrieval based on measuring divergence from randomness. *Manuscript*, 2001.

[2] C. Carpineto, R. De Mori, G. Romano, and B. Bigi. An information theoretic approach to automatic query expansion. *ACM Transactions on Information Systems*, 19(1):1–27, 2001.

[3] C. Carpineto and G. Romano. Trec-8 automatic ad-hoc experiments at fub. In *In Proceedings of the 8th Text REtrieval Conference (TREC-8), NIST Special Publication 500-246*, pages 377–380, Gaithersburg, MD, 2000.

[4] Irving John Good. *The Estimation of Probabilities: an Essay on Modern Bayesian Methods*, volume 30. The M.I.T. Press, Cambrige, Massachusetts, 1968.

[5] Stephen Paul Harter. A probabilistic approach to automatic keyword indexing. part I: On the distribution of specialty words words in a technical literature. *Journal of the ASIS*, 26:197–216, 1975.

[6] D Hawking. Overview of the trec-9 web track. In *In Proceedings of the 9th Text Retrieval Conference (TREC-9)*, Gaithersburg, MD, 2001.

FDU at TREC-10: Filtering, QA, Web and Video Tasks

Lide Wu, Xuanjing Huang, Junyu Niu, Yikun Guo, Yingju Xia, Zhe Feng
Fudan University, Shanghai, China

This year Fudan University takes part in the TREC conference for the second time. We have participated in four tracks of Filtering, Q&A, Web and Video.

For filtering, we participate in the sub-task of adaptive and batch filtering. Vector representation and computation are heavily applied in filtering procedure. Four runs have been submitted, which includes one T10SU and one T10F run for adpative filtering, as well as another one T10SU and one T10F run for batch filtering.

We have tried many natural language processing techniques in our QA system, including statistical sentence breaking, POS tagging, parsing, name entity tagging, chunking and semantic verification. Various sources of world knowledge are also incorporated, such as WordNet and geographic information.

For web retrieval, relevant document set is first created by an extended Boolean retrieval engine, and then reordered according to link information. Four runs with different combination of topic coverage and link information are submitted.

On video track, We take part in both of the sub-tasks. In the task of shot boundary detection, we have submitted two runs with different parameters. In the task of video retrieval, we have submitted the results of 17 topics among all the topics.

1. Filtering

Our research on filtering focuses on how to create the initial filtering profile and set the initial threshold, and then modify them adaptively. In this section, detailed introduction to the training and adaptation module of our adaptive runs is first presented. Then we introduced our batch runs briefly. Final part presents the experiment results.

1.1 Adaptive filtering

Figure 1.1 shows the architecture of the training in adaptive filtering. At first, feature vectors are extracted from positive and pseudo-positive document samples. The initial profile is the weighted sum of positive and pseudo-positive feature vectors. Then we compute the similarity between the initial profile and all the training documents to find the optimal initial threshold for every topic.

1.1.1 Feature selection

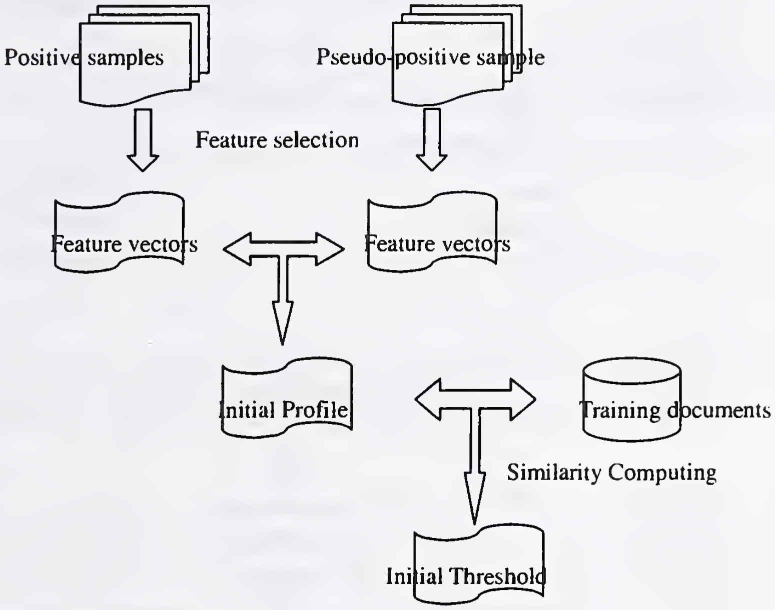
Since the total number of all words is very large and it costs much time in similarity computation, we decide to select some important words from them. First, we carry out morphological analysis and stopword removing. Then we compute the *logarithm Mutual Information* between remaining words and topics:

$$\log MI(w_i, T_j) = \log\left(P(w_i \mid T_j) \big/ P(w_i) \right) \qquad (1.1)$$

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.

Figure 1.1 Architecture of the training in adaptive filtering



1.1.2 Similarity Computation

The similarity between the profile and training documents is computed by the cosine formula:

$$Sim(d_i, p_j) = Cos\theta = \sum_k d_{ik} * p_{jk} \Big/ \sqrt{(\sum_k d_{ik}^2)(\sum_k p_{jk}^2)} . \quad (1.2)$$

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.

1.1.3 Creating initial profile and Setting initial threshold

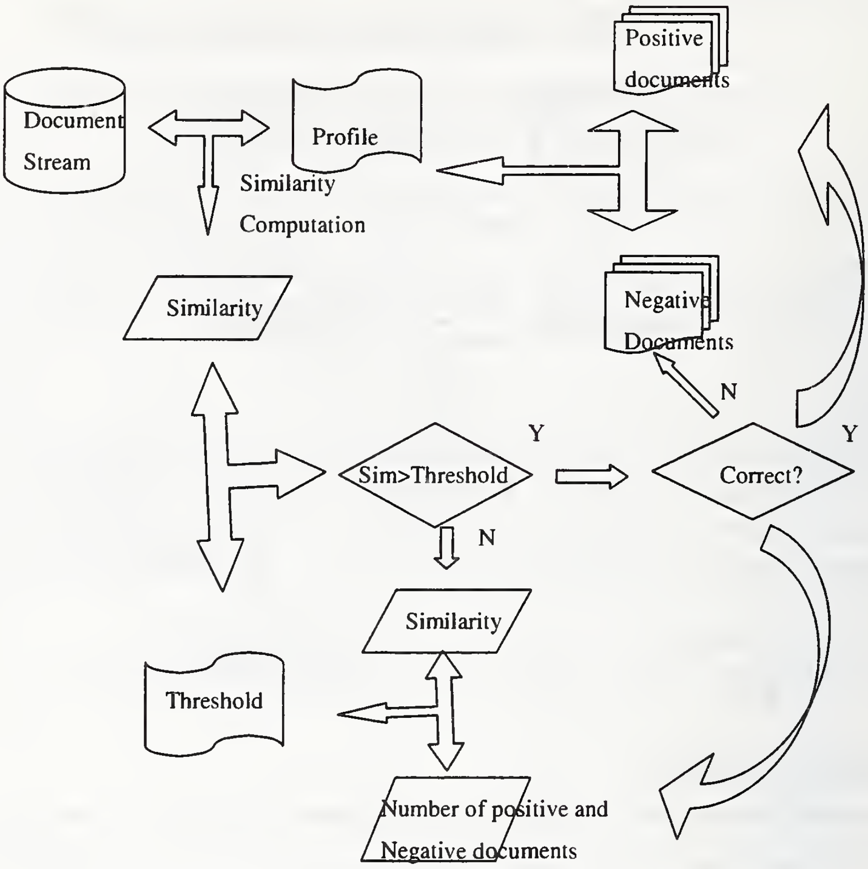
Each topic profile is represented by a vector which is the weighted sum of feature vector from positive (relevant) documents and feature vector from pseudo relevant documents with the ratio of 1: X_0.

To make use of the hierarchy of categories, those documents of the same high-level category are considered as pseudo relevant documents. Since the number of low-level categories is different among different high-level categories, we set different X_0 for different categories. In our experiment, X_0 is set to 0.03 for the topics from R1 to R79, and 0.15 for R80~R84. After combining the positive and pseudo-positive feature vectors, we get the initial profile. Once the initial profiles are acquired, the initial thresholds should be set to those values that can result in the largest value of T10U or T10F.

1.1.4 Adaptation of threshold and topic profile during filtering

For adaptive filtering, we adopt an adaptation procedure to modify the initial profile and threshold while filtering documents. Figure 1.2 shows the architecture for the adaptation:

Figure 1.2 Architecture for the adaptation



(1) Adjustment of threshold

We adjust the threshold once a positive document is retrieved. Let:

- t: denote the sequence number of document, since the documents are processed by temporal order, t also can be considered as time.
- $n(t)$: denote the number of documents processed up to t
- $n_R(t)$: denote the relevant documents retrieved up to t
- $n_N(t)$: denote the irrelevant documents retrieved up to t
- $T(t)$: denote the threshold at time t
- $\bar{S}(t_k, t_{k+1})$: denote the average similarity of the document been rejected in (t_k , t_{k+1}) interval
- $P(t_k, t_{k+1})$: denote the precision of system in (t_k , t_{k+1}) interval, here

$$P(t_k, t_{k+1}) = {}^{n_R(t_{k+1}) - n_R(t_k)}\Big/_{n(t_{k+1}) - n(t_k)}$$

Intuitionally, we should increase the threshold if the precision is too low and lower the threshold if too few documents are retrieved. So we can use $\bar{S}(t_{k+1}, t_k)$ and $P(t_{k+1}, t_k)$ to decide whether to increase or decrease the threshold. When the precision is lower than expected, we should increase the threshold. Otherwise, we can decrease the threshold. In particular, when the threshold is too higher than the similarity with the rejected documents, the threshold should be decreased quickly. The above strategy of threshold adjusting can be written as below:

If $p(t_k, t_{k+1}) \le EP(t_{k+1})$ then

$$T(t_{k+1}) = T(t_k) + \alpha(t_{k+1}) \bullet (1 - T(t_k))$$

Else If $S^-(t_k, t_{k+1}) < T(t_{k+1}) * D$ then

$$T(t_{k+1}) = T(t_k) \bullet A + S^-(t_k, t_{k+1}) \bullet (1 - A)$$

Else $T(t_{k+1}) = (1 - \beta(t_{k+1})) * T(t_k)$

Where $\alpha(t_k)$ is the coefficient for increasing the threshold, and $\beta(t_k)$ is the coefficient for decreasing the threshold, both of them can be considered as the function of $n_R(t)$. In our experiment, we use the following linear functions shown in equation 1.3.

$$\alpha(t_k) = \begin{cases} \alpha_0 * (\mu - n_R(t_k))/\mu, & n_R(t_k) \le \mu \\ 0, & n_R(t_k) > \mu \end{cases}, \beta(t_k) = \begin{cases} \beta_0 * (\mu - n_R(t_k))/\mu, & n_R(t_k) \le \mu \\ 0, & n_R(t_k) > \mu \end{cases} \quad (1.3)$$

Where α_0 and β_0 are the initial parameter. The parameter of μ indicates the maximum number of positive documents should be used to adjust the threshold and modify the profile. Here we set $\alpha_0 = 0.02$, $\beta_0 = 0.1$ and $\mu = 300$.

The introduction of parameter D aims at increasing the recall. Since the actual number of relevant documents of every topic cannot be observed, we can only acquire some indirect estimation. We believed when the average similarity between the profile and those rejected documents are too small, the similarity threshold should be decreased in order to enhance the recall. In our experiment, we set D = 0.1 and A = 0.8.

$EP(t_k)$ means the precision which we wish the system to reach. At first, we regarded this parameter as constant and tried several different values, but the results are not very satisfactory. Since it is impractical to require the system to reach the desired high precision at the beginning of filtering, we adopt a gradual-ascent function. The function is showed in equation 1.4.

$$EP(t_{k+1}) = \begin{cases} P_0 + (P_{final} - P_0) * n_R(t_{k+1})/\mu, & n_R(t_k) \le \mu \\ 0, & n_R(t_k) > \mu \end{cases} \quad (1.4)$$

Where, P_0 and P_{final} are the desired initial and final precision. In our experiment, $P_0 = 0.2$ and $P_{final} = 0.6$.

(2) Adaptation of profile

Once a retrieved document has been judged relevant, it is added to the positive document set otherwise it is added to the negative document set. During profile adaptation, feature vectors are extracted from positive documents and negative documents. The new topic profile is the weighted sum of feature vector from positive documents and negative documents with the ratio of $1:X_1$ (Here X_1= -0.25). For effectiveness and efficiency reason, we adjust the topic profile only after $L(L = 5)$ positive documents have been retrieved.

1.2 Batch filtering

Since this year's batch filtering task does not include batch-adaptive task, there should be no adaptation in the batch-filtering sub-task. Therefore, the profile and threshold acquired from training should remain the same during filtering.

There is only a slight difference in the initialization module of our batch and adaptive runs. Full relevance judgments are provided in batch filtering. As a result, for batch run, the given relevant judgments are enough for us to build the initial profile, so pseudo-relevant documents are not used in profile creation. In addition, we adopt the stratified tenfold cross-validation method to avoid the phenomenon of overfitting.

1.3 Evaluation results

This year Fudan University has submitted four runs for adaptive and batch filtering. We submit no routing runs. Table 1.1 summarizes our adaptive and batch filtering runs. Four evaluation criteria are calculated, including *T10SU*, *T10F*, *Set Precision* and *Set Recall*. Underlined value means that the run is optimized for the corresponding criterion. The last columns give the number of topics in which our runs perform better, equal and worse than median ones according to the criteria for which our runs are optimized.

Task	*Run*	*T10SU*	*T10F*	*Set Precision*	*Set Recall*	*Comparison with median*		
						>	=	<
Adaptive	FDUT10AF1	<u>0.215</u>	0.404	0.505	0.330	64	5	15
	FDUT10AF4	0.213	<u>0.414</u>	0.493	0.363	71	4	10
Batch	FDUT10BF1	<u>0.248</u>	0.441	0.563	0.313	32	13	39
	FDUT10BF2	0.244	<u>0.448</u>	0.526	0.373	27	17	40

Table 1.1 Adaptive and batch filtering results

From this table we can find that our adaptive runs perform better than median for most of the topics, while our batch runs do not perform as well. Although our batch runs performs better than adaptive runs, the divergence is not very significant. It helps to show that adaptation plays a very important role in filtering.

2. Question Answering

It is the second time that we take part in the QA track. We tried many natural language processing techniques, and incorporated many sources of world-knowledge. A novel question answering technique, known as "*syntactic constrained semantic verification*", has been put forward. In next section, we will describe the architecture of our QA system, followed by a detailed discussion of the main components.

2.1 The Overview of QA system

Our system contains four major modules, namely question processing module, offline indexing module, online searching and concept filtering module, as well as answer processing module. The online models are represented in Figure 2.1.

Our indexing module creates full-text index for the document collection. However, it is quite different from traditional indexing procedure in that it incorporates several NLP techniques not only to avoid errors due to traditional stemming process, but also to increase both the precision and recall while retrieving proper name.

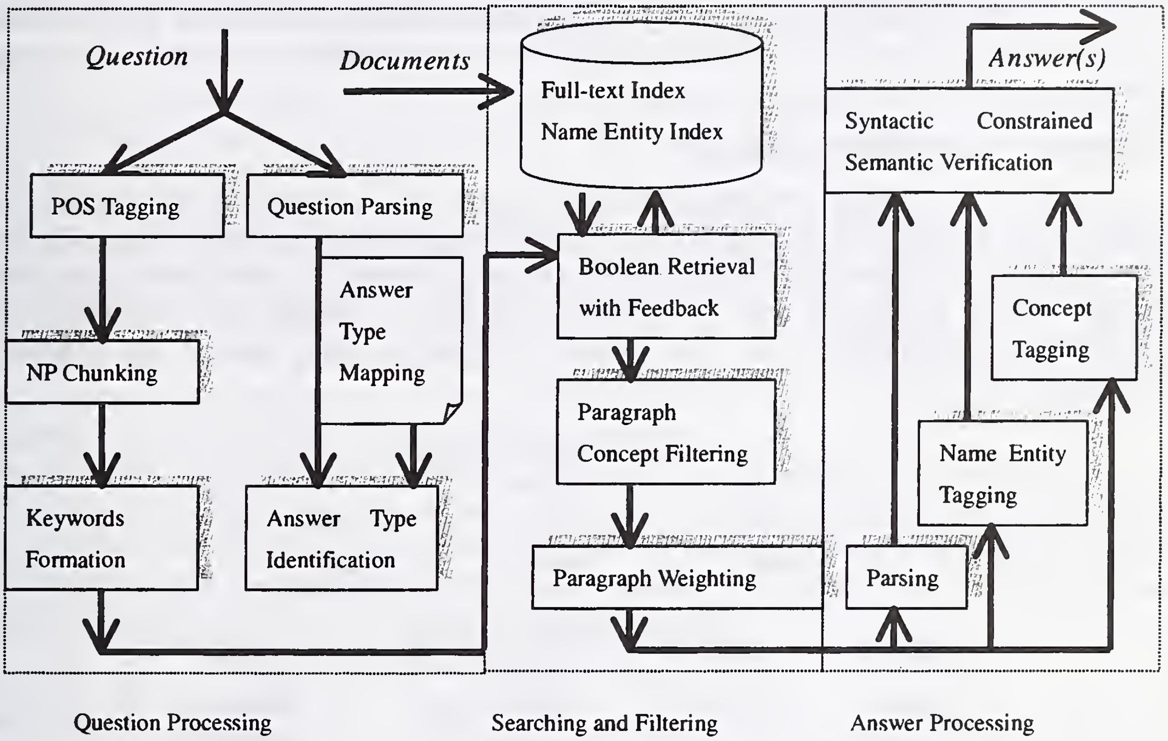
Question processing module tries to interpret the meaning of the input question by identifying answer type (the kind of information the question requires) from the question type, and extracting keywords. Next, the searching and filtering module use only non-replaceable keywords to retrieve relevant paragraph. After obtaining the result paragraphs, we use a concept thesaurus to filter and rank those paragraphs according to the number of occurring concepts, which are mainly derived from those replaceable keywords.

In the Answer Processing module, we use a dependency parser to analyze sentences in which the answer may lie in. Finally, a novel semantic verification scheme is applied after a WordNet-based concept tagging and a name entity tagging are completed.

2.2 Document Indexing

Our document indexing module actually includes two separate indices, i.e. a morphological analysis based full-text index and a proper name index. However, before we build these indexes, a sentence breaking module is applied to get correct sentence boundaries. We use a free sentence breaking tool, based on maximum entropy model, from Adwait Ratnaparkhi's web site.

Figure 2.1 Overview of our QA system (online part)



Proper name indexing is carried out to accelerate the online question processing speed. Our proper name tagging module depends heavily on a maximum entropy model based NP chunker. After reading each sentence with its part of speech tag (POS) for every word, it outputs NP chunks for the sentence. Figure 2.2 presents one sample sentence after NP chunking.

Figure 2.2 The output of NP chunking toolkit

[Ed Wilson] , [spokesman] for [the District] of [Columbia police] , said [street crime] in [Washington] has increased in [recent years] , but [there] have been [few reports] of [assaults] near [the Capitol grounds.]

2.3 Question Processing

The goal for question processing module is to find the user's information needs by examining the question. The query and expected answer type are transformed from every original natural language question.

2.3.1 Query Formation

First, considering synonyms, we define two kinds of keywords, i.e. the replaceable keywords and non-replaceable keywords. The replaceable keywords are referred to those words that could be replaced by other synonym without altering the information request of the question. Only the non-replaceable keywords are transformed into query. The documents returned by search engine will consist of all the candidate segments. Further, those candidates irrelevant to the question will be filtered out by replaceable keywords and their synonyms.

POS tagging and NP chunking are carried out to segment each question into segments. After that, we apply a HMM based Name Entity Identification tool to extract the non-replaceable keywords. It can recognize six kinds entity name, including people's name, place name, organization name, time, date and miscellaneous number, from normal NP phrases.

The NP phrases identified by Name Entity Identification module are regarded as the non-replaceable keywords and then submitted to the search engine, while other components are treated as the replaceable keywords.

2.3.2 Answer Type Concept Identification

Another task for Question Processing module is to determine the desired answer type concepts. First, we roughly classify 10 question types according to the question interrogatives, as shown in Figure 2.3. Next, 32 answer type concepts are introduced into our system, illustrated in figure 2.4. Among them, six are identified by Name Entity Identification tool, i.e. DATE, LOCATION, MONEY_UNIT, ORGANIZATION, PERCENTAGE and PERSON, while other concepts correspond to some synset in the WordNet noun hierarchies.

Figure 2.3 Question types

What	Who/whom	Where
How	why	how much
When	which	Name

Figure 2.4 Answer type concepts

ACTIVITY	COLOR	LINEAR_AMOUNT	PLANT
AMOUNT	COUNTRY	LOCATION	PRODUCT
ANIMAL	DATE	MONEY	PROVINCE
ARTIFACT	DEF	MONEY_UNIT	STAR
BODYPART	DISEASE	ORGANIZATION	TEMPERATURE
CAREER	ELEMENT	OTHER	TIME
CD	FOOD	PERCENTAGE	WEIGHT
CITY	LANGUAGE	PERSON	WORD

2.4 Search and Filtering Module

We employ the Boolean retrieval engine to find candidate answer paragraphs. We modify the search query to avoid returning too many and too few paragraphs. If too many paragraphs are retrieved, more keywords, such as replaceable keywords, will be added to restrict the number of candidate paragraphs. Otherwise, some of the key phrase will be removed.

After the paragraphs are retrieved, additional lexicon knowledge (Moby electronic thesaurus) is used to filter out irrelevant ones and sort remaining paragraphs according to the number of the words which appear in the question.

Moby electronic thesaurus contains about 1,000 concepts and each concept includes several words with similar word meaning [Moby00]. First, the replaceable keywords for each question are matched against the thesaurus to find one or more relevant concepts. Next, if the correspondent concept is found, the candidate paragraphs will be examined to find out the number of the words under the same concept. These words will be called extended query keywords (EQKs). Their number, which reflects the semantic closeness between a question and every paragraph, will be used to sort the paragraphs.

2.5 Answer Processing Module

We put forward a new approach in the Answer Processing module, which is named as “syntax constrained semantics verification”. The Answer Processing Module aims to determine and extract answers from the candidate paragraphs retrieved by the Search and Filtering module. Figure 2.5 gives the framework of this module.

Firstly, we determine the answer type of every noun word and noun phrase in candidate paragraphs by Name Entity Tagging, which has been described before. The words whose answer types correspond to the Question Type are marked *candidate answer.*

Then the candidate paragraphs are passed through a dependency parser, Minipar [Lin98], to get the parsing tree. In this dependency tree, every node corresponds to a syntax category and every word in the candidate paragraph resides in a node. The children of a node are those words that modify it. We try to find a path in the parsing tree connecting EQKs and candidate answer. If there exist such path, we extract the words on the path and the children of them in the parsing tree. Thus we get different word groups for each candidate answers. Here we assume that these word groups are more semantically closer to the corresponding candidate answer they extracted from than other words in the same sentence.

Now we have a word group for each candidate answer. We firstly extract the content words (noun, verb, adjective and adverb words) from question to form another word group. Both word groups are considered to be relative to the focus words in question and answers. Then we compute their semantic similarity using a approach named *extended vector space model.* The result of this step is a similarity score which varies from zero to one. This is the basic factor to determine the final answer.

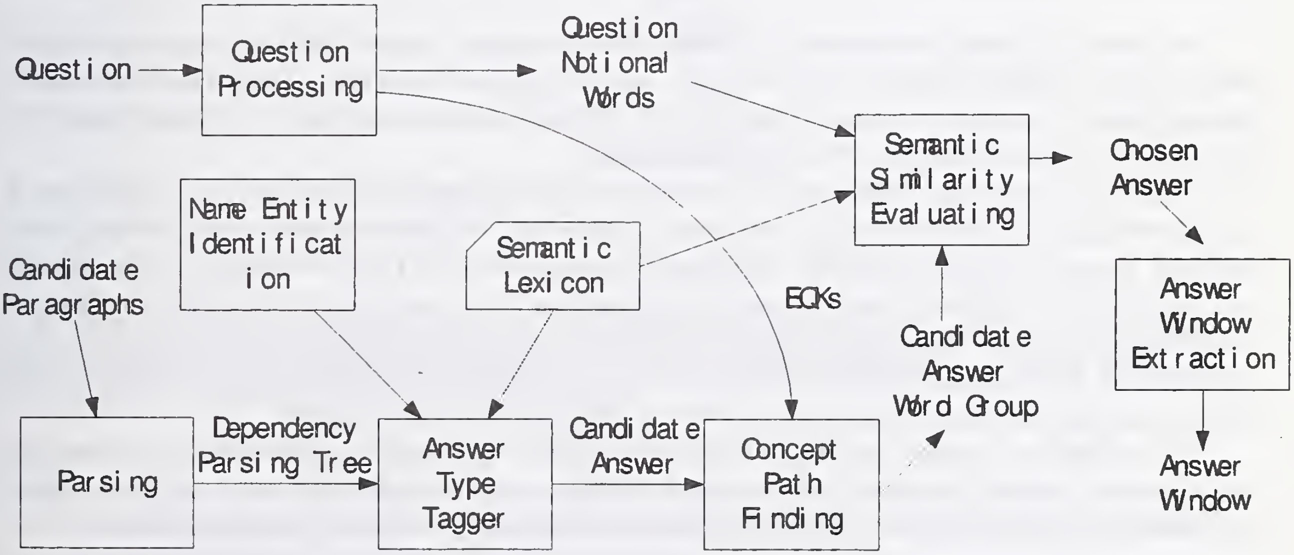


Figure 2.5 Syntax constrained semantic verification

For each candidate answer, a 50 byte-long section in the candidate paragraph, named *answer-windows*, is then created. This *answer-window* is centralized by the candidate answer. We evaluate each answer-window using the following three scores:

- *Semantic similarity:* This score has been computed using *extended vector space model.*
- *Syntax pattern score*: This score is based on the candidate answer sentence's syntactic structures. It takes several syntactic features into consideration, such as the length of the path in the parsing tree between the answer keywords and candidate answer, POS of EQKs and candidate answer.
- *Indicators score:* Some phrases or words, such as 'known as', 'called', 'named as' and some syntax features, such as appositive, strongly indicate a possible answer to specified questions such as 'who', 'what', 'which'.

The final score is a linear combination of all of these scores, where the weight of every score depends on the question feature.

2.6 Experiment results

This year we only take part in the main task of QA, and only submit one 50-byte run. Our results are not satisfactory. Statistics over 492 questions shows the strict mean reciprocal rank of 0.137 and lenient mean reciprocal rank of 0.145. Almost 80% of the questions return no correct response.

3. Web Retrieval

This year we attend the TREC-10 web subtask for the first time. We submit four runs for the web track. We used different combination of information in the four runs: title only and content only (fdut10wtc01), title with description and narrative information and content only (fdut10wac01), title only with link information (fdut10wtl01) and all title, description, narrative with link information (fdut10wal01).

3.1 System Architecture

In order to get better performance on web information retrieval, we have modified most of our original search engine, which is based on statistical model, and make a new search kernel that is based on extended Boolean search. Moreover, we split the document set and indices into several parts to efficiently handle the corpus of WT10g which contains 10G HTML documents.

Figure 3.1 shows the architecture of our web retrieval system. The first part in the left side is preprocessing module which can turn HTML pages into plain text. The second module is a preparation part for indexing, it combine all the small HTML documents in one directory of WT10g into a single file. The next step is indexing. We use stopword removing and morphology analysis to select entries of the indexing lexicon. What's more, we do not create index for the whole WT10g corpus due to the huge size of the corpus. Instead, the corpus is split into smaller parts, each of which is to be indexed independently. By this means, we can control the indices more easily than simply creating a larger index of the whole corpus.

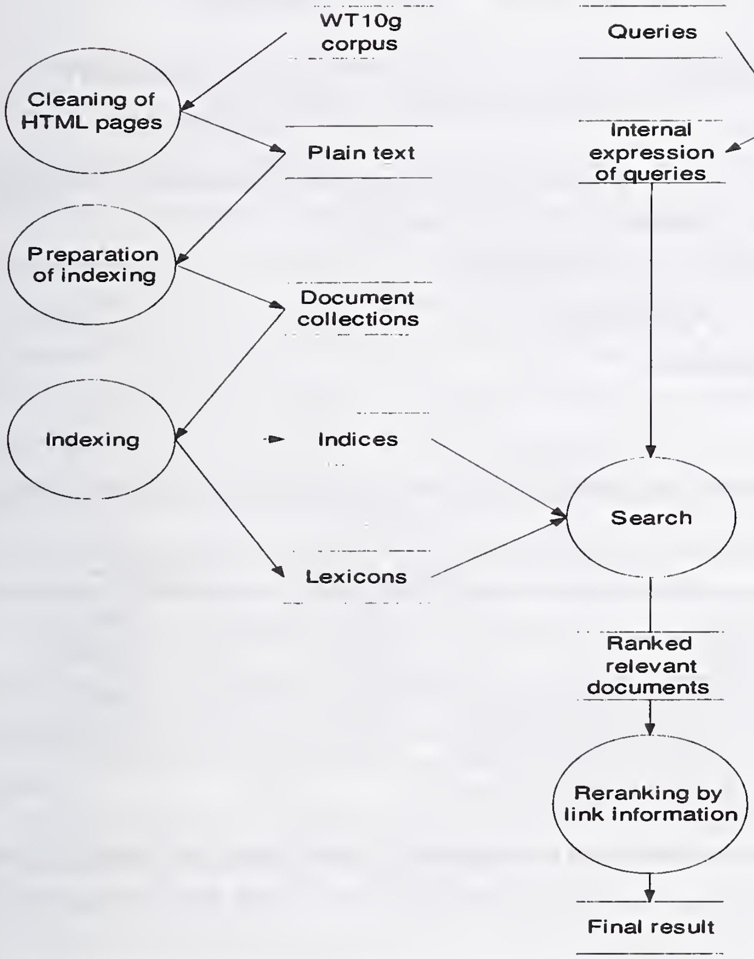
On the right side, the first step transfers the queries into several words, recognizes the phrase, and does some query expansion. The second module searches the index with the algorithm below and builds the ranked relevant document set. In the third step, link information is exploited to reorder the relevant documents.

3.2 Search Algorithm

The core algorithm is based on an extended Boolean retrieval engine called "short matching passages" [Kleinberg98], which intends to find the shortest passage that matching certain words. The assumption is that the shorter the passage is, the more possible that it will be relevant with the query. The following equation shows the basic idea of this algorithm. The *I(p,q)* represents the intensity of a shortest passage from the pth to qth word which contains certain keywords.

$$I(p,q)=\begin{cases}\left(\dfrac{K}{q-p+1}\right)^{a}, & \text{if } q-p+1\geq K\\ 1, & \text{otherwise}\end{cases} \tag{3.1}$$

Figure 3.1 Architecture of our web retrieval system



Before searching, natural language topics are first changed into Boolean queries, then the retrieval engine searches on all the individual indices simultaneously. Instead of the original equation in [Gordon98], we use our own equation (3.2) to calculate the score of each document.

$$S(D)=\sum_{(pi,qi)\in D} I(p_i,q_i)*w1(p_i,q_i) \tag{3.2}$$

Together with the length of short passage, we also consider the position of them (which is calculated in w1 in the equation above) to calculate the score of the document.

3.3 Reordering with link information

In order to improve Retrieving result by link information, we have tested Kleinberg Algorithm of hubs and authorities [Kleinberg98], co-citation and bibliographic coupling [Kraaij99]. After some experiments, we find that co-citation and bibliographic coupling can lead to the batter result. However, our best result still shows that link information cannot improve the search result.

The basic principle of co-citation is that if two documents (document A and B) cite the same documents, then document A and B are similar to some degree. The basic principle of bibliographic coupling is that if many documents cite both document A and B, then document A and document B are similar to some degree.

In our experiment, we use the formula of Wessel Kraaij [Kraaij99]:

$$Inlinkrel(d) = \frac{\sum_{i \in inlinks(d)} relevancy(i)}{\# inlinks(d)}, \quad Outlinkrel(d) = \frac{\sum_{i \in outlinks(d)} relevancy(i)}{\# outlinks(d)}$$

$$Citerel(d) = \frac{\sum_{i \in inlinks(d)} Outlinkrel(i)}{\# inlinks(d)}, \quad Bibcouplrel(d) = \frac{\sum_{i \in outlinks(d)} Inlinkrel(i)}{\# outlinks(d)}$$

Now, to the dth document, we have five scores: *S(D), InlinkRel(d), OutlinkRel(d), CociteRel(d), BibcoupleRel(d).* We can use the following formula to calculate the news core:

$$NewScore(d) = \alpha_1 * S(D) + \alpha_2 * InlinkRel(d) + \alpha_3 * OutlinkRel(d) \\ + \alpha_4 * CociteRel(d) + \alpha_5 * Bibcouple(d) \qquad (3.3)$$

Where, $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5$ are five parameters.

3.4 Experiment Results

We submitted four runs, whose names are *fdut10wtc01, fdut10wtl01, fdut10wac01, fdut10wal01.* The final result is shown in the following table.

Runs	Type	Average Precision	R-Precision
Fdut10wtc01	*Title-only/content-only*	0.1661	0.2061
Fdut10wtl01	*Title-only/content + link*	0.1544	0.1939
Fdut10wac01	*Title + Description + Narritive/content-only*	0.1661	0.2061
Fdut10wal01	*Title + Description + Narritive/content + link*	0.1248	0.1607

Table 3.1 Results of web track

We used many different combinations of parameters to reordering the content-only result, and find finally that α_1=1, α_2=0, α_3=0, α_4=1, α_5=0 can lead to the best result. But it still does not improve the content-only score.

4. Video Track

On Video Track, we take part in both Shot Boundary Detection and Video Retrieval. In the task of Shot Boundary Detection, we have submitted two runs with different parameters. One of them is precision-orientied, and another is recall-orientied. In the task of Video Retrieval, we submitted the results of 17 topics out of 74.

4.1. Shot Segmentation

In our system, we use FFD (*Frame-to-Frame Difference*) [Hanjalic97] to detect the shot boundary. But we redefined the difference between the *n*th frame and (*n*+*k*)th frame as Equation 4.1.

$$Z_k(n) = D_k(n) \times Y_k(n) \tag{4.1}$$

Where, $D_k(n)$ describes the difference on the luminance and $Y_k(n)$ describes the difference on the Color Histogram. The definitions of $D_k(n)$ and $Y_k(n)$ are as follows:

$$D_k(n) = C_1 \sum_i \sum_j | I(i,j,n+k) - I(i,j,n) | \tag{4.2}$$

$$Y_k(n) = 1 - C_2 \sum_i \min(H(i,n+k), H(i,n)) \tag{4.3}$$

Where, $I(i,j,n)$ is the average luminance of block (*i, j*) in frame *n*, and each block has 8*8 pixels. $H(i,n)$ is the Color Histogram value of frame *n* on the *i*th bin.

We use $Z_1(n)$ to detect the Cut Shot Boundary and $Z_k(n)$ to detect the Gradual Shot Boundary. θ_C and θ_G are thresholds for cut and gradual shot boundary. They will be discussed in next paragraph. If $Z_1(n)$ exceed the threshold θ_C, it may be caused by Cut Shot Boundary from frame *n* to frame *n*+1. We have also trying to reduce the influence of flashlight by compare the frames of both sides. When flashlight comes, it can be assumed that there will be more than one frame whose $Z_1(n)$ is larger than θ_C, also, the frames before and after the flashlight are similar. The Gradual shot boundary cannot be easily detected by the FFD of two continuous frames. We use $Z_k(n)$ (*k=50*) to magnify the frame-to-frame difference. But, it will cause more false alerts by motion in shot. To reduce that, we use motion detection: a sequence of continuous frames whose FFD is larger than θ_G will be labeled as Gradual Shot Boundary only if no efficient camera motion is detected and the frames before and after the sequence are dissimilar.

During the Shot Boundary Detection, threshold θ_C and θ_G are selected automatically every 500 frames [Zhu00]. The selection is according to the histogram of $Z_1(n)$ and $Z_k(n)$ in 500 frames. The histogram of $Z_1(n)$ and $Z_k(n)$ are calculated, and we find the first low point *p*. The value on *p* will be the threshold.

Other parameters are adjusted manually based on the 42 training video clips. The results show that the parameter selection is not sensitive in our method.

4.2. Video Retrieval

We submitted the results of 17 topics in video retrieval. Table 4.1 shows the type of these topics.

Topic Type	General	Known-Item	Total	Topic No.
People Searching	3	7	10	20,21,22,23,24,34,35,36,42,58
Object Serching	1	1	2	54,69
Video Text Searching	0	1	1	70
Camera Motion Searching	2	0	2	44,74
Shot Change Type Searching	1	0	1	65
Searching based on Document	0	1	1	62

Table 4.1 Submitted Topics

In Section 4.2.1, we will describe the architecture of our video retrieval system. Section 4.2.2 is the detailed description about implementation.

4.2.1 System Architecture

Our Video Retrieval System includes two parts. One is the off-line Indexing Sub-system, and another is on-line Searching Sub-system. Figure 4.1 describes the system architecture.

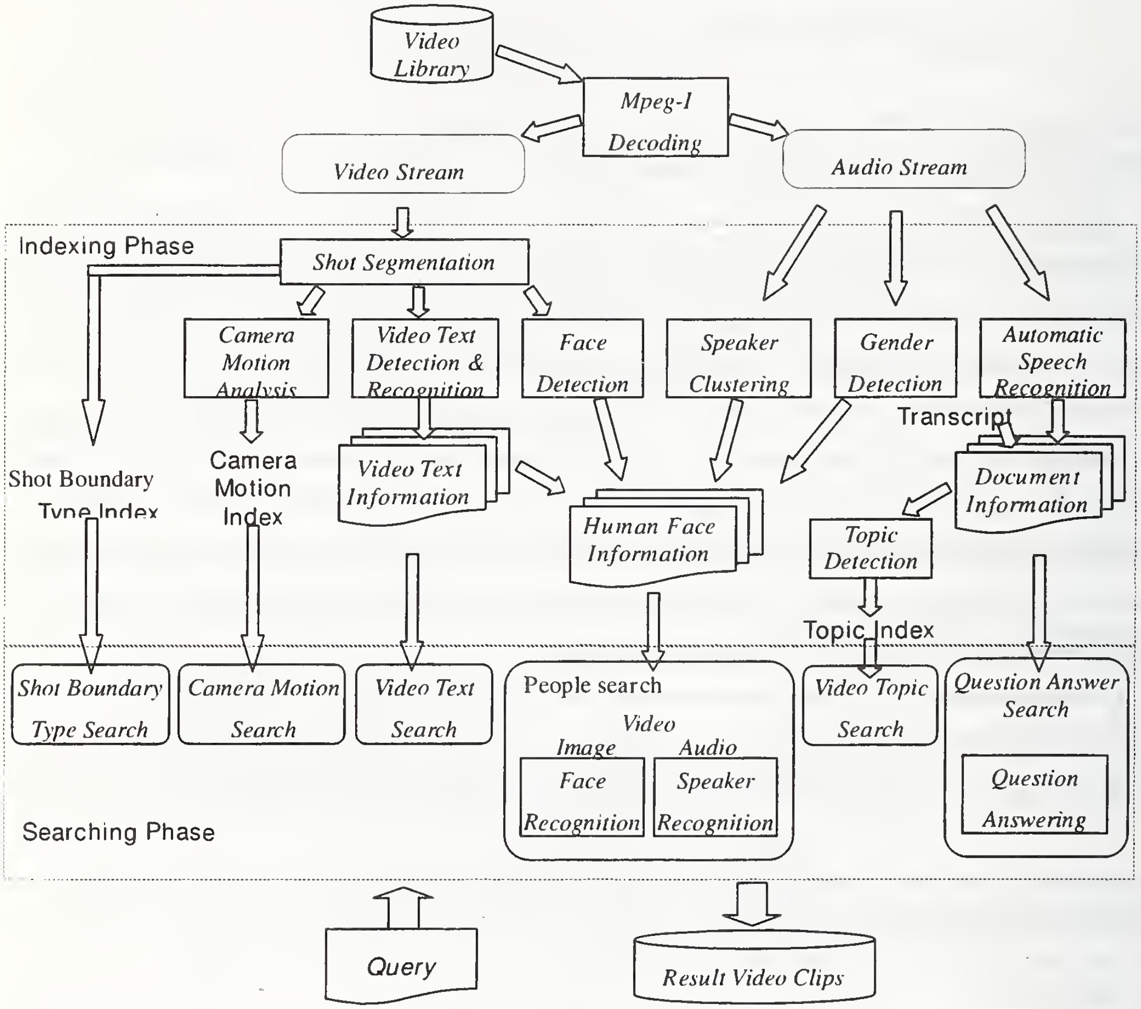


Figure 4.1 System architecture of video retrieval

4.2.2 Detailed Implementation

4.2.2.1 Qualitative Camera Motion Analysis

In our system, we analyze Camera Motion by the Motion Vectors obtained from MPEG stream. Each motion is composed of Motion Amplitude and Motion Direction. The system tries to segment shots into sub-shots automatically. We define sub-shot as some continuous frames in one shot with the similar camera motion.

4.2.2.2 Face Detection

Our method is designed mainly for interviewee detection. It has features: (1) The face is quite large, (2) the face has motion but the background is still. The method consists of three steps: Skin-Color based

Segmentation, Motion Segmentation, and Shape Filtering.

In skin-color based segmentation, we use several skin-color filters in different color spaces and combine them by AND operation. It is found that the result is better than the ones in any single color space. After that we have several skin-color regions. Similarly, we can have several motion regions by motion segmentation. By INTERSECTION operation, we have those regions which have both skin-color and motion. They are the candidates of face. For these candidates we use shape filtering to remove those too small or irregular ones and get the final results.

4.2.2.3 Face Recognition

In the training phase, we normalize all the training samples to 40*40 and make histogram equalization. After that, we clustering some of these samples and transform the face images to column vectors. Then:

(1) Let $\omega_1, \omega_2, \cdots, \omega_m$ be m training pattern classes, which correspond to m persons, respectively. Then calculate the within-class scatter matrix S_w of $\omega_1, \omega_2, \cdots, \omega_m$ and the covariance matrix S_t of all of samples.

(2) Calculate the zero subspace of the within-class matrix S_w.

(3) Calculate the eigenfaces in subspace $S_w^{-1}(0)$.

Suppose $S_w^{-1}(0) = span\{\alpha_1, \alpha_2, \cdots, \alpha_k\}$, where $\alpha_1, \alpha_2, \cdots, \alpha_k$ are the orthogonal unit vectors. Then, an arbitrary vector φ in $S_w^{-1}(0)$ can be represented as Equation 4.4:

$$\varphi = \xi_1 \alpha_1 + \cdots + \xi_k \alpha_k = PZ \qquad (4.4)$$

where, $P = (\alpha_1, \alpha_2, \cdots, \alpha_k)$, $Z^T = (\xi_1, \xi_2, \cdots, \xi_k)$

Define matrix S_{t1} as $S_{t1} = P^T S_t P$. Therefore, the calculating of eigenface in subspace $S_w^{-1}(0)$ can be transformed to the problem calculating the eigenvectors corresponding to the l largest eigenvalues of the matrix S_{t1}. Suppose $\beta_1, \beta_2, \cdots, \beta_l$ are the l orthogonal unit eigenvectors corresponding to the l largest eigenvalues of the matrix S_{t1}. Then the required eigenfaces are $P\beta_1, P\beta_2, \cdots, P\beta_l$.

In recognition phase, we also normalize the Example Face and the Testing Face to 40*40 at first. Then we make histogram equalization and transform them to column vectors x_1, x_2. Then the projection feature vectors of two images on the eigenface space can be obtained as Equation 4.5:

$$\xi_1 = (P\beta_1, P\beta_2, \cdots, P\beta_1)^T x_1, \quad \xi_2 = (P\beta_1, P\beta_2, \cdots, P\beta_1)^T x_2, \qquad (4.5)$$

Considering the distance $d = \|\xi_1 - \xi_2\|$, if $d \le \delta$, then we think that the Testing Face and the Example Face belong to same person. (δ is a threshold)

4.2.2.4 Video Text Detection and Recognition

There are three main parts in our Video Text Detection System: Text Block Detection, Text Enhancement and Binarization. To reduce the processing time, the system processes only one frame in every ten. On each processed frame, text block detection is first applied to get the position of each possible text line. This detection is based on edge image. Since edges are not sensitive to intensity changes and edges are dense in text line blocks, we calculate the gray scale edges in RGB space horizontally. The edge images are then binarized and run length analysis is applied to find candidate text blocks.

We use SSD (Sum of Square Difference) based block matching to track the detected text lines. All tracked text blocks are interpolated to a reasonable fixed size and then registered. At last, the tracked, interpolated and registered text blocks are combined by an average operation to reduce the noise and suppress the complex background.

An improved logical level technique (ILLT) is developed to binarize each candidate text block. This method can deal with different intensities of characters (i.e. characters may be brighter or darker than background) efficiently.

We have used commercial software: *TextBridge Pro Millennium* to recognize the binarized text block image. At last, the recognized string of each text blocks will be split into words and those words that are too short (less than 3) are removed in the filtering step.

4.2.2.5 Speaker Recognition and Speaker Clustering

During the Indexing Phase, Speaker Clustering can ensure the clustered shots that include human face are from the same person. And during the Searching Phase, Speaker Recognition can ensure the audio contained in the returned results is the same as the audio examples.

We use Vector Quantization (VQ) and Gaussian Mixture Model (GMM) to characterize each speaker. In these two methods, VQ worked faster than GMM while GMM performed better than VQ in some cases. In addition, in order to remedy the disturbance of noise and other backgrounds, a global model was used to modify the output of single speaker model.

Both VQ and GMM, especially after being integrated with global model, showed satisfactory detection and clustering effect if the speeches have the similar backgrounds. But if there is strong music background in the speech, the result will be worse.

4.2.2.6 Automatic Speech Recognition

We have used the Speech SDK of Microsoft as Automatic Speech Recognition (ASR) engine [www.Microsoft.com/speech]. There're different parameter sets for male and female speaker. One has to choose a proper parameter set to get better performance. And the engine has capability of background adaptation.

To increase the recall, we did not use gender detection. Instead, we use different parameter sets to recognize the same piece and give the confidence and time alignment of every recognized word. We have tried speaker change detection and audio classification. However, there is little improvement. The main reason is that the background music is too strong. Nevertheless, it recognized most of the keywords correctly and we use it for Topic Detection.

For the NIST video, we find that there are some errors in the human transcripts. We use ASR engine to give the result with time alignment. Then the result is aligned with the human transcript by ASR evaluation program. Finally, we get the rough time alignment between the audio and the human transcript. This time alignment is used for Question Answering.

4.2.2.7 Gender Detection

In our system, Gender Detection is made by Gaussian Mixture Model (GMM). We select pieces of audio that contains low background noise or music from the unused videos to train a Male Model and a Female Model. The feature is 12-dimension LPC cepstrum. Each model consists of 128 mixtures. Because the "clean" data of female is not enough, some male speakers are recognized as female. On the other hand, this error is also caused by the background music.

4.2.2.8 Topic Detection and Question Answering

In order to make Topic Detection and Question Answering, we should have a document library at first. In our system, the documents of videos are obtained in two ways. One is the manually created information. Another is the transcript created by automatic speech recognition (ASR). After that, Topic Detection and Question Answering module will run. These two modules come from Filtering and Question Answering, which can be found in Section 1 and 2. The training data of Topic Detection comes from the unused videos.

4.3. Results

Our results of shot boundary detection and video retrieval are presented in the following tables. As for shot boundary detection, our system acquires high performance on cut shot, while the results of gradual shot are not very good. Our performance of know-item search and general search both seems satisfactory. The reason may be attributed to the fact that we only submit the results of 10 know-item search topics and 7 general search topics.

Precision Oriented	Insert Rate	Delete Rate	Precision	Recall	Recall Oriented	Insert Rate	Delete Rate	Precision	Recall
Cut	0.039	0.028	0.961	0.972	Cut	0.039	0.028	0.961	0.972
Gradual	0.322	0.415	0.737	0.584	Gradual	0.350	0.391	0.727	0.608
All	0.133	0.154	0.889	0.845	All	0.143	0.146	0.883	0.853

Table 4.2 Shot Boundary Detection Results

Submitted Topics Number : 7	
Mean Precision	0.640

Table 4.3 General Search Results

Submitted Topics Number : 10				

	KI = 0.333
	RI = 0.333
		KI = 0.333
		RI = 0.666
			KI = 0.666
			RI = 0.333
				KI = 0.666
				RI = 0.666

Mean Precision	0.543	0.539	0.434	0.430
Mean Recall	0.678	0.636	0.528	0.486

Table 4.4 Known-Item Search Results

ACKNOWLEDGMENTS

This research was partly supported by NSF of China under contracts of 69873011 and 69935010. We are thankful to Lin Mei, Yuefei Guo, Weixiang Zhao Yi Zheng, Yaqian Zhou, Kaijiang Chen, Xiaoye Lu, Jie Xi, He Ren, Li Lian, Wei Qian, Hua Wan and Tian Hu for their help in the implementation.

Reference

[Hanjalic97] Alan Hanjalic, Macro Ceccarelli, Reginald L.Lagendijk, Jan Biemand: *Automation of Systems Enabling Search on Stored Video Data*.Proc.SPIE, Vol.3022 of Storage and Retrieval for Image and Video Database V, 1997

[Harabagiu99a] Sanda Harabagiu and Dan Moldovan. *A Parallel System for Textual Inference*. IEEE Transactions parallel and distributed systems, vol.10, 1999

[Harabagiu99b] Sanda Harabagiu and Dan Moldovan. *FALCON: Boosting Knowledge for Answer Engines*. CSE, Southern Methodist University, 1999

[Kleinberg98] Jon M. Kleinberg : *Authoritative Sources in a Hyperlinked Environment*, Proc. 9th ACM-SIAM Symposium on Discrete Algorithms 1998

[Kraaij99] Wessel Kraaij, Thijs Westerveld: *TNO/UT at TREC-9: How different are web documents*, Proceeding of TREC-9.

[Lin98] Dekang Lin. *A Dependency-based Method for Evaluating Broad-Coverage Parsers*. Natural Language Engineering. 1998.

[Moby00] http://www.dcs.shef.ac.uk/research/ilash/Moby/

[Zhu00]Xingquan Zhu, Xiangyang Xue, Lide Wu: *An Automatic Threshold Detection Method in Video Shot Segmentation*, Vol.37 No.1, Chinese Journal of Computer Research and Development, 2000

[Gordon98] Gordon V.Cormack, Christopher R.Palmer, Michael Van Biesbrouck, Charles L.A. Clarke. *Deriving Very Short Queryies for High Precision and Recall (MultiText Experiments for TREC-7)*, Proceeding of TREC-7.

Fujitsu Laboratories TREC2001 Report

Isao Namba

Fujitsu Laboratories Ltd.
namba@jp.fujitsu.com

Abstract

This year a Fujitsu Laboratory team participated in web tracks. Both for ad hoc task, and entry point search task, we combined the score of normal ranking search and that of page ranking techniques. For ad hoc style task, the effect of page ranking was very limitted. We only got very little improvement for title field search, and the page rank was not effective for description, and narrative field search.

For entry point search task, we compared three heuristics. The first heuristics supposed that entry point page contains key word and had high page rank score. The second heuristics supposed that entry point page contains key word in its head part and had high page ran score. The third heuristics supposes that entry point is pointed by the pages whose anchor string contains key word, and has high page rank score. The page rank improved the result of entry point search about 20-30% in rather small VLC10 test set, and the first heuristics got the best result because of its high recall.

1 System Description

For TREC2001, we added the new functions to trec_exec for entry point search. The functions includes score merging, evaluation of reciprocal rank and so on. We used Web Recommener Agent to get page ranking score. Except above modifications, the framework is same as that of TREC9[1].

1.0.1 Teraß

Teraß[2] is a fulltext search library, designed to provide an adequate number of efficient functions for commercial service, and to provide parameter combination testing and easy extension for experiments in IR.

1.0.2 trec_exec

trec_exec is designed for automatic processing of TREC. It contains a procedure controller, evaluation module , logging module, and all non-searching units such as query generation, query expansion and so on. trec_exec can execute all the TREC processing for one run in a few minutes, and it can be used for system tuning by hill-climing. The new functions added for TREC2001 Web track are heuristics for entry point search, evaluation of reciprocal rank, and accepting non-digit query number.

1.0.3 Web Recommender Agent

We used web recommeder agent tool developed for automatic domain specific web directory˙Tsuda et al[3] to get page ranking score. The page rank score is put into Teraß, and it is marged with normal ranking score.

2 Common Processing

2.1 Indexing/Query Processing

2.1.1 indexing vocabulary

The indexing vocabulary consists of character strings made up of letters, numbers, and symbols, and no stop words were used in indexing. For TREC8, we modified the grammar of the token recognizer to accept acronyms with symbols such as U.S., and AT&T as one token.

2.1.2 Stemmer

As the experiment in TREC8[1] shows, SMART[4] stemmer seems to be stable, we used SMART.

2.1.3 Information in inverted file

Text number, term frequency, and term position are stored for run time phrase processing.

2.1.4 Stop word list for query processing

As in the TREC8[1], we used a stop word list of about 400 words of Fox[5], and words with a high df (more than 1/7 of the number of all documents) were also treated as stop words.

2.1.5 Stop pattern removal

The expression of TREC queries are artificial, so frequently appearing patterns such as "relevant document“ are stop patterns. We generalized this observation, and removed the words which meet one of the following condition.

1. Word in stopword list is a stopword.
2. Word which is not a proper noun[1], and whose df in TREC1-7 queries is more than 400*0.1 is a stop word.
3. Word bi-gram whose df in TREC1-7 queries is more than 400*0.02 is a stop pattern.
4. Word tri-gram whose df in TREC1-7 queries is more than 400*0.01 is a stop pattern.
5. All the words in a sentence that contains "not relevant" are stop words.
6. 4 words following "other than" are stop words.
7. 4 words following "apart from" are stop words.

2.2 Weighting Scheme for Ranking

The scheme for term weight is

$$\begin{aligned} RankScore_o f_t erm(term) &= qtf * tf * idf \\ RankScore(t) &= \sum RankScore_o f_t erm() \end{aligned} \tag{1}$$

where

qtf is query term weight, *tf* is term weight in document, *idf* is inverse document frequency, and *t* is document. The score for one document is the sum of the term weights with co-occurence boosting.

[1] U.S appears 94 times in TREC1-7 queries.

1. qtf

   qtf is the combination of the following parameters

   $$qtf = \sum_{f} fw * tf * ttw \tag{2}$$

   where

   f is the topic field (title, description or narrative).
   fw is weight of the topic field.

   We set the value for the title field to 1.0, the value for the description field 1.0, the value for the narrative is 0.7. The weight for title field is decreased for TREC2001 because weighting title field that is weighting raw Web query, does not produce good result.

   Some teams [6], [7],[8] used weighting depending on field type, and we take the same approach.

   tf is the bare frequency in each field.

   ttw is the term type weight. It is set to 3 for terms, and set to 1 for phrase(word bi-gram).

2. tf

   We simply used the tf part of OKAPI[6].

   $$tf = \frac{(k_1 + 1) * term\_freq}{(k_1((1 - b) + \frac{b * doc\_length\_in\_byte}{average\_doc\_length\_in\_byte})} \tag{3}$$

   where $k_1 = 1.5, b = 0.75$

3. idf

   We used a modified idf of OKAPI. We introduced a cut off point for low df words, and decreased the idf value for high df words.

   $$idf = log_2 \frac{N - (n * \alpha)}{n} \tag{4}$$

   where

   N is the number of documents
   n is document frequency(df) if df > 1/10000 * N else n is $1/10000 * N$
   α is set to 3

2.3 Co-occurence Boosting

As in TREC8, we use co-occurence boosting techinique which favours co-occurence of query terms in a document. Co-ocurrence boosting is implemented by simply multipling the boost ratio to the similarity of each term.

$$S_i = \sum_{t} B * W_{t,i} \tag{5}$$

where

S_i is the degree of similarity between a document and topics.
i is the document number.
t is a term that document$_i$ includes.
$W_{t,i}$ is the part of similarity of term$_t$ in document$_i$.
B is the boost-ratio by term co-occurrence.

The best parameter B depends on the query, but it is difficult to tune them for each query. As in the case of field weighting, the weight for title field is decreased. We set the B to 1.05 for the title word, to 1.05 for the description word, and to 1.00 for the narrative word, and to 1.0 for the word added by query expansion.

2.4 phrase(bi-gram)

Instead of traditional IR phrase (two adjacent non-stopword pair with order or without order), we permitted limited distance in phrase. The motivation for introducing fixed distance is that that non-stopword may exist between two adjacent words in a query, and it producued slightly better result in the past experiment.[1] The term weight of bi-gram is fixed as 1/3 of a single word, and the distance is set to 4.

The phrase(bi-gram) is not used for entry point search, as it was too restrictive.

2.5 Query Expansion

Query Expansion was used for the ad hoc task, and small web track. The Boughanem formula[6] was used to select terms.

$$TSV = (r/R - \alpha s/S).w^{(1)} \tag{6}$$

where

$w^{(1)}$ is modified and more general version of Robertson/Sparck Jones weight.

The α was set 0.001, and k4 was -0.3, k5 was 1, and k6 was 64. The top 20 documents in the pilot search were supposed to be relevant, and the documents ranked from 500 to 1000 were supposed to be non-relevant. The top ranked 40 words which are not included in original query, which are not included in the stopword list of SMART, whose tsv score are more than 0.003, whose df are more than 60, and whose df are less than 200000 were added to the original query.

No collection enrichment technique was used, and query expansion was used only for ad hoc runs.

2.6 Page Ranking

Google is famous search engine that uses link based ranking approach[9]. The intutive idea of Google is that pages cited frequently are important, and that pages cited from important pages are also important. We adopted a Revised Page Ranking scheme which is proposed in Tsuda et al[3]. The scheme distingues the internal server(local) link, and external server(remote) link. The modification reflects the fact that the local link may be self link and less important than the link from external server (linked from others).

$$\begin{aligned} PageRank(A) &= (1-d) + d * \frac{PageRank(T_i)}{RC(T_i, A)} \\ RC(T,A) &= \frac{C(T)^2}{C_{rem}(T) + \alpha C_{loc}(T)} \\ &\quad (T, A : different\_domains) \\ &= \frac{\alpha C(T)^2}{C_{rem}(T) + \alpha C_{loc}(T)} \\ &\quad (T, A : same\_domains) \end{aligned} \tag{7}$$

where

$C_{rem}(T)$ is the number of remote link from T
$C_{loc}(T)$ is the number of local link from T
$C(T) = C_{rem}(T) + C_{loc}(T)$
$\alpha[0,1]$ is weighting factor for local link.

The d is set to 0.5, and the local link factor α is set to 0.1 for official runs.

2.7 Marging Score and Reranking

Both for entry point search, and ad hoc search of title field query, top N doucments are retrieved by normal ranking strategy first, and the documents are resorted by using page ranking score. To merge the normal ranking score and page ranking score, we levelize their gap by comparing their average score in top N documents. The equation8 is used for reranking.

$$
\begin{aligned}
S(t) = (1-\alpha) * gap * RankScore(t) \\
+\alpha * PageRank(t) \\
gap = \frac{\sum_{i=1}^{n} PageRank(i)}{\sum_{i=1}^{n} RankScore(i)}
\end{aligned}
\tag{8}
$$

where

RankScore is score of ranking for a document

PageRank is score of Page rank for a document

α is RankScore factor which takes between 0 and 1

n is the number of TREC output or the number of document retrieved.

The *RankScore* factor is different for ad hoc search, and entry point search. For ad hoc search, *RankScore* facotor is set to 0.95 or 1.0. It is because we got no improvment for ad hoc search except in title field using *PageRank* score. For entry point search, *RankScore* facotor is set to 0.47.

3 Ad hoc Search

Except title only runs, the query processing is same as that of traditional ad hoc task.

3.1 Result

Four runs were submitted, ie. flabxt, flabxtl, flabxtd, and flabxtdn. In the run id, the infix 'l' means link, 't' means using title field, 'd' means using description field, and 'n' means using narrative field.

Name	flabxt	flabxtl	flabxtd	flabxtdn
field	T	T	TD	TDN
link	NO	YES	NO	NO
Average Prec	.171	.170	.233	.184
R-Prec	.218	.208	.261	.224
P@20	.279	.277	.355	.316
Retrieved	50000	50000	50000	50000
Rel-ret	2155	2151	2449	2170
Relevant	3363	3363	3363	3363

Table 1: Official ad hoc result

The effect of page ranking is very limited or obscure for ad hoc search. We get very little improvement only for title only field search for test run, and no improvement for description, narrative field search at all.

It seems that web page with high page ranking score is often top of domain, or user, and is informative, but does not neccessarly match the information need of ad hoc style query.

4 Entry Point Search

For all entry point search runs, we used characteristics of Web, that is page rank score, anchor string and document structure.

4.1 Heuristics for entry point search

For entry search we experimented three different heuristics. We describe them here.

1. Simple Page Rank

   The first heuristics supposes that good entry point contains key words (theme of page) in it and has high page rank.

   This approach seems to be popular in web search engines such as google, teoma[10], and wisenut[11], and to produce good result if compared with simple ranking search.

   The ranking proceedure is that top 1000 pages are ranked by ranking equation1 , and they are rerankked using equation8.

2. Head Part and Page Rank

   The title of Web page often appears to contain key words (theme of page). For example, the entry point page for EP5 query "Haas Business School" contains "Haas School of Business"in head part, and the entry page for EP2 "Hunt Memorial Library" contains "Hunt Libary" in head part.

   The second heuristics supposes that good entry point contains key words (them of page) in head part of the page, and has high page rank score.

   As the head part of the page, we used top 256 byte of each page.

   This heuritstics might not get better result than simple page ranking, but was expected to get high precision if head part contains keywords.

   The ranking procedure is the same as that of simple page ranking heuristics.

3. Pointed by Anchor and Page Rank

   Web page name which is in anchor string seems to be most direct and reliable evidence of entry page though anchor string often contains pronoun such as "this", "here". Third heuristics suppose that good entry point is pointed by anchor string of high ranked pages, and has high page rank.

   In our experiment, we use 75 byte string around anchor, instead of using just anchor string. It is because we got little avaliable anchor string set. If WT10G test set contains enough web pages whose out link contains anchor string to the pages within WT10G, and that anchor strings match entry point search query, this heuristics is expected to be best in the three.

   The searching procedure is as follows.

   (a) Searching anchor string (around anchor string 75byte), and reraking using equation8. (anchoring page)

   (b) Collecting document ids which is pointed out by anchoring page.(referred page)

   (c) Scoring the referred page by following equation.

$$\begin{aligned} Ref(t) &= (1-\alpha) * gap * PageRank(t) + \\ &\qquad \alpha * ReferScore(t) \\ ReferScore(t) &= \sum_{i=0}^{n} Score(i) \\ Score(i) &= Rank(max)/Rank(i) * S(i) \end{aligned} \tag{9}$$

   where

   t is a document id in referred page set.

   α is Page Rank factor which takes between 0 and 1.

i is a document id of anchoring page which referts document id t
$Rank$ is rank of document id.
max is max number of retrieved text of anchoring page search.
$S(i)$ is equation8.

4.2 Result

Four runs were submitted:flabxcall, flabxct256, flabxc75a, and flabxemerge. flabxcall used Simple Page Rank1 heuristics, flabxct256 used Head Part and Page Rank2 heuristics, flabxc75a used Pointed by Anchor and Page Rank3, and flaxemerge merged the result of flabxcall, flabxct256, and flabxc75a. The table2 also includes entry point search without page ranking for comparison.

Name	flabxcall	flabxct256	flabxc75a	flaxemerge
Relevant	145	145	145	145
Retrieved@100	131	96	90	96
Retrieved@10	117	73	81	74
Rec-rank@100	.599	.363	.399	.363

Table 2: Entry Point Search Result

5 Conclusion

For ad hoc style search, we did not get improvment by just combing normal ranking score and page ranking score. But it is uncertain whether page ranking score has no effect for ad hoc style search, or WT10G test set is too small for ad hoc search using page ranking. For entry page search, we get about 30% improvement using page ranking score.

Acknowledgment

We thank to Dr. Tsuda who prepared page ranking score for WT10G test set.

References

[1] I Namba and N Igata. Fujitsu laboratories trec8 report. *The Eighth Text REtrieval Conference*, 2000.

[2] I Namba, N Igata, H Horai, K Nitta, and K Matsui. Fujitsu laboratories trec7 report. *The Seventh Text REtrieval Conference*, 1999.

[3] Hiroshi Tsuda, Takanori Ugai, and Kazuo Misue. Link-based acuqistion of web metadata for domain-specific directories. *The 2000 Pacific Rim Knowledge Acquistion Workshop(PKAW2000)*, 2000.

[4] SMART ftp cite. ftp://ftp.cs.cornell.edu/pub/smart/. 1999.

[5] Chiristopher Fox. Chapter 7, lexical analysis and stoplists. *Information Retrieval Data Structure and Algorithms ed. William B. Frakes, Ricardo Baeza-Yates Prentice Hall*, 1992.

[6] S E Robertson, S Walker, and M Beaulieu. Okapi at trec-7. *The Seventh Text REtrieval Conference*, 1999.

[7] D R H Miller, T Leek, and R M Schwarts. Bbn at trec-7. *The Seventh Text REtrieval Conference*, 1999.

[8] James Allan, Jamie Callan, Mark Sanderson, Jinxi Xu, and Steven Wegmann. Inquery and trec-7. *The Seventh Text REtrieval Conference*, 1999.

[9] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *WWW7 Conference*, 1998.

[10] teoma. http://www.teoma.com. 2000.

[11] wisenut. http://www.wisenut.com. 2000.

Hummingbird SearchServer™ at TREC 2001

Stephen Tomlinson[1]
Hummingbird
Ottawa, Ontario, Canada

February 4, 2002

Abstract

Hummingbird submitted ranked result sets for the topic relevance task of the TREC 2001 Web Track (10GB of web data) and for the monolingual Arabic task of the TREC 2001 Cross-Language Track (869MB of Arabic news data). SearchServer's Intuitive Searching™ tied or exceeded the median Precision@10 score in 46 of the 50 web queries. For the web queries, enabling SearchServer's document length normalization increased Precision@10 by 65% and average precision by 55%. SearchServer's option to square the importance of inverse document frequency (V2:4 vs. V2:3) increased Precision@10 by 8% and average precision by 12%. SearchServer's stemming increased Precision@10 by 5% and average precision by 13%. For the Arabic queries, a combination of experimental Arabic morphological normalizations, Arabic stop words and pseudo-relevance feedback increased average precision by 53% and Precision@10 by 9%.

1 Introduction

Hummingbird SearchServer[2] 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™. 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), called SearchSQL™, 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. Many character sets and languages are supported. SearchServer's Intuitive Searching algorithms were updated for version 4.0 which shipped in Fall 1999, and in subsequent releases of other products. SearchServer 5.0, which shipped in Spring 2001, works in Unicode internally [4] and contains improved natural language processing technology, particularly for languages with many compound words, such as German, Dutch and Finnish.

2 System Description

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, and running Windows NT 4.0 Service Pack 6. An

[1] Core Technology, Research and Development, stephen.tomlinson@hummingbird.com

[2] Fulcrum® is a registered trademark, and SearchServer™ , SearchSQL™, Intuitive Searching™ and Ful/Text™ are trademarks of Hummingbird Ltd. All other copyrights, trademarks and tradenames are the property of their respective owners.

internal development build of SearchServer 5.0 was used for the official TREC runs in July 2001 (build 5.0.504.156), which for the web and Arabic tasks should give essentially the same rankings as the commercial release version.

3 Setup

We describe how SearchServer was used to handle the topic relevance task of the TREC 2001 Web Track (10GB of web data) and the monolingual Arabic task of the TREC 2001 Cross-Language Track (869MB of Arabic news data).

3.1 Data

The WT10g collection of the Web Track consists of pages downloaded from the World Wide Web in 1997. It was distributed on 5 CDs. We copied the contents of each CD onto the OTWEBTREC e: drive (e:\data\wt10g\cd1 - e:\data\wt10g\cd5). The cd5\info subdirectory, containing supporting information not considered part of WT10g, was removed to ensure it wasn't indexed. The 5157 .gz files comprising WT10g were uncompressed. No further pre-processing was done on the data. Uncompressed, the 5157 files consist of 11,032,691,403 bytes (10.3GB), about 2MB each. Each file contains on average 328 "documents", for a total of 1,692,096 documents. The average document size is 6520 bytes. For more information on this collection, see [2].

Arabic Newswire A Corpus of the Cross-Language Track consists of articles from the Agence France Presse (AFP) Arabic Newswire from 1994-2000. It was distributed on 1 CD. We copied the contents of its TRANSCRIPTS directory to e:\data\Arabic. The 2337 gz files comprising the corpus were uncompressed. No further pre-processing was done on the data. Uncompressed the 2337 files consist of 911,555,745 bytes (869 MB), about 370KB each. Each file contains on average 164 "documents", for a total of 383,872 documents. The average document size is 2375 bytes. For more information on this collection, see [1].

3.2 Text Reader

To index and retrieve data, SearchServer requires the data to be in Fulcrum Technologies Document Format (FTDF). SearchServer includes "text readers" for converting most popular formats (e.g. Word, WordPerfect, etc.) to FTDF. A special class of text readers, "expansion" text readers, can insert a row into a SearchServer table for each logical document inside a container, such as directory or library file. Users can also write their own text readers in C for expanding proprietary container formats and converting proprietary data formats to FTDF.

Last year, for TREC-9, we wrote a custom text reader called cTREC to handle expansion of the library files of WT10g collection and to make a few conversions to the HTML format, described in [10]. We used cTREC again this year and made no significant changes regarding WT10g. This year, we will just describe how cTREC was extended for the Arabic collection.

The library files of the Arabic collection, like WT10g, consist of several logical documents, each starting with a <DOC> tag and ending with a </DOC> tag. After the <DOC> tag, the unique id of the document, e.g. 19940513_AFP_ARB.0001, is included inside <DOCNO>..</DOCNO> tags. The cTREC /E switch handles expansion of the Arabic library files into logical documents identically as for WT10g.

The Arabic documents contain SGML tags describing its structure (e.g. the headline is preceded by a <HEADLINE> tag and followed by a </HEADLINE> tag). The Document Type Definition (DTD) which specified the tags and entities used in the documents was provided in the ldc_arabic.dtd file on the CD. When invoked without the /w or /E switch, cTREC by default inserts control sequences to turn off indexing around all tags listed in the Arabic collection DTD (and additionally tags listed in the TREC disk 1-5 DTDs, as described last year), and converts all entities listed in the DTDs (e.g. "&AMP;" is converted to the ampersand "&"). By default, cTREC also turns off indexing for data delineated by certain tags because its content isn't considered helpful (for the Arabic collection, data delineated by HEADER, FOOTER and TRAILER tags is not indexed). cTREC looks ahead at most 5000 bytes for an end tag when it encounters a tag indicating indexing should be turned off; if the end tag is not found, indexing is not turned off.

The Arabic documents are in the UTF-8 character set, a variable-width encoding of Unicode, for which ASCII characters are represented with 1 byte (e.g. the Latin letter A, which is hexadecimal value 0x0041 in the UTF-16 encoding of Unicode, is 1 byte in UTF-8 (hexadecimal 0x41 or decimal 65)), and non-ASCII characters are represented with 2 to 4 bytes (e.g. the Arabic letter ALEF, which is 0x0627 in UTF-16, is 2 bytes in UTF-8 (0xd8 0xa7)). cTREC passes through the bytes of the documents unchanged (aside from the control sequences inserted and entities converted as described previously). SearchServer's Translation Text Reader (nti), was chained on top of cTREC and the UTF8_UCS2 translation was specified via its /t option to translate from UTF-8 to the UTF-16 encoding desired by SearchServer's parser.

3.3 Indexing

We created a SearchServer table called WT10GW for the web collection and two different SearchServer tables called ARAB01 and ARAB01A for the Arabic collection. For example, the SearchSQL statement to create the ARAB01A table was as follows:

```
CREATE SCHEMA ARAB01A CREATE TABLE ARAB01A
(DOCNO VARCHAR(256) 128) PERIODIC
BASEPATH 'E:\DATA' STOPFILE 'MYARAB.STP';
```

The stopfile differed for each table. For WT10GW, we used the same MYTREC.STP stopfile as last year, which contained 101 stopwords to not index, including all letters and single-digit numbers. For ARAB01, we did not use a stopfile. For ARAB01A, the stopfile MYARAB.STP did not actually contain any stopwords, but specified a non-default option to the parser to apply experimental Arabic morphological normalizations to the words before indexing:

```
PARSER="unicode/a=1"
```

The PARSER line of the stopfile specified the built-in unicode parser with the non-default option of a=1 which enables the experimental Arabic morphological normalizations. A powerful new feature of SearchServer 5.0 is the ability to "plug-in" a custom parser to extend or replace the default parser.

Into each table, we just inserted one row, specifying the top directory of the data set. e.g. for the ARAB01A table, we used this Insert statement:

```
INSERT INTO ARAB01A ( FT_SFNAME, FT_FLIST )
VALUES ( 'ARABIC', 'cTREC/E/d=128:s!nti/t=UTF8_UCS2:cTREC/@:s');
```

To index each table, we just executed a Validate Index statement such as the following:

```
VALIDATE INDEX ARAB01A VALIDATE TABLE
TEMP_FILE_SIZE 2000000000 BUFFER 256000000;
```

The VALIDATE TABLE option of the VALIDATE INDEX statement causes SearchServer to review whether the contents of container rows, such as directory rows and library files, are correctly reflected in the table. In this particular case, SearchServer initially validated the directory row by inserting each of its sub-directories and files into the table. Then SearchServer validated each of those directory and library file rows in turn, etc. Validating library file rows invoked the cTREC text reader in expansion mode to insert a row for each logical document in the library file, including its document id.

After validating the table, SearchServer indexed the table, in this case using up to 256MB of memory for sorting (as per the BUFFER parameter) and using temporary sort files of up to 2GB (as per the TEMP_FILE_SIZE parameter). The index includes a dictionary of the distinct words (after some Unicode-based normalizations, such as converting to upper-case and decomposed form, and in the case of the ARAB01A table, Arabic-specific normalizations as previously described) and a reference file with the locations of the word occurrences. Additionally, by default, each distinct word is stemmed and enough information saved so that SearchServer can efficiently find all occurrences of any word which has a particular stem. By default, the stemming is done with an English lexicon which has no effect on Arabic words.

4 Search Techniques

For the topic relevance task of the Web Track, the 50 "topics" were in a file called "topics.501-550". The topics were numbered from 501-550, and each contained a Title (which was an actual web query taken from a search engine log), a Description (NIST's interpretation of the query), and a Narrative (a more detailed set of guidelines for what a relevant document should or should not contain). We assumed the topics were in the Latin-1 character set, the default on North American Windows systems.

For the Cross-Language Track, the 25 topics were in 3 different languages (English, French and Arabic). We just used the Arabic topics in a file called "arabic_topics.txt". The Arabic topics were numbered AR1 to AR25. They were encoded in the ISO 8859-6 (Latin-6) character set.

We created an ODBC application, called QueryToRankings.c, based on the example stsample.c program included with SearchServer, to parse the topics files, construct and execute corresponding SearchSQL queries, fetch the top 1000 rows, and write out the rows in the results format requested by NIST. SELECT statements were issued with the SQLExecDirect api call. Fetches were done with SQLFetch (typically 1000 SQLFetch calls per query).

4.1 Character Set

SearchServer easily handled the issue of the Arabic queries and documents being in different character sets. Before running the Arabic queries, the SearchSQL statement "SET CHARACTER_SET 'ISO_8859_6'" was executed so that SearchServer would transcode the queries from Latin-6 to Unicode. The web queries were assumed to be in the Latin-1 character set, the default.

4.2 Intuitive Searching

For all runs, we used SearchServer's Intuitive Searching, i.e. the IS_ABOUT predicate of SearchSQL, which accepts unstructured text. For example, for topic 507 of the Web Track, the Title was "dodge recalls?". A corresponding SearchSQL query would be:

```
SELECT RELEVANCE('V2:3') AS REL, DOCNO
FROM WT10GW
WHERE FT_TEXT IS_ABOUT 'dodge recalls?'
ORDER BY REL DESC;
```

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 1000" was previously executed so that the working table would contain at most 1000 rows.

4.3 Stemming

SearchServer "stems" each distinct word to one or more base forms, called stems, using lexicon-based natural language processing technology. For example, in English, "baby", "babied", "babies", "baby's" and "babying" all have "baby" as a stem. Compound words in languages such as German, Dutch and Finnish produce multiple stems; e.g., in German, "babykost" has "baby" and "kost" as stems.

By default, Intuitive Searching stems each word in the query, counts the number of occurrences of each stem, and creates a vector. Optionally some stems are discarded (secondary term selection) if they have a high document frequency or to enforce a maximum number of stems, but we didn't discard any stems for our TREC runs this year. The index is searched for documents containing terms which stem to any of the stems of the vector.

The VECTOR_GENERATOR set option controls which stemming operations are performed by Intuitive Searching. For most Web Track runs, we used the default VECTOR_GENERATOR setting ('word!ftelp/base/single | * | word!ftelp/inflect') which assumes the English language, but for one submitted run we disabled stemming (using SET VECTOR_GENERATOR ''). (By default, SearchServer's index supports both exact matching (after some Unicode-based normalizations, such as converting to upper-case and decomposed form) and matching on stems.) For the Arabic runs, we always disabled stemming. When searching SearchServer table ARAB01A, for which Arabic morphological normalizations were applied to each word at index-time, the same normalizations were automatically applied to each query term.

Besides linguistic expansion from stemming, we did not do any other kinds of query expansion this year. For example, we did not use approximate text searching for spell-correction because the queries were known to be spelled correctly this year. We did not use row expansion or any other kind of blind feedback technique for the official runs.

4.4 Statistical Relevance Ranking

SearchServer calculates a relevance value for a row of a table with respect to a vector of stems based on several statistics. The inverse document frequency of the stem is estimated from information in the dictionary. The term frequency (number of occurrences of the stem in the row (including any term that stems to it)) is determined from the reference file. The length of the row (based on the number of indexed characters in all columns of the row, which is typically dominated by the external document), is optionally incorporated. The already-mentioned count of the stem in the vector is also used. To synthesize this information into a relevance value, SearchServer dampens the term frequency and adjusts for document length in a manner similar to Okapi [6] and dampens the inverse document frequency in a manner similar to [7]. SearchServer's relevance values are always an integer in the range 0 to 1000.

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'). SearchServer's RELEVANCE_DLEN_IMP parameter controls the importance of document length (scale of 0 to 1000) to the ranking.

4.5 Query Stop Words

Our QueryToRankings program removed words such as "find", "relevant" and "document" from the topics before presenting them to SearchServer, i.e. words which are not stop words in general but were commonly used in the topics as general instructions. For our CLEF runs this year [9], we expanded the list for several languages based on examining the CLEF 2000 topics (not this year's TREC topics). The full list for English is now as follows: "item", "items", "find", "documents", "document", "relevant", "report", "what", "identify", "about", "discussing". (Some of these words, such as "about", were also in the mytrec.stp stopfile, so removing them was redundant.) Although they were unlikely to appear, corresponding words for other languages, e.g. the German word "dokumente", were removed if encountered. No Arabic words were in the list. This step was found to be only minor benefit for CLEF [9].

If a query returned no results, based on our experience with the TREC-9 Large Web Task last year, the reason was often that the queries consisted entirely of stop words. The most famous stopword query, "to be or not to be", is a philosophical question, so for the Web Track this year we pre-selected document WTX094-B32-167 (the Yahoo Philosophy page) to be returned if otherwise the query would return no results. (It turned out the situation came up this year for topic 531 (who and whom), which was judged to be a grammar question, and hence the Philosophy page was properly judged non-relevant.) As a more general technique, we may just return the results of the query "philosophy" for this situation in future years.

5 Results

The evaluation measures are explained in an appendix of the conference proceedings. Briefly: *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).

We use the following abbreviations for the evaluation measures in this paper:

AvgP: Average Precision
P@5, P@10, P@15, P@20, P@30: Precision after 5, 10, 15, 20 and 30 documents retrieved, respectively
Rec0, Rec30: Interpolated Precision at 0% and 30% Recall, respectively
AvgH: Average Precision just counting Highly Relevants as relevant
H@5, H@10, H@15, H@20, H@30: P@5, P@10, P@15, P@20 and P@30 just counting Highly Relevants as relevant, respectively
H0, H30: Rec0 and Rec30 just counting Highly Relevants as relevant, respectively

We refer to the scores with a fixed cutoff (P@5, P@10, P@15, P@20, P@30) as *early precision* scores. The other scores (AvgP, Rec0, Rec30), which can be influenced by results later in the result list, we call *recall-oriented* scores.

5.1 Web Track

The topic relevance task of the Web Track was to run 50 web queries against 10GB of web data and submit a list of the top-1000 ranked documents to NIST for judging.

NIST produced a "qrels" file: a list of documents judged to be highly relevant, relevant or not relevant for each topic. From these, the scores were calculated with Chris Buckley's *trec_eval* program, which counts all relevants the same, including highly relevants. To produce scores which just counted highly relevants as relevant, we ran *trec_eval* a 2nd time on a modified version of the qrels file which had the ordinary relevants filtered out, then multiplied by 50/44 (in 6 of the 50 topics, there were no highly relevants). Hence the scores focused on highly relevants are averaged over just 44 topics.

We submitted 4 runs for the topic relevance task of the Web Track: hum01t, hum01tl, hum01tlx and hum01tdlx. The run codes we used are as follows:

hum: Hummingbird
01: TREC 2001
t: title field used
d: description field used
n: narrative field used
l: linguistic expansion (stemming) enabled
x: weighting scheme squared importance of inverse document frequency and increased importance of document length normalization (i.e. RELEVANCE_METHOD 'V2:4', RELEVANCE_DLEN_IMP 750, instead of 'V2:3' and 500 respectively)

Tables 1 and 2 show various scores of our submitted Title-only runs, i.e. runs which just used the original web query. Additionally, for some measures, NIST reported the median scores of the 77 submitted Title-only runs from all groups for each of the 50 topics; we show the average of the median scores:

Run	AvgP	P@5	P@10	P@15	P@20	P@30	Rec0	Rec30
1a: hum01tlx	0.1949	38.4%	33.2%	30.53%	28.6%	25.47%	0.6168	0.2708
1b: hum01tl	0.1784	36.8%	32.2%	28.93%	26.6%	23.73%	0.5940	0.2485
1c: hum01t	0.1582	39.6%	30.8%	28.13%	26.0%	22.67%	0.5665	0.2210

Median (77 runs)	0.1402	n/a	26.6%	n/a	22.5%	20.53%	n/a	n/a

Table 1: Precision of Submitted Title-only runs counting all relevants the same

Run	AvgH	H@5	H@10	H@15	H@20	H@30	H0	H30
2a: hum01tlx	0.1909	18.6%	13.0%	11.51%	9.9%	7.81%	0.4186	0.2417
2b: hum01tl	0.1855	18.6%	13.0%	10.76%	9.3%	7.58%	0.3890	0.2235
2c: hum01t	0.1684	20.9%	14.3%	11.36%	9.5%	7.73%	0.3807	0.2116

Table 2: Precision of Submitted Title-only runs just counting Highly Relevants as relevant

Impact of Stemming (compare hum01t to hum01tl): When counting just highly relevants as relevant, the early precision scores (P@5, P@10, P@15, P@20, P@30) were higher with stemming disabled, and the recall-oriented scores (AvgP, Rec0, Rec30) were higher with stemming enabled. This result fits intuition: stemming ought to increase recall because more word variants are allowed to match, but sometimes the variants may not reflect the query as accurately, hurting precision. When counting all relevants the same, the earliest precision score (P@5) was again higher with stemming disabled; the other early precision scores were modestly higher with linguistic expansion enabled, though by a smaller margin than for the recall-oriented scores. The difference from the result for highly relevants may be from precision suffering less when the quality of the match is not required to be as high. None of these differences were statistically significant at the 5% level by the two-sided Wilcoxon signed rank test [5].

If stemming helps recall but hurts early precision, it may be better to give a higher weight to the original query word than the generated variants. We haven't yet run any experiments with this approach.

Note that all topics were English. In our CLEF 2001 experiments this year, we found that the impact of SearchServer's stemming was normally larger in other European languages, particularly in German and Dutch. [9]

There are two differences in the weighting scheme between the submitted hum01tl and hum01tlx runs. To isolate the impact of each change, Tables 3 and 4 show diagnostic runs whose settings are the same as for hum01tlx except for the document length importance setting (RELEVANCE_DLEN_IMP). Rows 3d and 4d are the same as rows 1a and 1b, respectively (the hum01tlx run). Rows 3c and 4c differ from hum01tl in just the relevance method (V2:4 vs. V2:3):

DLen Importance	AvgP	P@5	P@10	P@15	P@20	P@30	Rec0	Rec30
3a: 0	0.1289	21.6%	21.0%	18.93%	17.7%	16.80%	0.3896	0.1828
3b: 250	0.1927	36.0%	31.6%	29.20%	26.6%	23.80%	0.5967	0.2708
3c: 500	0.2004	39.6%	34.8%	32.13%	29.3%	25.47%	0.6178	0.2803
3d: 750	0.1949	38.4%	33.2%	30.53%	28.6%	25.47%	0.6168	0.2708
3e: 1000	0.1725	34.8%	30.8%	28.27%	26.3%	23.33%	0.5663	0.2326

Table 3: Impact of Document Length Normalization (Title-only runs)

DLen Importance	AvgH	H@5	H@10	H@15	H@20	H@30	H0	H30
4a: 0	0.1084	10.0%	8.6%	6.67%	6.5%	5.31%	0.2259	0.1594
4b: 250	0.1795	17.7%	12.7%	11.22%	9.8%	8.18%	0.3447	0.2194
4c: 500	0.2000	20.5%	14.3%	11.97%	9.8%	8.56%	0.4228	0.2516
4d: 750	0.1909	18.6%	13.0%	11.51%	9.9%	7.81%	0.4186	0.2417
4e: 1000	0.1595	14.5%	11.4%	10.15%	9.1%	7.35%	0.3210	0.2131

Table 4: Impact of Document Length Normalization (Title-only runs) on Highly Relevants

Impact of Document Length Normalization: Ignoring document length (rows 3a and 4a) hurt all scores; average precision was 30-55% higher in the other rows, and Precision@10 was 45-65% higher in the other rows. The impact on highly relevants was even larger; average precision was up to 85% higher in the other rows. For most measures, a setting of 500 produced the highest scores of the settings investigated. When comparing the document length importance setting of 500 with 0 (i.e. compare 3c to 3a, and 4c to 4a), the differences in all of the shown measures (i.e. from AvgP through H30) are statistically significant at the 1% level by the two-sided Wilcoxon signed rank test.

Impact of squaring the importance of inverse document frequency (compare 1b to 3c, and 2b to 4c, which are the same except for the relevance method (V2:3 vs V2:4)): All measures were higher with the importance of inverse document frequency squared (relevance method V2:4). The differences were statistically significant at the 1% level for AvgP, P@10 and P@20, and at the 5% level for P@15, P@30, Rec30, AvgH, H@30 and H30, by the two-sided Wilcoxon signed rank test. Note that on some other test collections, such as the CLEF news collections, we have seen V2:3 receive higher scores.

For the benefit of the relevance assessment pools, we donated one run with the Description field included (hum01tdlx) and assigned it highest judging priority. Tables 5 and 6 show scores for hum01tdlx and a diagnostic run which is the same as hum01tdlx except that the Narrative field was also included. Table 5 also shows averages of the medians reported by NIST which were based on a group including all submitted non-Title-only runs, including 2 manual runs and some runs using the Narrative.

Impact of including the Description field (compare hum01tlx to hum01tdlx, i.e. 1a to 5a, and 2a to 6a): All scores were higher when including the Description. The differences were statistically significant at the 5% level for AvgP, P@20, Rec0, H@30 and H30 by the two-sided Wilcoxon signed rank test. None of the differences were statistically significant at the 1% level.

Run	AvgP	P@5	P@10	P@15	P@20	P@30	Rec0	Rec30
5a: hum01tdlx	0.2201	44.4%	36.6%	33.73%	32.2%	28.33%	0.7207	0.2977
5b: 5a + Narr	0.2362	46.8%	39.0%	35.07%	33.3%	28.93%	0.7361	0.3252
Median (20 runs)	0.1877	n/a	37.2%	n/a	31.2%	26.87%	n/a	n/a

Table 5: Precision of non-Title-only runs counting all relevants the same

Run	AvgH	H@5	H@10	H@15	H@20	H@30	H0	H30
6a: hum01tdlx	0.2082	20.5%	15.0%	12.73%	11.4%	9.69%	0.4478	0.2913
6b: 6a + Narr	0.1950	20.9%	14.5%	13.18%	11.8%	9.63%	0.4241	0.2647

Table 6: Precision of non-Title-only runs just counting Highly Relevants as relevant

Impact of including the Narrative field (compare 5a to 5b, and 6a to 6b): When counting all relevants the same, all investigated scores were higher when including the Narrative. When just counting highly relevants as relevant, most of the scores were lower when including the Narrative. None of the differences were statistically significant at the 5% level by the two-sided Wilcoxon signed rank test.

Table 7 shows per-topic comparisons of our submitted runs with the medians in their category for the measures reported by NIST: Average Precision, Precison@10, Precision@20 and Precision@30, respectively. In each comparison, we show the number of topics on which the run scored higher than the median, lower than the median and tied with the median (Higher-Lower-Tied). Differences statistically significant at the 1% level by the two-sided Wilcoxon signed rank test are marked with two asterisks (**), and differences just significant at the 5% level are marked with a single asterisk (*):

Run	AvgP	P@10	P@20	P@30
hum01tlx	41-8-1 **	23-9-18 **	27-7-16 **	28-4-18 **
hum01tl	36-13-1 **	21-4-25 **	22-10-18 **	24-11-15 **
hum01t	28-21-1 *	21-9-20 *	23-11-16 *	25-13-12
hum01tdlx	31-17-2 **	14-15-21	18-15-17	20-15-15

Table 7: Per-topic comparison of Submitted runs with Medians

The per-topic comparisons show a lot of ties in the early precision scores, particularly P@10, because of the small number of documents considered. Still, in each measure, the difference of the hum01tlx and hum01tl runs with the medians is statistically significant at the 1% level by the two-sided Wilcoxon signed rank test (the calculation of the significance level discards the ties, following [5]).

The significance level (p-value) for the two-sided Wilcoxon signed rank test defined in [5] was computed by our own implemented algorithm. The computation is exact (aside from double-precision roundoff errors) even in the case of tied absolute differences. For the Wilcoxon signed rank test we assume that the differences on differing topics are independent, and that the differences are from a distribution which is symmetric about a median difference. The test tests the hypothesis that the median difference is zero. For more details, see [5].

5.2 Cross-Language Track

Table 8 shows our submitted Arabic runs, which were all monolingual runs, i.e. used the Arabic versions of the topics. The baseline run was humAR01td, a Title+Description run which applied some experimental Arabic morphological normalizations to the words before indexing. The other runs were the same as the baseline except for one factor. Run humAR01tdm disabled the Arabic-specific normalizations (but not general ones such as case normalization). Run humAR01tdx used a different weighting scheme which squared the importance of inverse document frequency and also increased the adjustment for document length. Run humAR01t was Title-only. Run humAR01tdn was Title+Description+Narrative. No stop words were applied:

Run	AvgP	P@5	P@10	P@15	P@20	P@30	Rec0	Rec30
8a: humAR01td	0.2441	50.4%	49.2%	47.73%	46.2%	43.07%	0.7494	0.3149
8b: humAR01tdm	0.2087	48.0%	48.4%	48.27%	46.2%	41.20%	0.7238	0.2848
8c: humAR01tdx	0.2465	51.2%	48.0%	46.13%	43.8%	39.07%	0.7486	0.3235
8d: humAR01t	0.2663	53.6%	48.8%	48.0%	45.0%	41.33%	0.7755	0.3390
8e: humAR01tdn	0.2395	62.4%	51.6%	49.33%	45.8%	43.47%	0.8312	0.2823

Table 8: Precision of Submitted Monolingual Arabic runs

Impact of Arabic morphological normalizations (compare humAR01tdm to humAR01td): All investigated scores except P@15 were tied or higher when the Arabic morphological normalizations were applied. None of the differences were statistically significant at the 5% level by the two-sided Wilcoxon signed rank test. (The Arabic test collection just contained 25 topics, making it harder to detect significant differences than for the web collection, which had 50 topics.)

Impact of changing the weighting scheme (compare humAR01td to humAR01tdx): It made little difference.

Impact of excluding the Description field (compare humAR01td to humAR01t): Some scores were higher when just using the Title field (AvgP, P@5, P@15, Rec0, Rec30). Others were higher when the Description was included (P@10, P@20, P@30). It was noted at the conference that the Titles for these topics were a little longer than usual.

Impact of including the Narrative field (compare humAR01td to humAR01tdn): Some scores were higher when including the Narrative (P@5, P@10, P@15, P@30, Rec0) but a few were lower (AvgP, P@20, Rec30).

After the conference, we added approximately 2000 Arabic stop words based on the ISI list [8]. Table 9 shows the new scores when redoing runs humAR01td, humAR01t and humAR01tdn with the stop words (note: an experimental version of SearchServer 5.3 (pre-release) was used for the Arabic diagnostic runs, but its document ranking with respect to Arabic was the same as SearchServer 5.0's except for the experimental differences described in this section):

Run	AvgP	P@5	P@10	P@15	P@20	P@30	Rec0	Rec30
9td: 8a + stp	0.2617	50.4%	52.0%	48.27%	47.8%	43.87%	0.7558	0.3415
9t: 8d + stp	0.2692	52.0%	48.0%	45.87%	43.0%	40.27%	0.7624	0.3523
9tdn : 8e + stp	0.2639	60.8%	54.4%	51.47%	50.6%	44.40%	0.8631	0.3335

Table 9: Precision of runs using Arabic Stop Words

Impact of Arabic stop words (compare 8a to 9td, 8d to 9t, and 8e to 9tdn): While the scores just increased modestly, the increases were consistent across topics. For example, in the Title+Description case (runs 8a vs 9td), 22 topics had a higher score in average precision, just 1 lower and 2 tied, when using the stop words. In the Title+Description case, the difference in average precision was statistically significant at the 1% level, and the differences in P@10 and P@20 were statistically significant at the 5% level, by the two-sided Wilcoxon signed ranked test.

As another experiment, we added some morphological normalizations mentioned by other groups which we weren't already using, particularly the orthographic variation mentioned by BBN [11] (YEH vs. ALEF MAKSURA at end of word) and removing WAW prefixes as mentioned by Berkeley [3]. Table 10 shows the scores when redoing the runs of Table 9 with the additional rules:

Run	AvgP	P@5	P@10	P@15	P@20	P@30	Rec0	Rec30
10td: 9td +rules	0.2831	57.6%	51.2%	48.53%	45.8%	43.60%	0.7917	0.3837
10t: 9t +rules	0.2939	57.6%	49.6%	47.47%	46.6%	42.80%	0.8269	0.3853
10tdn: 9tdn+rules	0.2802	62.4%	56.8%	51.47%	49.4%	45.07%	0.8746	0.3646

Table 10: Precision of runs with Additional Arabic Morphological Normalizations

Impact of additional Arabic morphological normalizations (compare 9td to 10td, 9t to 10t, and 9tdn to 10tdn): Most of the scores modestly increased from the new rules. Focusing on the Title+Description case, the difference in P@5 was statistically significant at the 5% level by the two-sided Wilcoxon signed rank test, but the other differences were not.

Combined impact of (updated) Arabic morphological normalizations and stop words (compare 8b to 10td): The combination of Arabic morphological normalizations (including the experimental updates) and the stop words increased average precision by 36% and the difference in average precision was statistically significant at the 1% level by the two-sided Wilcoxon signed rank test. None of the other differences were statistically significant even at the 5% level. Precision@10 just increased 6%. Early precision may benefit less from normalization rules because there may be enough exact matches to find.

Some groups found that query expansion worked well on this collection, so we applied the "row expansion" technique described in last year's paper [10]. Roughly speaking, row expansion is a pseudo-relevance feedback technique in which it is assumed that the top rows of the initial query are relevant and SearchServer's Intuitive Searching uses them to generate new, broader queries. (In practice, a SearchServer user would specify which rows are relevant, which should produce better results than the "blind" automatic technique applied here.) Like last year, we used the top-5 rows; a minor difference is that for the row expansion queries, we used a document frequency parameter of 5% (i.e. RELEVANCE_METHOD 'V2:3:05') instead of the experimental secondary term selection approach used last year. Table 11 shows the scores after applying row expansion to the runs of Table 10:

Run	AvgP	P@5	P@10	P@15	P@20	P@30	Rec0	Rec30
11td: 10td +exp	0.3185	59.2%	52.8%	52.53%	50.6%	46.80%	0.7918	0.4430
11t: 10t +exp	0.3268	58.4%	56.0%	53.33%	50.6%	47.47%	0.8070	0.4262
11tdn: 10tdn+exp	0.3285	63.2%	60.4%	55.47%	53.8%	50.13%	0.8684	0.4351

Table 11: Precision of Arabic runs after Row Expansion

Impact of row expansion (compare 10td to 11td, 10t to 11t, and 10tdn to 11tdn): Most of the scores modestly increased from row expansion; average precision was up 11-17%. Focusing on the Title+Description case, the differences in average precision and Rec30 were statistically significant at the 1% level by the two-sided Wilcoxon signed rank test; the other differences were not statistically significant at even the 5% level. Query expansion techniques such as row expansion may help recall-oriented measures by contributing terms from the top documents which are not automatically generated from the initial query.

Combined impact of (updated) Arabic morphological normalizations, stop words and row expansion (compare 8b to 11td): Applying all the techniques described above increased average precision by 53%, but increased Precison@10 by just 9%. The differences in average precision and Rec30 were statistically significant at the 1% level, and the difference in Precision@5 was statistically significant at the 5% level, by the two-sided Wilcoxon signed rank test. None of the other differences were statistically significant at the 5% level.

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] Peter Bailey, Nick Craswell and David Hawking, Engineering a multi-purpose test collection for Web retrieval experiments (DRAFT, accepted, subject to revision, by Information Processing and Management). http://www.ted.cmis.csiro.au/TRECWeb/

[3] Aitao Chen and Frederic Gey. (University of California at Berkeley.) Translation Term Weighting and Combining Translation Resources in Cross-Language Retrieval. Notebook paper in draft TREC 2001 Conference Proceedings.

[4] Andrew Hodgson. Converting the Fulcrum Search Engine to Unicode. In *Sixteenth International Unicode Conference*, Amsterdam, The Netherlands, March 2000.

[5] Myles Hollander and Douglas A. Wolfe. *Nonparametric Statistical Methods.* Second Edition, 1999. John Wiley & Sons.

[6] 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

[7] Amit Singhal, John Choi, Donald Hindle, David Lewis and Fernando Pereira. AT&T at TREC-7. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7).* NIST Special Publication 500-242. http://trec.nist.gov/pubs/trec7/t7_proceedings.html

[8] 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

[9] Stephen Tomlinson. Stemming Evaluated in 6 Languages by Hummingbird SearchServer™ at CLEF 2001. To appear in Carol Peters, editor, Proceedings of the Cross-Language Evaluation Forum (CLEF) 2001, Darmstadt, Germany, September 2001. Springer Lecture Notes for Computer Science (LNCS) series.

[10] 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

[11] Jinxi Xu, Alex Fraser and Ralph Weischedel. TREC 2001 Cross-lingual Retrieval at BBN. Notebook paper in draft TREC 2001 Conference Proceedings.

Juru at TREC 10 – Experiments with Index Pruning

David Carmel, Einat Amitay, Miki Herscovici, Yoelle Maarek,
Yael Petruschka, Aya Soffer
IBM Labs, Haifa University, Haifa 31905, Israel

1 Introduction

This is the first year that Juru, a Java IR system developed over the past few years at the IBM Research Lab in Haifa, participated in TREC's Web track. Our experiments focused on the ad-hoc tasks. The main goal of our experiments was to validate a novel pruning method, first presented at [1], that significantly reduces the size of the index with very little influence on the system's precision. By reducing the index size, it becomes feasible to index large text collections such as the Web track's data on low-end machines. Furthermore, using our method, Web search engines can significantly decrease the burden of storing or backing up extremely large indices by discarding entries that have almost no influence on search results.

In [1] we showed experimentally, using the LA-TIMES collection of TREC, that our pruning algorithm attains a very high degree of pruning with hardly any effect on precision. We showed that we are able to reduce the size of the index by 35% pruning with a slight decrease in average precision (7%), and with almost no effect on the precision of the top 10 results. For 50% pruning, we were still able to maintain the same precision at the top 10 results. Thereby, we obtained a greatly compressed index that gives answers that are essentially as good as those derived from the full index.

One important issue that was not addressed in our previous work dealt with the scalability of our pruning methods. In this work, we repeat our previous experiments on the larger domain of the Web Track data. While our previous experiments were conducted on a collection of 132,000 documents (476 MB), for the current experiments we built a core index for the entire Wt10g collection, (1.69M documents, 10GB). We then ran our pruning algorithms on the core index varying the amount of pruning to obtain a sequence of pruned indices. Section 4 describes the reuslts of the runs obtained from this sequence of indices. In order to be able to index such a large collection and retreive high quality results, we had to scale Juru's basic indexing and retrieval techniques as well as modify them to specifically handle Web data.

The rest of the paper is organized as follows: Section 2 describes Juru's main functionality. Section 3 briefly describes the pruning algorithms used for the experiments conducted in this work. Section 4 describes the experiments and the results we obtained. Section 5 concludes.

2. Juru's Main Functionality

Juru is a full-text search engine purely written in Java. Juru is based on the Guru search engine described in [2]. The major tasks performed by Juru are: (1) indexing large collections of documents, (2) formulating a query from a free-text query entered by the user, and (3) retrieving relevant documents for a given query and ranking them by order of relevance. In the following sections, we briefly describe the main techniques and algorithms embodied in Juru.

2.1 Profiling

Following the classical inverted index approach, Juru creates an index by associating terms with the documents that contain them, and then storing this mapping in inverted files for efficient retrieval. The first stage for creating this mapping is document parsing which constructs a canonical form called a *document* *profile*. Parsing includes HTML parsing, tokenization, lower case conversion, sentence splitting, and stemming. For HTML parsing Juru uses a parser that extracts from an HTML page the title, the set of out links, and the regular text that appears in the page. For tokenization and sentence splitting Juru uses procedures described in [3]. For stop-word filtering, the system uses a very short default list of stop-words but allows users to define their own special purpose list. Stemming is performed using Porter's stemmer [4]. A default list of proper names is managed by Juru and can be expanded by a special purpose proper name list provided by the user. A term identified as a proper name is not stemmed during profiling. The document profile can be described as a vector of terms, each one associated with its number of occurrences in the document.

For the experiments conducted on the Web Track data we also developed a special description database that provides for each page *p* in the collection a set of descriptions extracted from other pages that cite (i.e., link to) *p*. A description is defined as the anchor text associated with the link. Juru indexes every page based on its content as well as its set of descriptions. Simulations that we conducted on previous Web Track data show that using descriptions as indexing units for HTML pages improves precision by about 20%.

2.2 The indexing process

Documents are indexed as follows: when adding a document to the index, it is assigned an internal unique identifier (*id*). The document *id*, its name, its title, and any additional metadata are stored in a special database called the *document database*. Each term in the document profile is then inserted into a dictionary. The dictionary is managed using a trie data structure. Allow us to remind here that a trie is a search tree, where each node represents a word in a given lexicon or dictionary and each edge is associated with a sequence of characters. Sequencing the characters on the path from the trie root to a trie node composes the word associated with that node. As new words are added to the trie, new nodes are created, and old nodes (containing words) are possibly split.

Each term in the dictionary has a corresponding posting list – a list of all documents where it appears. Each entry in the posting list of term *t* consists of the document *id* of the document containing *t*, the number of occurrences of *t* in *d*, and a list of occurrence offsets of *t* in *d*. The posting lists are stored in a repository called the *repository database* and each term in the dictionary is associated with a pointer to its corresponding posting list. The posting lists are compressed using compression techniques described in [5]. Indexing of the entire collection of documents is carried out in a two-stage process:

Stage 1: creating a forward index:
Each document profile is split into sub-profiles, where each sub-profile contains all terms with a common prefix. For each prefix, all the corresponding sub-profiles from all documents are written to an appropriate forward index file. A forward file for a specific prefix holds for each document a list of all the document terms that begin with that prefix.

Stage 2: inverting the forward index:
After all the document profiles have been split into the forward index files, each file is traversed, using the following algorithm:

```
For each document d in the forward file
   For each term t in d
          If t is found in the dictionary
            retrieve the posting list of t, p(t), from the repository database
          else
            add t to the dictionary
            create a new posting list  p(t)
          add d with all occurrence information to  p(t)
          update p(t) in the repository database
```

The forward index structure allows us to keep only a small part of the dictionary in main memory during indexing, thus enabling index creation with limited memory resources. The dictionary is not required at all during the first stage of indexing, while the second stage requires only the part of the dictionary that corresponds to the prefix of the particular forward file being processed. In addition to restricting the amount of memory needed for indexing, forwarding can be done in parallel on several machines. The forward index technique also speeds up the indexing process by employing effective caching and buffering techniques.

2.3 Query Evaluation

Queries are treated as follows: a query profile is created using the same profiling method that is used for the full document collection during indexing. Recall that a profile is a vector of terms, where each term is associated with the number of occurrences of this term in the document/query. The following algorithm, based on the ranking process of the SMART system [6], describes the ranking process applied by Juru:

```
Input:   a query profile - $q = (t1,t2,..tk)$,
         the number of documents to retrieve - $N$
Output: the $N$ most relevant documents for the input query in document collection $D$

1. For each query term, retrieve its posting list from the repository database
2. Sort the query terms according to the length of their posting lists (handle infrequent terms first)
3. For each $id$ in collection $D$, $Score(id) = 0$
4. For each term $t$ in $q$ with posting-list $p(t)$
         for each posting entry $(d,OccNo(t,d))$ in $p(t)$
                  $Score(id) = Score(id) + tf(t,q) * tf(t,d) * idf(t)$    (*)
5. Normalize document scores by document length $|d|$
   For each $id$ in collection $D$,
                  $Score(id) = Score(id) / |d|$    (**)
6. Return $N$ documents with the highest scores.

(*) term frequency for profile $x$ ($x$ is $d$ or $q$):
                  $tf(t,x) = log(1+OccNo(t,x))/log(1+avg.OccNo(x))$
                           $OccNo(t,x)$ – number of occurrences of $t$ in $x$
                           $AvgOccNo(x)$ – average number of term occurrences in $x$
    inverse document frequency:
                  $idf(t) = log (|D|/|Dt|)$
                           $|D|$ - number of documents in the collection $D$
                           $|Dt|$ - number of documents containing $t$
(**) document length:
                  $|d| = (0.8*avgDocLength + 0.2*(\# \text{ of unique terms in } d))^{0.5}$
                  $avgDocLength$ – average number of unique terms per document in $D$
```

In order to optimize its query processing time, Juru applies dynamic pruning methods as described in [7]. The main idea is to order the query terms by decreasing weight, according to the length of their posting lists, and to process the infrequent terms first until some stopping condition is met. The algorithm marks each query term as infrequent, frequent, or very frequent. After assigning scores to a sufficient number of documents, very frequent terms are completely ignored. Only posting lists of infrequent terms are fully processed. Frequent terms contribute only to the scores of the documents that have already been scored previously.

2.4 Improving search precision by incorporating lexical affinities

Lexical affinities (*LAs*) were first introduced by Saussure in 1947 to represent the correlation between words co-occurring in a given language and then restricted to a given document for IR purposes [2]. LAs are identified by looking at pairs of words found in close proximity to each other. It has been described elsewhere [2] how LAs, when used as indexing units, improve precision of search by disambiguating terms. Juru's profiling component uses this technique as part of the profiling process to improve search precision by extracting lexical affinities from the text.

During query evaluation, the query profile is constructed to include the query's lexical affinities in addition to its individual terms. This is achieved by finding all pairs of words found close to each other in a window of some predefined small size (the sliding window is only defined within a sentence) based on the lexicographic evidence that 98% of LAs in the English language relate words that are in a distant of +/-5 words (see [2] for more details). For each *LA=(t1,t2),* Juru creates a pseudo posting list by merging the posting lists of *t1* and *t2*. It finds all documents in which these terms appear close to each other, and adds them to the posting list of the *LA* with all the relevant occurrence information. After creating the posting list, the new *LA* is treated by the retrieval algorithm as any other term in the query profile.

Lexical affinities can improve search results significantly, especially for short queries. The user can control whether to use LAs or not in the retrieval process. The user can also control the relative weight between keywords and *LA*s, thus giving more (or less) significance to *LAs* in relation to simple keywords in computing the relevance score. Figure 1 shows the relation between the system's precision and the relative weight given to *LAs* and to simple keywords. For a zero weight the queries are constructed with no *LAs*. For a weight of one the queries consist of *LAs* only (i.e., keywords are ignored). The experiments were done on the Wt10g collection with ad-hoc topics 501-550. Queries were formulated from the topic's title. From the graph we can see that the optimal relative weight is 0.3 for precision at 10 and 0.5 for avg. precision. For the experiments at TREC 10 described in this paper the LA weight was fixed to 0.3.

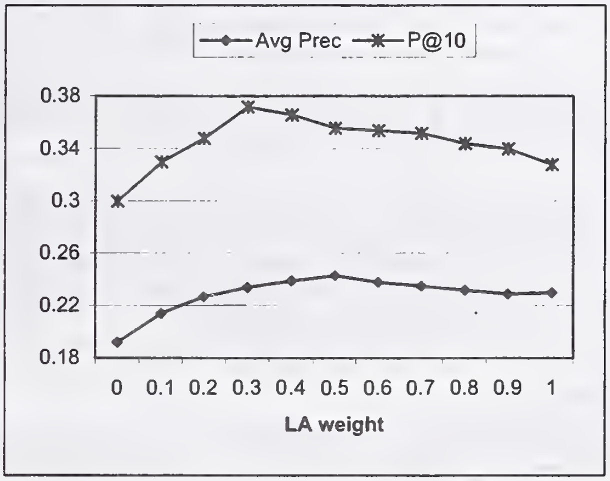


Figure 1: relation between the system's precision and the relative weight given to *LAs*

3 Index Pruning

Indexing a large collection of documents might result in extremely large index files that are difficult to maintain. Therefore, it is important to utilize efficient compression methods for index files. There are two complementary approaches: *lossless compression* and *lossy compression*. Lossless approaches do not lose any information; instead, they use more efficient data structures. Thus, under lossless approaches, posting lists have a very compact representation. On the other hand, under lossy approaches, certain information is discarded.

We propose here lossy methods that prune the index at the posting level. That is, in our approach, a term can be retained in the index, but some document postings may be eliminated from this term's posting list. The idea is to remove those postings whose potential contribution to the relevance score of a document is so small that their removal will have little effect on the accuracy of the system. The selection of which document postings to prune is guided by certain user-specified parameters.

Our goal is to perform index pruning in such a way that a human "cannot distinguish the difference" between the results of a search engine whose index is pruned and one whose index is not pruned. Therefore, as in any lossy compression technique, we wish to remove the least important terms from the index, so that the visible effects of the compression (in terms of the results obtained) are very small. Thus, the question we need to address is how to identify the least important entries in the index.

We begin with the usual assumption that for each query, there is a *scoring function* that assigns a score to each document, so that the documents with the highest scores are the most relevant. The scoring function is often based on a 2-dimensional scoring table, A, indexed by terms and documents. Table entries are set according to the scoring model of the search engine; thus, $A(t,d)$ is the score of document d for term t.

The first static pruning algorithm that we consider removes from the index all posting entries whose corresponding table values are bounded above by some fixed cutoff threshold. We refer to this type of pruning as *uniform pruning*, since the threshold is uniformly chosen, with the same cutoff value being used for every term. Uniform pruning has an obvious drawback. Low-scoring terms may have all of their entries in the index pruned away. Therefore, given a query consisting only of low-scoring terms, the pruned index may fail to provide any good results for this query.

This insight leads us to propose a second, and more sophisticated, pruning algorithm, in which the cutoff threshold may depend on the term. We refer to this type of pruning as *term-based pruning*. Term-based pruning guarantees that each term will have some representative postings left in the index. Therefore, queries with low-scoring terms will fare better than under uniform pruning. How do we determine the cutoff thresholds? We are guided by the intuition that all we really care about are the top k documents, since this is all the user sees. Thus, we care only about whether the pruned index returns the same top k documents; we do not care about the score it might assign to the remaining documents. Our term-based pruning algorithm attempts to minimize the effect of pruning on the top k results for each query.

Recall that the scoring table is not stored as such in the Juru index. Instead, each term is stored with an associated posting list. The following algorithm describes how to prune a given inverted file using the top k pruning algorithm. The algorithm takes as input an inverted file I, along with the parameters k and ε, and creates a pruned inverted file. Note that the entries of the scoring table A are computed on a term-by-term basis in order to find the cutoff value for each particular posting list.

```
Top k prune($I,k,\varepsilon$)
    For each term $t$ in $I$
        Retrieve the posting list $P_t$ from $I$
        If $|P_t| > k$
            For each entry $d$ in $P_t$
                Compute $A(t,d)$ according to the scoring model
                Let $z_t$ be the $k$th best entry in row $t$ of $A$
                $\tau_t = \varepsilon * z_t$
                For each entry $d$ in $P_t$
                    If $A(t,d) < \tau_t$ remove entry $d$ from $P_t$
        Save ($t$, $P_t$) in the pruned inverted file
```

The time complexity of the pruning algorithm is linearly proportional to the index size. For each term *t*, the algorithm first computes a threshold by finding the *k*th best entry in the posting list of *t* (this can be done in *O(N)* time, where *N* is the number of documents). It then scans the posting list to prune all the entries smaller than the threshold. Thus, if there are *M* terms in the index, the time complexity of the algorithm is *O(M*N)*.

In [1] we gave a formal proof that for all queries with a moderate number of search terms, the results obtained from the pruned index are indistinguishable from those obtained from the original index.

4 Experimental Results

Our experiments tested the impact of pruning on the search results. First we created a sequence of pruned indices using the uniform pruning algorithm where we varied τ, the cutoff threshold. Next we created a sequence of pruned indices by invoking the term-based pruning algorithm, where we fixed *k* to 10, and used varying values of ε. For each index we ran 50 queries, constructed automatically from the titles of topics 501-550. Our first experiment tested the effect of pruning on the similarity of the top results to the top results of the original index. The second tested the effect of pruning on the precision of the results.

4.1 The effect of pruning on similarity

The similarity of the top results was measured by two metrics. First, the *symetric difference* between the top 10 lists that evaluates the similarity between two lists by considering the common presence/absence of items in both lists. Second, a variation of *Kendall's tau* measure that considers not only the common presence/absence of items in the lists but also their rank. The symmetric difference is evaluated as follows: if *y* is the size of the union of the two top 10 lists, and *x* is the size of the symmetric difference, then we take the symmetric difference score to be $1 - x/y$. This score lies between 0 and 1. The highest score of 1 occurs precisely when both lists are identical (although the order of results may be different), and the lowest score of 0 occurs precisely when the two lists are disjoint.

The second metric we used is a variation of Kendall's tau method that was obtained in [8] and used in [1]. The original Kendall's tau method for comparing two permutations assigns a penalty for each pair of distinct items for which one item appears before the second in one permutation and the second appears before the first in the other permutation. The sum of the penalties over all pairs reflects the overall penalty. The modified version of Kendall's tau handles the case where we care about, comparing the top 10 in one list against the top 10 in another list, rather than comparing permutations. The penalties assigned for each pair of distinct items are redefined, since two distinct items might not appear in the top 10 of one or both lists. By normalizing the sum of penalties to lie between 0 and 1, the highest score of 1 occurs precisely when both lists are the same and in the same order, and the lowest score of 0 occurs precisely when the two lists are disjoint. More details appear in [1,8].

Figure 2 shows the similarity between the top 10 results of the original index and the pruned index, at varying levels of pruning, for both uniform pruning and term-based pruning. The relatively high similarity between the top 10 lists for moderate pruning levels supports the claim that the top 10 results of the pruned indices are very similar to the top 10 results of the original index. Surprisingly, in contrast to our previous results [10], there is no advantage for term-based pruning over uniform pruning. Both algorithms create pruned indices with similar behavior in terms result similarity.

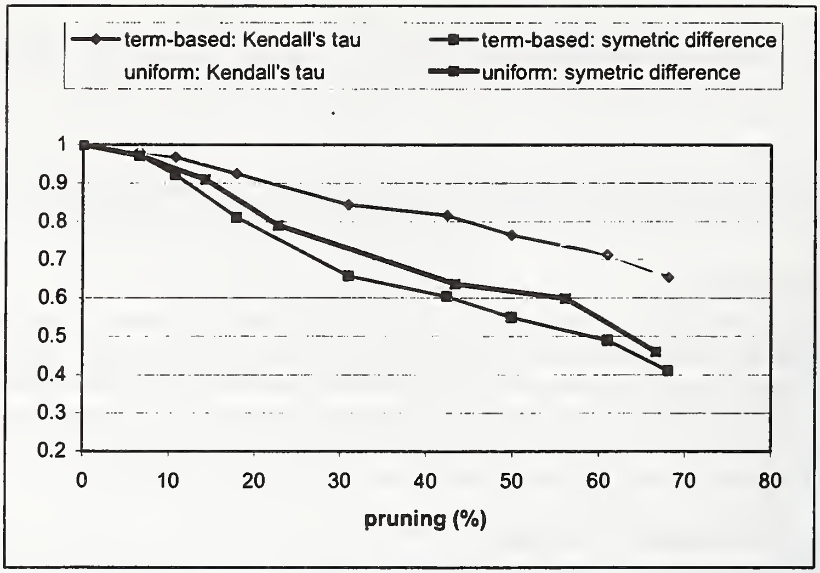


Figure 2: similarity as a function of pruning

4.2 The effect of pruning on precision

In order to measure the effect of pruning on precision we had to wait for TREC's official results. Four runs were submitted to TREC for evaluation. The first run consists of the results obtained from Juru's core index. The second and the third runs were obtained from two pruned indices created by the term-based pruning algorithm with parameters $k = 10$, $\varepsilon = 0.05$ (10.7% pruning) and $k = 10$, $\varepsilon = 0.1$ (17.8% pruning), respectively. The fourth run consists of the results of an experiment we performed with query expansion. Our expansion method failed to improve search performance, thus we ignore it in this report.

The following table shows the precision of the official runs submitted to TREC. The results support our claim that P@10 is barely affected for short queries even after significant pruning. Furthermore, while there is some loss in the mean average precision (MAP), it is negligible.

ε	Index size	Pruning (%)	MAP	P@10
0	3.53 GB	0	0.211	0.362
0.05	3.15 GB	10.7	0.207	0.362
0.1	2.9 GB	17.8	0.205	0.360

Figure 3 shows P@10 results obtained from the core index and the pruned indices for all the Web Track ad-hoc queries. For most of the queries (35 queries), P@10 remained the same. For 8 of the queries, it even improved and only 7 queries exhibited some loss in precision, where the largest loss is 0.2 (for query 530).

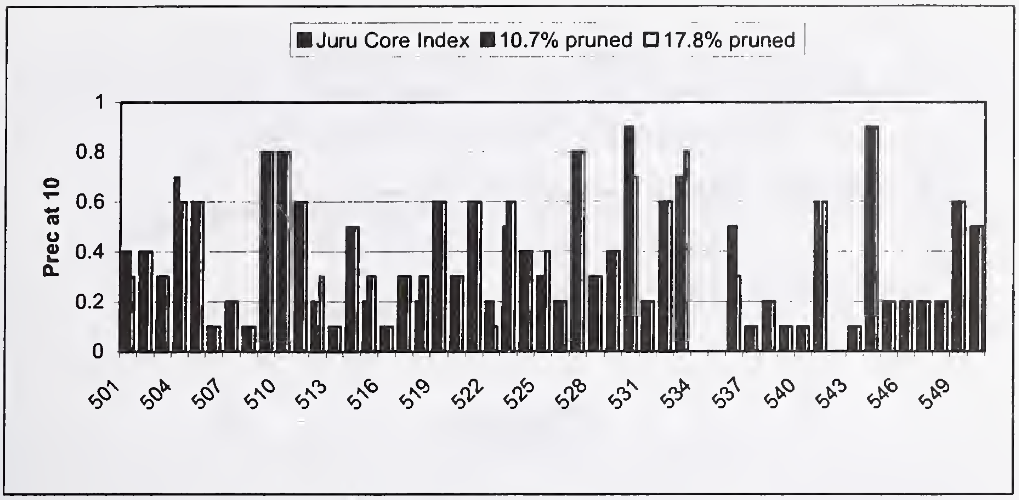


Figure 3: P@10 of topics 501-550 for the three official runs

Figure 4 shows the MAP for all queries obtained from the three runs. Although for many queries there is some decrease in precision, the drecrease is quite small (maximum of 16% loss for topic 504).

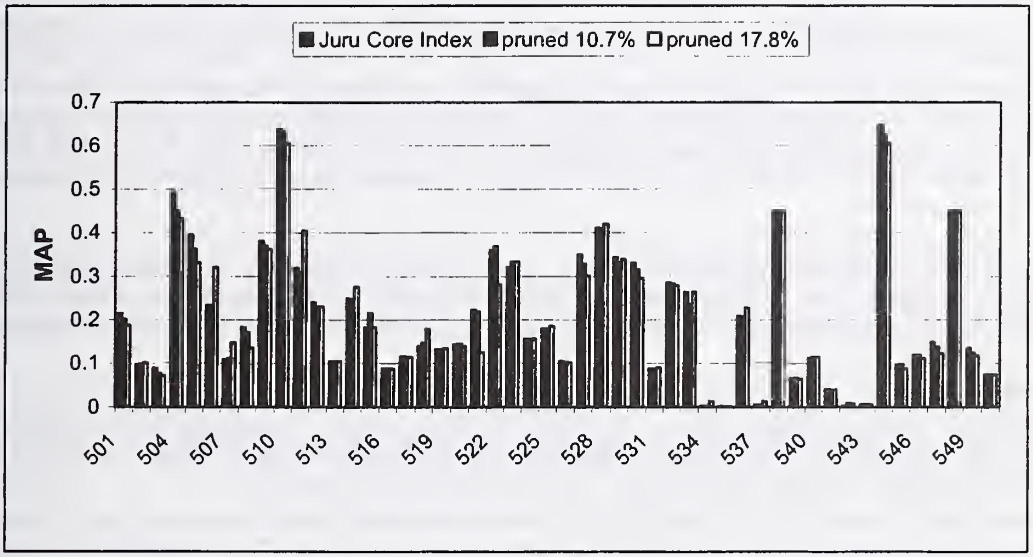


Figure 4: MAP of topics 501-550 of the three official runs

After receiving TREC's official results, we repeated the experiments with several additional indices created by our pruning algorithms. The goal was to test how much further we could prune the indices, before significant loss in precision occurs. We used the newly published "qrels" files to measure precision. Figure 5 shows the impact of pruning on precision as measured by MAP and P@10. From these tests, it is apparent that P@10 remains more or less stable up to 40% pruning. There is a slight decrease in average precision at 30% pruning, but a significant loss of MAP also occurs only at 40% pruning. As for the similarity experimental results, in contrast to our previous results [10], there is no advantage for term-based pruning over uniform pruning.

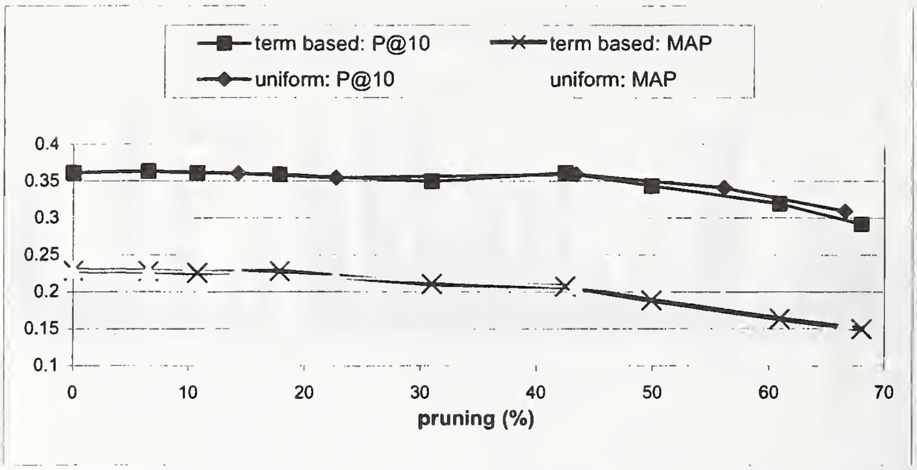


Figure 5: the impact of pruning on precision as measured by MAP and P@10.

5 . Summary

The results we obtained from the TREC experiments support and corroborate our previous results with index pruning: it is possible to discard a significant part of the index, and still attain answers that are almost as good as those obtained from the full index. By reducing the index size, it becomes feasible to index large text collections such as the Web track's data on low-end machines. Furthermore, using our method, Web search engines can significantly decrease the burden of storing extremely large indices by removing entries that have no influence on search results. Our experiments show that we can reduce the index size by 40% while attaining the same P@10 (which is most critical for Web search engines), and with only a slight decrease in the mean average precision.

In addition to validating the pruning methods, the Web Track results also demonstrate the overall high quality of the Juru search engine. From the initial results available at this point, it is apparent that Juru is significantly above the median of results attained by all participants for most of the queries as well as in overall precision.

References

[1] D. Carmel, D. Cohen, R. Fagin, E. Farchi, M. Herscovici. Y. Maarek, A. Soffer. Static Index Pruning for Information Retrieval Systems. Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 43-50, New Orleans, 2001.

[2] Y. Maarek and F. Smadja. Full text indexing based on lexical relations: An application: Software libraries. Proceedings of the 12th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 198-206, 1989.

[3] G. Grefenstette and P. Tapainen. What is a word? What is a sentence? Problems of Tokenization. In Proceedings of the 3rd International Conference on Computational Lexicography (COMPLEX '94), Research Institute for Linguistics, Hungarian Academy of Sciences, Budapest, 1994.

[4] M. F. Porter. An Algorithm for Suffix Stripping. Program 14(3), pages 130 – 137, 1980.

[5] I. H. Witten, A. Moffat, and T. C. Bell. Managing Gigabytes, Morgan Kaufman Publishers, San Francisco, California, 1999.

[6] C. Buckley and A. Singhal and M. Mitra and G. Salton. New retrieval approaches using SMART: TREC 4. Proceedings of the Fourth Text REtrieval Conference (TREC-4), pages 25 – 48, 1995

[7] D. Harman and G. Candela. Retrieving records from a gigabyte of text on a minicomputer using statistical ranking. Journal of the American Society of Information Science 41(8), pages 581 – 589, 1990.

[8] R. Fagin, R. Kumar, D. Sivakumar. Top *k* orderings and near metrics. To appear, 2002.

Integrating Link Structure and Content Information for Ranking Web Documents

Tapas Kanungo and Jason Y. Zien
IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120
Email: kanungo,jasonz@almaden.ibm.com

SUMMARY

In this article we describe the results of our experiments on combining the two sources of information: The web link structure and the document content. We developed a new scoring function that combines TF*IDF scoring with the document rank and show that it is particularly effective in the Home Page Finding task.

1 The Indexer

The IBM Almaden system consists of three components: the indexer, the DocRanker, and the query engine.

- The indexer tokenizes the documents and records the presence of various attributes: capitalization, presence in title or bulleted lists, color, etc. Prior to pushing the attributed token into the index, the token is stemmed using the Porter stemmer. Document frequency of tokens and other global statistics are also recorded on the fly.
- The DocRanker ranks documents by extracting the link structure of the crawled pages, then computing the SameSite function [2], and finally computing the PageRank [3], on the graph. Our PageRank calculations used a weighted graph where the weight of a link was 1.0 if the link was composed of two pages from different sites, and 0.0001 if they were on the same site, to deemphasize self-references.
- The query engine retrieves the filtered set of documents that contain the the rarest query term. Then, all the filtered documents are scored using all of the query terms (both stemmed and unstemmed versions). These pages are ranked using the integrated ranking function and then sorted result is returned.

2 Scoring Function Details

The first part in our scoring algorithm is deciding which documents to filter out. Our scoring algorithm uses the rarest term to drive the query evaluation. Only documents which contain the rarest term are examined during scoring.

Let $w_{d,t}$ be the score of a document with respect to term t. The general form of a TF*IDF scoring function looks like this:

$$w_{d,t} = r_{d,t} \cdot w_t \tag{1}$$

where $r_{d,t}$ (the TF component) is a function based on the frequency of a term t in document d, and w_t (the IDF component) is the weight of a term in the corpus.

2.1 Inquery/Okapi

The Inquery scoring function is a variation of the well-known Okapi scoring function. Let A be the average length of a document in terms (not bytes), and $|D_d|$ be the length of document d in terms (not bytes), and N be the number of documents in the collection. Let $f_{d,t}$ be the number of occurrences of t in a document d and f_t be the number of documents in which t occurs. Give a query Q, the Inquery scoring formula is:

$$score = \sum_{t \in Q \cap D_d} TF \cdot IDF \tag{2}$$

$$score = \sum_{t \in Q \cap D_d} r_{d,t} \cdot w_t \tag{3}$$

$$r_{d,t} = 0.4 + 0.6 \frac{f_{d,t}}{f_{d,t} + 0.5 + 1.5 \frac{|D_d|}{A}} \tag{4}$$

$$w_t = \frac{log_e(\frac{N+0.5}{f_t})}{log_e N + 1} \tag{5}$$

In order to take advantage of document contextual information, we modified $f_{d,t}$ so that it is not just the number of occurrences of t in d, but instead, is a weighted sum of occurrences of t in d. The weighting of occurrences in titles and headings, was configurable. Also of note, our $f_{d,t}$ includes both stemmed and unstemmed occurrences of a term. The effect of this is to essentially boost the score for pages that have an exact match, while also giving a chance for pages that have only the stemmed term to appear in the result set.

2.2 Incorporating DocRank into the Scoring Function

Link analysis methods may be used to obtain an ordered ranking of documents, for instance, through a PageRank calculation. Although PageRank provides useful information for scoring, it is unclear how this information should be combined into a page-content-based scoring function. We propose a new scoring function that blends document ranking with a TF*IDF formulation. Let us examine the Inquery/Okapi function's TF component in detail. The component $1.5\frac{|D_d|}{A}$ is the only portion of the function that contains document-related information. This component provides a bias to the score based on the importance of a document. In this case, importance is defined by the size of a document. When a document is large, this score component is large (and hence since this is in the denominator, the overall score is reduced). We propose incorporating a document rank (DocRank) ρ_d into this component. ρ_d is the scaled ordinal rank of a page. For instance, if the document is the the 3^{rd} ranked document in a collection of N documents, $\rho_d = 3/N$. Note that in particular, the DocRank is not the actual PageRank value. Rather, the use of ordinal rank provides a smoother, more gradually changing value than the actual PageRank. Also, it is obvious that any algorithm that can generate an ordering of documents could be used in place of PageRank for our purposes. There are two straightforward ways of combining the DocRank with the document component of the score — multiplicative and additive. An example of the multiplicative form is:

$$1.5\rho_d \frac{|D_d|}{A}. \tag{6}$$

However, after experimentation, the form that we settled on is the additive form:

$$\alpha\rho_d + 1.5\frac{|D_d|}{A}, \tag{7}$$

where α is a user-specified constant. The α term allows the user to tune the relative importance of the document ranking component in the scoring function. For the Home Page Finding task, we used $\alpha = 10.0$. For the Ad Hoc task, we used $\alpha = 1.5$

Our final modified scoring formula is:

$$score = \sum_{t \in Q \cap D_d} TF \cdot IDF \tag{8}$$

$$r_{d,t} = 0.4 + 0.6 \frac{f_{d,t}}{f_{d,t} + 0.5 + \alpha\rho_d + 1.5\frac{|D_d|}{A}} \quad (9)$$

$$w_t = \frac{log_e(\frac{N+0.5}{f_t})}{log_e N + 1} \quad (10)$$

3 Results at TREC 2001

We participated in the two Web tracks at TREC 2001: Ad Hoc, and Home Page Finding. Our contribution is a method of incorporating a DocRank term into a TF*IDF cost function that allows us to control the relative contribution of a document's rank to that of text content. The ranking function itself is based on the Inquery variant of the Okapi ranking function [1].

Also, to take advantage of contextual cues on a page, we made use of heading and title information by giving more weight to term occurrences in those contexts.

Results for the Home Page Finding task:

Metric	Rank not used	Rank used
Average reciprocal rank over 145 topics	0.382	0.611
Number of topics for which entry pages found in top 10	90 (62.1%)	113 (77.9%)
Number of topics for which no entry pages was found	17 (11.7%)	15 (10.3%)

The Home Page Finding task shows clearly that when our DocRank scoring is used, both the average reciprocal rank and the top ten scoring method showed substantial improvement. Using linkage information was a clear win with Home Page Finding.

Results for the Ad Hoc task:

Metric	Rank not used	Rank used
Precision at 5 docs	0.40	0.20
Precision at 10 docs	0.20	0.20
Precision at 15 docs	0.133	0.20

For Ad Hoc queries, link information did not improve the results.

4 Conclusion

We introduced a novel new scoring function that combines TF*IDF scoring with link-based ranking. Our experiments showed that this combined scoring method was exceptionally well-suited to Home Page Finding.

References

[1] J. Allan, M. Connell, W. B. Croft, F. F Feng, D. Fisher, and X. Li. INQUERY and TREC-9. 2000.

[2] Soumen Chakrabarti, Byron Dom, David Gibson, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Spectral filtering for resource discovery. In *Proceedings of the SIGIR 98 Workshop on Hypertext Information Retrieval for the Web*, August 1998.

[3] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, Stanford, CA, November 1999. http://dbpubs.stanford.edu/pub/1999-66.

Integrating Features, Models, and Semantics for TREC Video Retrieval

John R. Smith†, Savitha Srinivasan‡, Arnon Amir‡, Sankar Basu†, Giri Iyengar†,
Ching-Yung Lin†, Milind Naphade†, Dulce Ponceleon‡, Belle Tseng†
†IBM T. J. Watson Research Center, 30 Saw Mill River Road, Hawthorne, NY 10532 USA
‡IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120 USA

Abstract

In this paper, we describe a system for automatic and interactive content-based retrieval of video that integrates features, models, and semantics. The novelty of the approach lies in the (1) semi-automatic construction of models of scenes, events, and objects from feature descriptors, and (2) integration of content-based and model-based querying in the search process. We describe several approaches for integration including iterative filtering, score aggregation, and relevance feedback searching. We describe our effort of applying the content-based retrieval system to the TREC video retrieval benchmark.

1 Introduction

The growing amounts of digital video are driving the need for more effective methods for storing, searching, and retrieving video based on its content. Recent advances in content analysis, automatic feature extraction, and classification are improving capabilities for effectively searching and filtering digital video using information based on perceptual features, content structure, models, and semantics. The emerging MPEG-7 multimedia content description standard promises to further improve content-based searching by providing a rich set of standardized tools for describing multimedia content in XML [SS01]. However, MPEG-7 does not standardize methods for extracting descriptions nor for matching and searching. The extraction and use of MPEG-7 descriptions remains a challenge for future research, innovation, and industry competition [Smi01].

In this paper, we describe a system for automatic and interactive content-based retrieval that integrates features, models, and semantics [SBL$^+$01]. The system analyzes the video by segmenting it into shots, selecting key-frames, and extracting audio-visual descriptors from the shots. This allows the video to be searched at the shot-level using content-based retrieval approaches. However, we further analyze the video by developing and applying models for classifying content. The approach requires the manual- or semi-automatic annotation of the video shots to provide training data. The models are subsequently used to automatically assign semantic labels to the video shots. In order to apply a small number of models but have at the same time to have large impact on classifying the video shots, we have primarily investigated models that apply broadly to video content, such as indoor *vs.* outdoor, nature *vs.* man-made, face detection, sky, land, water, and greenery. However, we have also investigated several specific models including airplanes, rockets, fire, and boats. While the models allow the video content to be annotated automatically using this small vocabulary, the integration of the different search methods together (content-based and model-based) allows more effective retrieval.

In the paper, we describe the approach for integrating features, models, and semantics in a system for content-based retrieval of video. We have applied these systems and methods to the NIST TREC video retrieval benchmark, which consists of 74 queries of a video corpus containing approximately 11 hours of video. The queries, which were designed to access video based on semantic contents, permit automatic and/or interactive approaches for retrieving the results. We enhance the automatic retrieval by using the models in conjunction with the features to match the query content with the target video shots. For interactive retrieval, we allow the user to apply several methods of iterative searching that combines features, semantics, and models using different filtering operations and weighting methods. In this paper, we describe more details about the approach and discuss results for the TREC video retrieval benchmark.

2 Content analysis system

The video content is analyzed through several processes that involve shot detection, feature extraction, and classification, as shown in Figure 1. The video is segmented temporally according to shot boundaries, and descriptors are extracted for each shot. The descriptors are ingested into a storage system. The descriptors are used as input into the model-based classification system which assigns semantic labels to each shot. The system also ingests any meta-data related to the content such as title, format, source, and so forth.

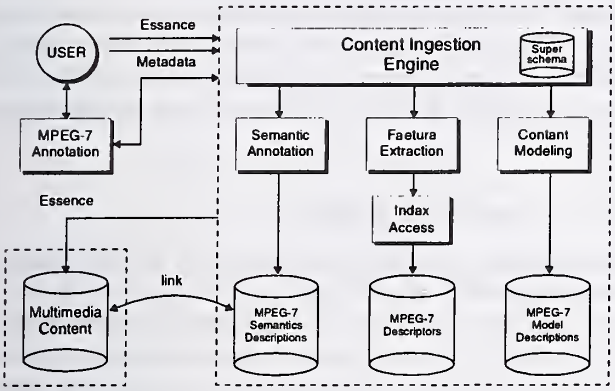


Figure 1: The video content ingestion engine first segments the video temporally using shot detection and selects key-frames, then extracts descriptors of the audio-visual features and applies models in order to classify the content.

2.1 Shot detection

The video content is pre-processed by splitting it into temporal segments using the *IBM CueVideo* (program `cuts.exe` with the default settings) [Cue]. After the shots are detected, key-frames are selected and extracted, and all MPEG I-frames are extracted, as shown in Figure 2. These images are stored and indexed and are used for accessing the shots.

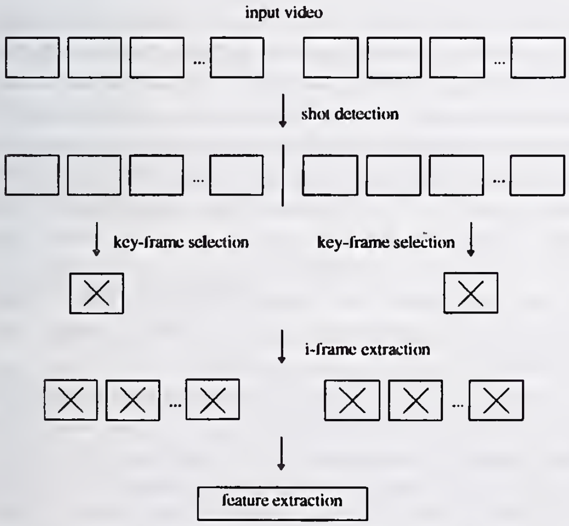


Figure 2: The shot detection system automatically segments the video into temporal segments and selects a key-frame for each shot.

CueVideo 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 frames. Frame pairs at one, three and seven frames apart and their corresponding thresholds are shown by the three upper graphs in Figure 3. Statistics of frame differences are computed in a moving window around the processed frame and are used to compute the adaptive thresholds. Hence the program does not require sensitivity-tuning parameters.

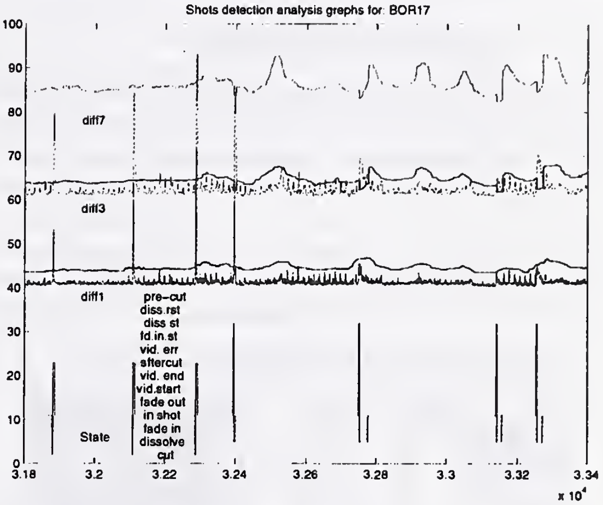


Figure 3: This example represents a 53 Seconds sequence with four cuts and three dissolves in high noise (from *bor17.mpg*, frame numbers: 31800–33400). The middle cut is mistakenly detected as a short dissolve (Alm2).

A state machine is used to detect and classify the different shot boundaries, shown at the botom of Figure 3 with all thirteen states listed. At each frame a state transition is made from the current state to the next state, and any required operation is taken (e.g., report a shot, save a key-frame to file). The algorithm classifies shot boundaries into Cuts, Fade-in, Fade-out, Dissolve and Other. It works in a single pass, is robust to possibly uncompliant MPEG streams, and runs about 2X real time on a 800MHz P-III.

2.2 Feature extraction

The system extracts several different descriptors for each of the key-frames and i-frames. We have used the following descriptors:

1. color histogram (166 bin HSV color space),
2. grid-based color histogram (4x4 grid of the HSV histogram),
3. texture spatial-frequency energy (variance measure of

each of 12 bands of quadrature mirror filter wavelet decomposition, and

4. edge histogram (using Sobel filter and quantization to 8 angles and 8 magnitudes).

Each of these descriptors is stored and indexed separately. However, at retrieval time, the CBR matching function allows the descriptor values to be combined using an arbitrary weighting function in order to determine the similarity of the query and target images based on multiple features.

2.3 Semi-automatic annotation

In order to allow a model-based approach to video retrieval, ground-truth data is needed for training the models. In order to create training data, we developed a video annotation tool that allows the users to annotate each shot in the video sequence, as shown in Figure 4. The tool allows the user to identify and label scenes, events, and object by applying the labels at the shot-level. The tool also allows the user to associate object-labels with individual regions in a key-frame.

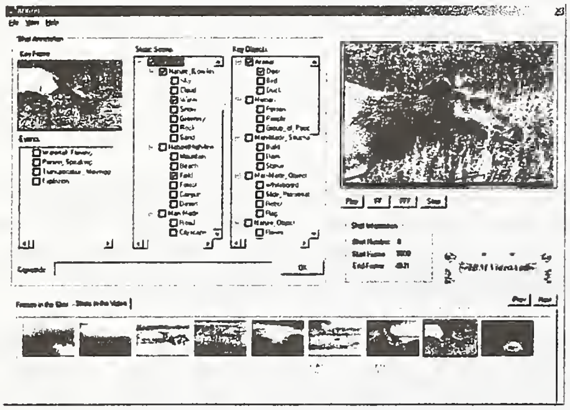


Figure 4: The video annotation tool allows users to label the events, scenes, and objects in the video shots.

For annotating video content, we created a lexicon for describing events, scenes, and objects; the following excerpt gives some of the annotation terms:

- **Events**: water skiing, boat sailing, person speaking, landing, take-off/launch, and explosion;
- **Scenes**: outer space (moon, mars), indoors (classroom, meeting room, laboratory, factory), outdoors (nature, sky, clouds, water, snow, greenery, rocks, land, mountain, beach, field, forest, canyon, desert, waterfall), and man-made (road, cityscape);
- **Objects**: non-rigid objects (animal, deer, bird, duck, human), rigid objects (man-made structure, building, dam, statue, tree, flower), transportation (rocket, space shuttle, vehicle, car, truck, rover, tractor), and astronomy.

The video anntotation tool allows the user to process the video shot-by-shot, and assign the labels to each shot. The tool is semi-automatic in that it automatically propagates labels to "similar" shots as described in [NLS+02]. The system requires the user to confirm or reject the propagated labels.

2.4 Content modeling

The content modeling system uses the labeled training video content to classify other video content (in our case, the test TREC video corpus). We have investigated several different types of static models including Bayes nets, multinets [NKHR00], and Gaussian mixture models. In some cases, we have used additional descriptors in the models, which are not applied for content-based retrieval, such as motion activity and color moments.

We have developed statistical models for the following concepts:

- **Events**: fire, smoke, launch;
- **Scenes**: greenery, land, outdoors, rock, sand, sky, water;
- **Objects**: airplane, boat, rocket, vehicle.

2.4.1 Statistical modeling

In the statistical modeling approach, the descriptors extracted from the video content are modeled by a multidimensional random variable X. 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. In particular, each semantic concept is represented by a binary random variable. The two hypotheses associated with each such variable are denoted by H_i, $i \in \{0, 1\}$, where 0 denotes absence and 1 denotes presence of the concept. Under each hypothesis, we assume that the descriptor values are generated by the conditional probability density function $P_i(X)$, $i \in \{0, 1\}$.

In case of scenes, we use static descriptors that represent the features of each key-frame. In case of events, which have temporal characteristics, we construct temporal descriptors using time series of static descriptors over the multiple video frames. We use a *one-zero* loss function [Poo99] to penalize incorrect detection. This is shown in Equation 1:

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{otherwise} \end{cases} \qquad (1)$$

The risk corresponding to this loss function is equal to the average probability of error and the conditional risk with action α_i is $1 - P(\omega_i|x)$. To minimize the average probability of error, class ω_i must be chosen, which corresponds to the maximum a posteriori probability $P(\omega_i|x)$. This corresponds to the minimum probability of error (MPE) rule.

In the special case of binary classification, the MPE rule can be expressed as deciding in favor of ω_1 if

$$\frac{p(x|\omega_1)}{p(x|\omega_2)} > \frac{(\lambda_{12} - \lambda_{22})P(\omega_2)}{(\lambda_{21} - \lambda_{11})P(\omega_1)} \tag{2}$$

The term $p(x|\omega_j)$ is the *likelihood* of ω_j and the test based on the ratio in Equation (2) is called the *likelihood ratio test* (LRT) [DH73, Poo99].

2.4.2 Parameter estimation

For modeling the TREC video content, we assume that the conditional distributions over the descriptors X under the two hypotheses – concept present (H_1) and concept absent (H_0) – have been generated by distinct mixtures of diagonal Gaussians. The modeling of these semantic concepts involves the estimation of the unknown but determinsitic parameters of these Gaussian mixture models (GMMs) using the set of annotated examples in the training set. For this purpose the descriptors associated with training data corresponding to each label are modeled by a mixture of five gaussians. The parameters (mean, covariance, and mixture weights) are estimated by using the Expectation Maximization (EM) [DLR77] algorithm.

The rest of the training data is used to build a negative model for each label in a similar way, which corresponds to a garbage model for that label. The LRT is used in each test case to determine which of the two hypotheses is more likely to account for the descriptor values. The likelihood ratio can also be looked upon as a measure of the *confidence* of classifying a test image to the labeled class under consideration. A ranked list of confidence measures for each of the labels can be produced by repeating this procedure for all the labels under consideration.

2.4.3 Region merging

We use manually assigned bounding boxes encompassing regions of interest obtained during annotations for extracting features. The testing is also done at the regional bounding box level. To fuse decisions from several bounding boxes in a key-frame, we use the following hypothesis: If a concept is to be declared absent in a frame, it must be absent in each and every bounding box tested. We can then compute the product of the probability of the "concept absent" hypothesis to obtain the probability of the concept being absent in the frame. Alternately, we can also use the maximum possible probability of the concept being detected in any region as the probability of its occurrence in the image/frame. For concepts which are global in terms of feature support, this step is not needed. Localized or regional concepts include *rocket*, *face*, *sky*, and so forth.

2.4.4 Feature fusion

The objective of feature fusion is to combine multiple statistical models for the different video features. Separate GMM models are used for each of the different descriptors (e.g., color histogram, edge direction histogram, texture, and so forth). This results in separate classifications and associated confidence for each test image depending on the descriptor. While the classifiers can be combined in a many ways, we explored straightforward methods such as taking sum, maximum,and product of the individual confidences for each descriptor in computing an overall classification confidence.

While this strategy of "late feature fusion" is fairly simple, one can envision other "early feature fusion" methods such as concatenating different descriptors into a single vector and then building a single GMM. We did not pursue this strategy due to the large dimensionality of the descriptors, especially in view of the paucity of training video content depicting the concepts of interest. However, it may be possible to consider discrimination in reduced dimensional subspaces of the feature space by using techniques such as the principal component analysis (PCA) or by using more sophisticated dimensionality reduction techniques that would allow concatenation and modeling of high-dimensional descriptors.

2.4.5 Training

The performance of statistical models such as the GMM depend to a large extent on the amount of training data. Due to the relatively small amount of labeled training video data beyond the TREC video corpus, we adopted a "leave one clip out strategy." This means that we trained a model for each concept as many number of times as the number of video clips. During each such training, one clip was left out from the training set. The models for the two hypotheses thus trained were used to detect the semantic concept in the clip that was left out.

2.5 Speech indexing

In addition to automatic analysis and modeling of the features of the video content, we also investigated the use of speech indexing as an alternative approach for video retrieval [PS01]. We used the *IBM ViaVoice* speech recognition engine to transcribe the audio and generate a continuous stream of words. We define a unit-document to be a 100 word temporal segment where consecutive segments overlap partially in order to address the boundary truncation effect. There are several operations performed in sequence in this processing.

First, the words and times from the recognizer output are extracted to create the unit-document files with associated timestamps. The Julian time at the start of the audio is used as the reference basis. This is followed by tokenization to detect sentence/phrase boundaries and then part-of-speech tagging such as noun phrase, plural noun etc. The morphological analysis uses the part-of-speech tag and a morph dictionary to reduce each word to its morph. For example, the verbs, lands, landing and land will all be reduced to land. Then, the stop words are removed using a standard stop-words list. For each of the remaining words, the number of unit-documents that it belongs to (the inverse document frequency) is computed and is used to weight these word.

3 Video retrieval

Once the video content is ingested, the descriptors and model results are stored and indexed. This allows the user to carry out the searches in a video query pipeline process as shown in Figure 6, in which queries are processed in a multi-stage search in which the user selects models and clusters or examples of video content at each stage. By operating on the interim results, the user controls the query refinement. As shown in Figure 6, at each stage of the search, a query Q_i produces a result list R_i. The result list R_i is then used as input into a subsequent query Q_{i+1}, and through various selectable operations for combining and scoring R_i with the matches for Q_{i+1}, the result list R_{i+1} is produced. The user can continue this iterative search process until the desired video content is retrieved.

3.1 Content-based retrieval

Content-based retrieval is the most amenable to automatic retrieval in the case that the query provides example content. For TREC video retrieval, each of the queries provided example content which included anywhere from a single image to several video clips. For automatic content-based retrieval, the following approach was adopted: the query content was analyzed using shot detection, key-frame selection, and feature extraction to produce a set of descriptors of the query content. Then, the query descriptors were matched against the target descriptors. We considered two approaches for automatic content-based matching: (1) matching of descriptors of the query and target key-frames, and (2) matching of descriptors for multiple frames (i-frames) from the query and target video, as shown in Figure 5.

3.1.1 Multi-frame matching

For multi-frame matching, different semantics of the matching are possible depending on the nature of the query. For example, if all of the individual images in the query content are important for the query ("all" semantics), then the matching semantics is such that the best target video shot from the database should have the best overall score of matching all of the query images to images in the target shot.

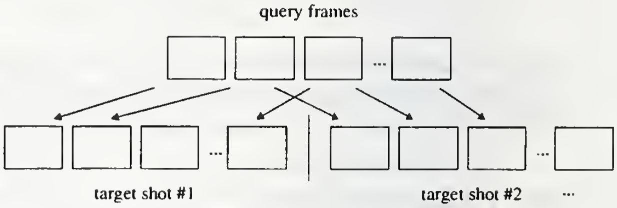


Figure 5: Content-based retrieval matches multiple query frames against multiple frames in the target shots.

Multi-frame matching requires first the determination of the best matches among individual images from the query and target, and then computation of the overall score of all the matches. However, alternatively, if the query images are meant to illustrate different variations of the content ("or" semantics), then the matching semantics is such that the best target video should be the ones that have a single frame that best matches one of the query images.

3.1.2 Interactive retrieval

For interactive retrieval, we enhanced the content-based approach by allowing the user to conduct multiple rounds of searching operations in which each successive round refines or builds on the results of a previous round. Each round consists of the following: (1) a similarity search in which target shots are scored against query content (using single frame or multi-frame search), and (2) a combining of these search results with the previous results list. This way, each successive round combines new results with a current list. We investigated several ways of combining results which involve different ways of manipulating the scores from the successive rounds. We have used a choice of the following aggregation functions for combining the scores:

$$D_i(n) = D_{i-1}(n) + D_q(n), \quad (3)$$

and

$$D_i(n) = \min(D_{i-1}(n), D_q(n)), \quad (4)$$

where $D_q(n)$ gives the score of video shot n for the present query, and $D_{i-1}(n)$ gives the combined score of video shot n for the previous query, and $D_i(n)$ gives the combined score result for the current round. Eq. 3 simply takes the sum of the score of each target video shot for the current

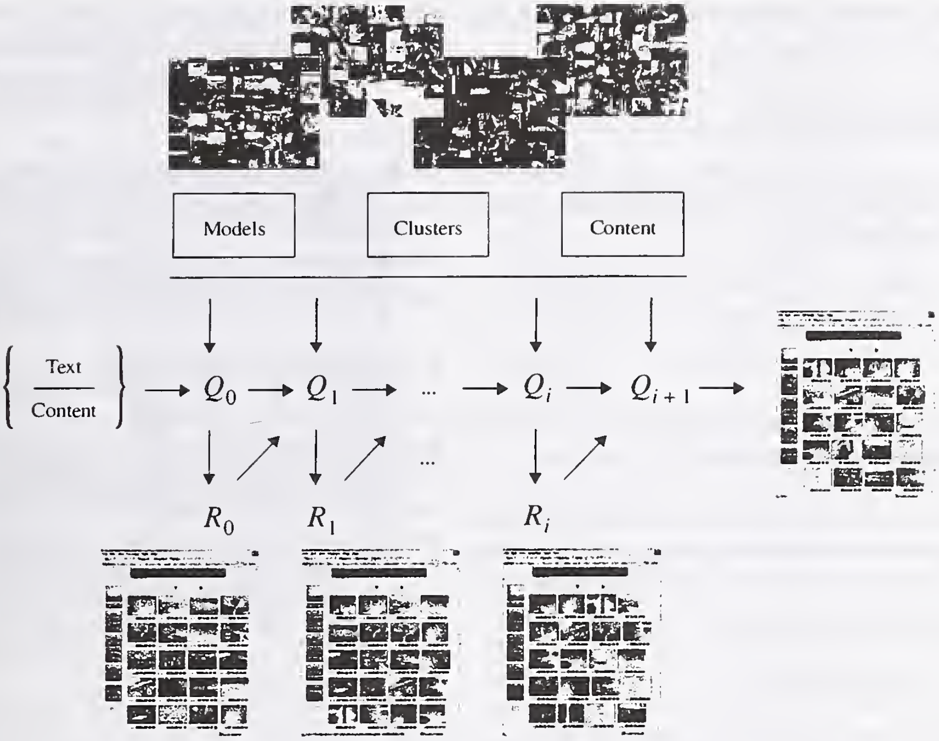


Figure 6: The video content retrieval engine integrates methods for searching in an iterative process in which the user successively applies content-based and model-based searches.

query plus the cumulative score of the previous queries. This has the effect of weighting the most recent query equally with the previous queries. Eq. 4 takes the minimum of the current score and the previous scores for each target video shot. This has the effect of ranking most highly the target shots that best match any one of the query images. Although, Eq. 3 and Eq. 4 are simple monotonic functions, other combining functions that use arbitrary join predicates are possible [NCS$^+$01].

For combining content-based and model-based retrieval, we allow the above methods for combining results, however, we allow additionally a filtering method that computes the intersection of the previous result list with the results from the current query, as described next.

3.2 Model-based retrieval

The model-based retrieval allows the user to retrieve the target shots based on the semantic labels produced by the models. Each semantic label has an associated confidence score. The user can retrieve results for a model by issuing a query for a particular semantic label. The target video shots are then ranked by confidence score (higher score gives lower rank). Since the models do not assign labels to all of the target shots, only the ones that are positively classified tothe semantic class, the model-based search does not give a total ranking of the target shots. That is, the model-based search both filters and ranks the target shots, which has implications for its use in iterative searching.

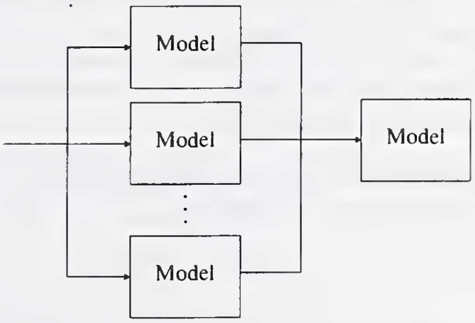


Figure 7: Parallel model search allows the user to define weighting of multiple models.

The models can be applied sequentially or in parallel as shown in Figure 7. In the case of parallel search, the user defines weighting of multiple models in a single query. In sequential search, the user decides based on interim results which models to apply. For example, a parallel model-based search is as follows: nature = 0.5∗outdoors+0.25∗water+

$0.25 * \text{sky}$. An example sequential model-based search is as follows: outdoors → no faces → no water.

3.3 Video query pipeline

The integrated search is carried out by the user successively applying the content-based and model-based search methods as shown in Figure 8.

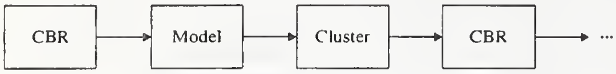


Figure 8: Integration of content-based and model-based searching in the video query pipeline.

For example, a user looking for video shots showing a beach scene can issue the following sequence of queries in the case that beach scenes have not been explicitly labeled::

1. Search for model = "outdoors",
2. Aggregate with model = "sky",
3. Aggregate with query image (possibly selected image) resembling desired video shot,
4. Aggregate with model = "water",
5. Aggregate with selected relevant image, video shot,
6. Repeat.

The iterative searching allows the users to apply sequentially the content-based and model-based searches. Different options can be used for scoring the results at each stage of the query and combining with the previous results. For TREC video retrieval, a choice of the following different approaches using different aggregation functions were provided for combining the scores:

1. **Inclusive**: each successive search operation issues new query against target database:
$$D_0(n) = D_q(n), \tag{5}$$
2. **Iterative**: each successive search operation issues query against current results list and scores by new query:
$$D_i(n) = D_q(n), \tag{6}$$
3. **Aggregative**: each successive search operation issues query against current results list and aggregates scores from current results and new query results:
$$D_i(n) = f(D_{i-1}(n), D_q(n)), \tag{7}$$

where $f(.)$ corresponds to min, max, or avg. The distance scores $D_i(n)$ are based on feature similarity (for CBR) and label confidence (for models). For the models, $D_q(n) = 1 - C_q(n)$, where $C_q(n)$ gives the confidence of the query label for video shot n, and $D_{i-1}(n)$, and $D_i(n)$ are defined as above. The lossy filtering is accounted for in that some target shots n^* have confidence score $C_q(n^*) = -\infty$. Eq. 7 combines the label score of each target video shot for the current query plus the cumulative label score of the previous queries, whereas Eq. 6 takes only the latest score.

3.4 Speech retrieval

To compute the video retrieval results using speech indexing for the TREC video retrieval, we used the textual statement of information need associated with each topic without any refinement or pruning of the text. The speech retrieval system works as follows: the system first loads the inverted index and precomputed weights of each of the non-stop words. A single pass approach is used to compute a relevancy score with which each document is ranked against a query, where the relevancy score is given by the Okapi formula [RWSJ+95].

Each word in the query string is tokenized, tagged, morphed and then scored using the Okapi formula above. The total relevancy score for the query string is the combined score of each of the query words. The scoring function takes into account the number of times each query term occurs in the document normalized with respect to the length of the document. This normalization removes bias that generally favor longer documents since longer documents are more likely to have more instances of any given word.

4 Retrieval system

We have applied this type of iterative and integrated content-based and model-based searching procedure for computing the results for many of the TREC video retrieval topics. Example topics for which this approach was used include: "scenes with sailing boats on a beach,", "scenes with views of canyons," and "scenes showing astronaut driving a lunar rover." The video retrieval system is illustrated in Figure 9.

4.1 Benchmark

The TREC video retrieval benchmark[1] was developed by NIST[2] to promote progress in content-based retrieval (CBR) from digital video via open, metrics-based evaluation. The benchmark involves the following tasks:

- Shot boundary detection

[1] http://www-nlpir.nist.gov/projects/t01v/revised.html
[2] http://trec.nist.gov/call01.html

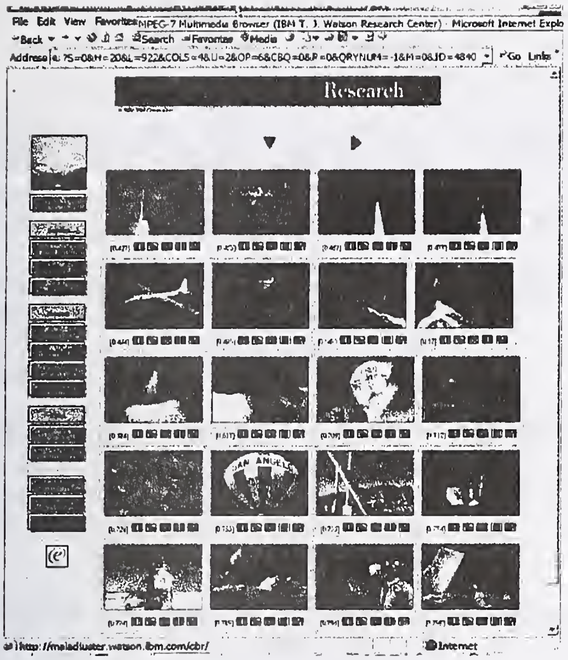

File Edit View Favorites
MPEG-7 Multimedia Browser (IBM T. J. Watson Research Center) · Microsoft Internet Explo
Back
Search
Favorites
Media
Address
75=0&H=20&L=922&COLS=4&U=2&OP=6&CBQ=0&P=0&QRYNUM=-1&M=0&JD=4840
Go
Links
Research
http://mediadluster.watson.ibm.com/cbr/
Internet

Figure 9: Screen image of the video retrieval system.

- Known item search
- General statements of information need.

The benchmark consists of the following:

- Approximately 11 hours of video
- 74 query topics, which include statements of information needs in text and example content
- Ground truth assessments (provided by participants for known-item queries)
- Quantitative metrics for evaluating retrieval effectiveness (i.e., precision *vs.* recall).

The benchmark focuses on content-based searching in that the use of speech recognition and transcripts is not emphasized. However, the queries themselves typically involve information at the semantic-level, i.e., "retrieve video clips of Ronald Reagan speaking," and opposed to "retrieve video clips that have this color." The two kinds of queries, known-item and general information need, are distinguished in that the number of matches for the known-item queries is pre-determined, i.e., it is known that there are only two clips showing Ronald Reagan. On the other hand, for the general searches, the number of matches in the corpus in not known, i.e., "video clips showing nature scenes."

4.2 Shot detection benchmark results

The results of the shot boundary detection on the TREC video corpus is shown in Table 1. The system performed extremely well for shot detection giving very high precision and recall.

	Ins. Rate	Del. Rate	Precision	Recall
Cuts	0.039	0.020	0.961	0.980
Gradual	0.589	0.284	0.626	0.715
All	0.223	0.106	0.831	0.893

Table 1: Shot boundary detection results for TREC video shot detection.

The results in Table 1 shows that the results for gradual changes could be improved. We found that in many of the cases, which were reported as errors, there was a detection of a boundary but the reported duration was too short. In such a case, the ISIS-based evaluation algorithm [ISI99] rejects the match, and considers it as both a deletion error and an insertion error. This is an undesired property of the evaluation criteria. If, for example, the system would not find a boundary at all, the evaluation would conider it as just a deletion, and rank the system better. In some other cases, a cut was reported as a short dissolve, with similar consequences.

Shot detection errors also resulted from the high noise level in the compressed MPEG video. For example, a periodic noisy pattern can be observed in Figure 3 at a period of 15 frames (one GOP) due to the color coding errors introduced by the MPEG encoding scheme. From our experience this noise level seemed somewhat high, but we have not quantified it.

4.3 Retrieval benchmark results

The results of the first retrieval experiment are shown in Table 2, which evaluates the average number of hits over the 46 "general search" queries. The interactive content-based retrieval (CBR) method is compared an automatic speech recognition (ASR) approach in which ASR was applied to the audio, and text indexing was used for answering the queries. The results show a signficant increase in retrieval quality using the interactive CBR approach.

Approach	Hits/query
Automatic speech recognition (ASR)	1.9
Interactive Content-based retrieval (CBR)	4.3

Table 2: Video retrieval results (avg. hits/query over 46 general searches).

Specific examples comparing retrieval performance for interactive CBR and ASR approaches are given in Table 3.

In some cases, such as topics VT66 and VT47, the ASR approach gave better retrieval results. In these topics, the relevant information was not easily captured by the visual scenes. However, for other topics, such as VT55, VT49, VT43, and VT42, the interactive CBR approach gave better performance than the ASR approach.

Topic#	Description	ASR	CBR
VT66	Clips about water project	9	3
VT47	Clips that deal with floods	8	1
VT55	Pictures of Hoover Dam	3	8
VT49	Lecture showing graphic	4	20
VT43	Shots showing grasslands	0	8
VT42	Shots of specific person	1	9

Table 3: Video retrieval results (hits/query) comparing interactive CBR and ASR methods for specific queries.

We also compared the interactive CBR approach to non-interactive (or automatic) CBR in which only a single iteration of searching was allowed. The results for two of the topics given in Table 4 show a significant increase in retrieval performance using the interactive CBR approach.

Topic \#	Description	Automatic CBR	Interactive CBR
VT54	Glen Canyon Dam	3	12
VT15	Shots of corn fields	1	5

Table 4: Video retrieval results (hits/query) comparing automatic and interactive CBR methods for specific queries.

5 Summary

In this paper, we described a system for automatic and interactive content-based retrieval that integrates features, models, and semantics. The system extracts feature descriptors from shots, which allows content-based retrieval, and classifies the shots using models for different events, scenes, and objects. The retrieval system allows the integration of content-based and model-based retrieval in an iterative search process. We developed also an approach based on speech indexing to provide a comparison with the content-based/model-based approach. We described the results of applying these methods to the TREC video retrieval benchmark.

References

[Cue] *IBM CueVideo Toolkit Version 2.1, http://www.almaden.ibm.com/cs/cuevideo/.* Download at http://www.ibm.com/alphaworks.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley Eastern, New York, 1973.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Proceedings of the Royal Statistical Society*, B(39):1–38, 1977.

[ISI99] In *European Workshop on Content Based Multimedia Indexing*, Toulouse, FR, October 1999. http://www-asim.lip6.fr/AIM/corpus/aim1/indexE.html.

[NCS+01] A. Natsev, Y.-C. Chang, J. R. Smith, C.-S. Li, and J. S. Vitter. Supporting incremental join queries on ranked inputs. In *Proc. Conf. on Very Large Databases (VLDB)*, Rome, Italy, September 2001.

[NKHR00] M. Naphade, I. Kozintsev, T. S. Huang, and K. Ramchandran. A factor graph framework for semantic indexing and retrieval in video. In *Proc. IEEE Workshop on Content-based Access to Image and Video Libraries (CBAIVL)*, Hilton Head, SC, June 12 2000.

[NLS+02] M. R. Naphade, C. Y. Lin, J. R. Smith, B. L. Tseng, and S. Basu. Learning to annotate video databases. In *IS&T/SPIE Symposium on Electronic Imaging: Science and Technology - Storage & Retrieval for Image and Video Databases 2002*, volume 4676, San Jose, CA, January 2002.

[Poo99] H. V. Poor. *An Introduction to Signal Detection and Estimation,.* Springer-Verlag, New York, 2 edition, 1999.

[PS01] D. Ponceleon and S. Srinivasan. Structure and content-based segmentation of speech transcripts. In *Proc. ACM Inter. Conf. Res. and Develop. in Inform. Retrieval (SIGIR)*, September 2001.

[RWSJ+95] 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.

[SBL+01] J. R. Smith, S. Basu, C.-Y. Lin, M. Naphade, and B. Tseng. Integrating features, models, and semantics for content-based retrieval. In *Proc. Multimedia Content-based Indexing and Retrieval (MMCBIR) workshop*, Rocquencourt, FR, September 2001.

[Smi01] J. R. Smith. MPEG-7 standard for multimedia databases. In *ACM Proc. Int. Conf. Manag. Data (SIGMOD)*, Santa Barbara, CA, May 2001. Tutorial.

[SS01] P. Salembier and J. R. Smith. MPEG-7 multimedia description schemes. *IEEE Trans. Circuits Syst. for Video Technol.*, August 2001.

Use of WordNet Hypernyms for Answering What-Is Questions

John Prager, Jennifer Chu-Carroll
IBM T.J. Watson Research Center
Yorktown Heights, N.Y. 10598
jprager/jencc@us.ibm.com

Krzysztof Czuba[1]
Carnegie-Mellon University
Pittsburgh, PA 15213
kczuba@cs.cmu.edu

Abstract

We present a preliminary analysis of the use of WordNet hypernyms for answering "What-is" questions. We analyse the approximately 130 definitional questions in the TREC10 corpus with respect to our technique of Virtual Annotation (VA), which has previously been shown to be effective on the TREC9 definitional question set and other questions. We discover that VA is effective on a subset of the TREC10 definitional questions, but that some of these questions seem to need a user model to generate correct answers, or at least answers that agree with the NIST judges. Furthermore, there remains a large enough subset of definitional questions that cannot benefit at all from the WordNet isa-hierarchy, prompting the need to investigate alternative external resources.

1. Introduction

Work in the field of Question-Answering has taken off since the introduction of a QA track in TREC in 1999 (see, e.g. [Voorhees and Tice, 2000]). Much of the published work in the field has centered around the fact-based questions that form the current basis of this track. While differing greatly in the specifics, most of the systems published in the literature to date use a similar approach (at the coarsest level of description) of a sequence of processing stages: the question is analysed to discover the answer-type that is sought, a query is constructed from the question (with appropriate vocabulary expansions and morphological normalization), a standard IR search is performed, documents or passages are retrieved and these texts are examined for presence of terms of the appropriate answer type, possibly in a context that satisfies other derived criteria (see for example [Clarke et al. 2001, Ittycheriah et al. 2001, Moldovan et al. 2000, Prager et al. 2000, Srihari and Li 2000]). Some systems, such as Falcon [Harabagiu et al., 2001] and some of our own experimental prototypes, are using feedback loops to retry stages that are deemed unsuccessful.

One recurring question type is the definitional question, usually of the form "What is/are <noun phrase>", although other syntactic forms are used but with essentially the same meaning. The difficulty that arises with these questions is that the answer type is left completely open. Even the Webclopedia system [Hovy et al., 2001], which employs an extensive question typology, cannot be very specific with these questions. The TREC9 question set consisted of about 5% definitional questions, while the TREC10 set, which appears to better mirror actual user questions (Ellen Voorhees, personal communication), consisted of about 26% definitional. Thus we believe that examining what is required to answer this kind of question is worthwhile.

Granted, there are many occasions where the text explicitly provides a definition with sentences of the form "X is <something>" – in fact by a cursory analysis of the judgment sets some 82% of the TREC10 definitional questions are answered by copular expressions. However, relying on this is easily seen to be problematic. Firstly, definitions are provided using overall a wide variety of syntactic structures, but more importantly, very sophisticated NLP is required to determine that the <something> above is a definition rather than some arbitrary predicate. Clearly some additional component is required. WordNet [Miller, 1995] is currently the preferred resource for ontological information, and promises to be very helpful for this particular problem. We have previously shown [Prager et al. 2001] its effectiveness for a small class of "What-is" questions; in this paper we examine the effectiveness of the WordNet

[1] Work performed while at the IBM T.J. Watson Research Center.

hypernym (or "isa") hierarchy for the TREC10 "What-is" questions.

2. Predictive/Virtual Annotation for Question-Answering

Our Question-Answering system employs the technique of Predictive Annotation, introduced and described in [Prager et al. 2000a]. The technique revolves around the concept of semantic class labels that we call QA-Tokens, corresponding loosely to some of the Basic Categories of [Rosch et al. 1976]. These are used not only as Named Entity descriptors, but are actual tokens processed by the indexer. For example, people are tagged with PERSON$, lengths of time with DURATION$. For named entity detection we use Textract [Wacholder, Ravin and Choi, 1997, Byrd and Ravin, 1999], and for search we use Guru-QA, based on Guru [Brown and Chong, 1997], but with a specialized weighting scheme and ranking algorithm.

Identifying the semantic answer-type (QA-Token set) in the question (e.g. "Who" -> PERSON$ and "How long" -> DURATION$ or LENGTH$) and matching against a semantically tagged corpus only works if such information is conveyed by the question either explicitly or implicitly. Questions beginning with "Who", "When" and "Where" fulfill this requirement, as do those with "How + <adjective>" or "How + <adverb>", and also "What (or Which) + <noun phrase>". However, definitional "What-is" questions (e.g., "What is a nematode?") do not indicate the answer type, thus rendering the annotations in the corpus ineffective.

For such questions we need an alternative approach. One possibility is to find all occurrences of the question term in the corpus, and to analyze all these documents (or at least the passages surrounding the instances) for key terms or phrases indicating a definition, as did Hearst [1998], and Joho and Sanderson [2000]. However, we have adopted another approach, more in line with our disposition to shift the computational burden in the direction of IR rather than NLP. As described in [Prager et al. 2001], this approach has been shown to give an accuracy of 83% on TREC9 "What-is" questions. This sample set was rather small (24 questions) and was thus not a reliable indicator of its general efficacy.

Our approach stems from the observations that (I) providing the parent class should be a good answer to a definitional "What-is" question, and (2) frequently terms are encountered in text along with their class (e.g. "nematodes and other worms", "metals such as tungsten", "gecko (a lizard)", and so on). WordNet is a good, easily-accessible ontological resource for finding the isa-hierarchy of a term, and so we use WordNet to find the best class descriptor(s) for the question term and include them as additional search terms.

Our WordNet lookup algorithm works by counting co-occurrences of the question term with each of its WordNet ancestors in the TREC corpus, and dividing this number by the number of isa-links between the two. The best terms, by this calculation, win. This approach guarantees that the selected terms co-occur[2] with the question term, and therefore that answer passages can be found.[3]

Since our search process ([Prager et al. 2000]) is passage-based, we look for short passages that contain both the question term and any of its ancestors that our WordNet lookup algorithm proposes. According to criteria such as described in [Radev at al 2000, Chu-Carroll et al. in progress], the best answer fragments are returned.

3. Performance Evaluation and Data Analysis

While our algorithm was shown to be very effective on TREC9 "What-is" questions, it was much less so on TREC10. Hence we decided to examine the assumptions inherent in the process in order to understand more fully the conditions under which our algorithm is effective.

The assumptions underlying our approach were as follows:

1. The question term is in WordNet.
2. At least one of its ancestors is useful as a definition.
3. Such ancestors (in #2) are themselves sufficient as definitions.
4. Our algorithm can find the ancestor(s).

We need to explain the distinction between conditions #2 and #3. We have found that there are some cases where an ancestor provides a definition that would best be extended by further qualification on the ancestral term, e.g., by citing the difference between the term and others in its ancestral class.[4] For example, saying that

[2] within a two-sentence passage.

[3] In a small number of cases, the question term is present in WordNet but none of the ancestors co-occur with it anywhere in the TREC corpus.

[4] We realize that it is a subjective decision as to whether or not a term makes for an acceptable definition. We have made

an amphibian (TREC10 #944) is an animal is technically correct, but it is considerably more useful to say that it is an animal that lives both on land and in water. (It can also be a vehicle, but the same analysis holds.) Thus, just calling an amphibian an animal violates assumption #3. However, we maintain that, even though "animal" by itself does not provide for a sufficiently useful answer for "amphibian", including the search term "animal" will likely lead us to passages in which good definitions for amphibian can be found. On the other hand, we have found that there are terms for which none of the ancestors are particularly useful, even as partial definitions. For example, the parentage of "eclipse" (TREC10 #1016) in WordNet is the synset-chain: {interruption, break, abrupt change}, {happening, occurrence, natural event}, {event}, while a good definition would talk about one astronomical body blocking or obscuring another. In other words, one cannot easily make a simple definition by adding pre-modifiers or prepositional phrases to the ancestral noun. For questions like this one, assumption #2 is violated.

For the purpose of analyzing the effectiveness of our algorithm, we identified 130 TREC10 questions which sought the definition of a given term or phrase. Although most of these questions are phrased in the "What is/are X?" format, we included those questions that were similar in nature, such as "What does X mean?" and "What does X do?" Since WordNet includes a small number of famous people, we also process "Who is/was X" questions in the same way.

Granting that there is occasional subjectivity involved, we have grouped the 130 definitional questions into 5 groups according to which of assumptions 1-4 have been violated. More specifically, questions are classified based on the following criteria:

?? Group 1: question term is not in WordNet.

?? Group 2: no hypernym is particularly useful as part of a definition.

?? Group 3: "best" ancestor is useful as a partial definition, but needs to be further qualified.

?? Group 4: "best" ancestor is sufficient as a definition for the question term by itself. But our WordNet lookup algorithm failed to return it as the best candidate.

?? Group 5: "best" ancestor is a good definition by itself and our algorithm found it.

Table 1 shows a summary of relevant statistics for the 5 groups,[5] while Table 2 - Table 6 contain detailed information about each group used to generate the summary. MRR is Mean Reciprocal Rank of the first correct answer and is in the range 0-1.

Group	Count	MRR
1	25	0.171
2	19	0.097
3	40	0.283
4	14	0.232
5	32	0.812

Table 1 Summary of Question Classification

Table 2 - Table 6 consist of the following columns: the TREC10 question number, the question term, what our algorithm finds as a suitable ancestor (possibly a disjunction), and the score our system receives (given as rank of first correct answer). Note that this score "r" is based not on our run as submitted to NIST, but after fixing a bug that was later found; where the fixed system differed from the original, evaluation was done by reference to the NIST-supplied judgment sets. Question terms in italics are those for which NIST asserts there was no answer in the TREC corpus.

Trec#	Question Term	WordNet Ancestor(s)	r
915	biosphere		0
947	fibromyalgia		0
961	spider veins		0
997	Duke Ellington		1
1022	Wimbledon		5
1026	target heart rate		0
1034	severance pay		0
1042	Phi Beta Kappa		0
1051	nanotechnology		5
1075	neuropathy		5
1077	cryptography		0
1114	ozone depletion		0
1116	Sitting Shiva		0
1141	*home equity*		0
1148	pilates		0
1160	dianetics		0
1180	pulmonary fibrosis		0
1185	foot and mouth disease		0
1262	Moulin Rouge		2
1267	mad cow disease		0

every attempt to base our analysis on the TREC10 judgment set whenever possible.

[5] The MRR for each group is calculated by disregarding those questions known to have NIL as answers.

1289	die-casting		0
1324	bangers and mash		0
1330	spirometer test		1
1385	bio-diversity		0
1393	e-coli		1

Table 2 Group 1: Question Term Not in WordNet

Table 2 enumerates those 25 questions in which the question term is not present in WordNet. For these questions, our system does not benefit from the virtual annotation mechanism, and, as a result, found answers to only 7 of them within the 50-byte answer fragments using our default mechanism.

Trec#	Question Term	WordNet Ancestor(s)	r
897	atom	{particle, matter, molecule}	0
903	autism	{syndrome}	2
974	prism	{form, optical prism}	0
985	desktop publishing	{business}	0
992	coral reefs	{formation}	0
1016	eclipse	{break}	0
1033	platelets	{blood platelet}	1
1046	sunspots	{point}	0
1054	*obtuse angle*	*	0
1088	relative humidity	*	0
1121	spleen	{ire, anger, tissue}	0
1135	Valentine's Day	*	0
1170	viscosity	{property}	0
1179	antacids	{cause}	4
1243	acid rain	{acid precipitation}	0
1255	ciao	{message}	0
1273	annuity	{payment}	0
1303	metabolism	{activity}	0
1363	compounded interest	{cost, charge}	0

Table 3 Group 2: No Hypernym Forms Useful Part of Definition

In Table 3, we show 19 questions for which none of the ancestors of the question term in WordNet are particularly useful even as partial definitions. The third column in the table shows what our WordNet lookup algorithm proposes as the "best" ancestor,[6] although to classify a question into this category, we have manually examined the other ancestors to ensure that our algorithm did not overlook other suitable candidates.

Table 2 and Table 3 contain a total of 44 questions, or about 1/3 of all definitional questions, for which WordNet's utility in aiding question answering is minimal at best. This fact is further confirmed by the statistics shown in Table 1, where the MRR scores for groups 1 and 2 are substantially lower than those for the other groups. This prompts the need to investigate other supplemental sources of information for when the WordNet isa-hierarchy fails. In addition, although it is obvious when additional information is needed for those questions in Table 2, it is not a trivial task for a system to determine when an ancestor proposed by WordNet is unlikely to be found in the definition of a question term and should therefore be discarded. We leave the investigation of both of these issues as future work.

Trec#	Question Term	WordNet Ancestor(s)	r
918	cholesterol	{alcohol}	0
920	caffeine	{compound}	0
926	invertebrates	{animal}	0
935	Teflon	{plastic}	2
944	amphibian	{vehicle, amphibious vehicle, animal, aircraft}	0
969	pH scale	{measure}	1
982	xerophytes	{plant, planting}	0
991	cryogenics	{science, field}	0
994	neurology	{study, medicine}	0
1005	acupuncture	{treatment}	1
1028	*foreclosure*	{proceeding, proceed}	0
1043	nicotine	{substance}	4
1055	polymers	{compound}	0
1067	supernova	{star}	1
1102	defibrillator	{device}	0
1108	fungus	{plant, planting}	0
1129	sonar	{device}	2
1131	phosphorus	{element}	1
1138	bandwidth	{measure}	0
1140	parasite	{organism, leech, sponge}	1
1142	meteorologist	{expert, specialist}	0
1152	Mardi Gras	{carnival, day}	3
1166	osteoporosis	{health problem}	1
1169	esophagus	{passage}	0
1192	barometer	{instrument}	0
1196	solar wind	{radiation}	0
1209	fuel cell	{device}	1
1214	diabetes	{disorder}	4

[6] In those questions marked by an asterisk, our algorithm did not return any candidate term because none of the question term's WordNet ancestors co-occur with it in the TREC corpus.

1258	acetic acid	{compound}	5
1266	pathogens	{microorganism}	1
1285	carcinogen	{substance}	3
1288	nepotism	{favoritism}	3
1300	carbon dioxide	{compound, CO}	3
1309	semiconductors	{semiconductor device, material}	0
1310	nuclear power	{energy}	0
1322	enzymes	{protein}	0
1362	solar cells	{photovoltaic cell}	0
1365	antigen	{drug}	0
1370	thermometer	{instrument}	0
1384	pectin	{sugar}	0

Table 4 Group 3: "Best" Hypernym Not Specific Enough as Definition

Table 4 shows 40 questions where WordNet proposes an ancestor which requires further qualification (either in the form of a premodifier or a prepositional phrase postmodifier) in order to constitute a useful definition. For example, “cholesterol” can be defined as a “fatty alcohol” and “invertebrates” as “animals without backbones.” Column three in the table again shows the ancestor returned by our WordNet lookup algorithm, which is included as part of at least one NIST-judged correct answer in each case.

Note that the Virtual Annotation algorithm we originally described in [Prager et al. 2001] looked strictly at ancestor terms in the isa-hierarchy. Synonyms were only examined when explicitly called for by questions of the form “What is another name for X”. However, following the observation that sometimes in “What is X” questions the “X” is a rare synonym for a better-known term, in this experiment we treated the question-term’s synset as a level-0 parent. This backfired when it initially found “oesophagus” as the meaning of “esophagus”, for example, and “grippe” for “influenza”, but we found that in general it was more helpful to include the synset of the question term in the analysis. Testing for orthographic or other such variations helped eliminate the former kind of problem, and filtering on occurrence count ratios the latter.

Trec#	Question Term	WordNet Ancestor(s) found/could have been found	r
912	epilepsy	{disorder}/ {neurological disorder}	1
917	bipolar disorder	{condition}/ {manic depression}	0
1081	leukemia	{cancer}/ {cancer of the blood}	1
1113	influenza	{disease}/ {contagious disease}	4
1159	fortnight	{period}/ {two weeks}	0
1183	strep throat	{disease}/ {sore throat}	0
1188	Aborigines	{ }/ {(original) inhabitant}	0
1207	pneumonia	{disease}/ {respiratory disease}	0
1224	mold	{plant}/ {fungus}	2
1248	quicksilver	{substance, matter}/ {mercury}	0
1280	Muscular Dystrophy	{disease}/ {genetic disorder, genetic disease}	0
1317	genocide	{kill, killing}/ {racial extermination}	0
1377	rheumatoid arthritis	{disease}/ {inflammatory disease}	0
1379	cerebral palsy	{disorder}/ {nervous disorder}	2

Table 5 Group 4: "Best" Ancestor Makes Good Definition But Was Not Found

Table 5 illustrates 14 examples in which there exists a better WordNet ancestor than the one proposed by our lookup algorithm. The third column in the table shows two sets of terms, the first of which is the term selected by our algorithm and the second of which is a term also present in the WordNet hierarchy that we prefer over the selected term as a definition of the question term. In all cases, the selected term is a hypernym of the preferred term, which has a very low or zero co-occurrence count with the question term. In addition, note that in many cases, the selected term consists of the head noun of the preferred term, which includes an additional adjectival premodifier or a prepositional phrase postmodifier.

As discussed earlier, our question answering system includes the proposed WordNet hypernym as an additional search term for passage retrieval. For questions in groups 3 and 4, this means that the search is biased toward passages that include terms that could potentially form a definition for the question term. The effect of the inclusion of such terms is evidenced by the statistics in Table 1, where the MRRs for groups 3 and 4 are higher than for groups 1 and 2, which received no help for WordNet at all. However, the improvement in MRR scores is less than we would have liked. We plan to investigate more sophisticated answer-selection mechanisms for identifying contexts in which definitions are provided.

Trec#	Question Term	WordNet Ancestor(s)	r
896	Galileo	{astronomer}	1
936	amitriptyline	{antidepressant}	1
937	shaman	{priest}	1
959	Abraham Lincoln	{(frontier) lawyer}	1
980	amoxicillin	{antibiotic}	1
999	micron	{micrometer}	0
1038	poliomyelitis	{infantile paralysis}	1
1044	vitamin B1	{thiamine}	1
1058	Northern Lights	{aurora borealis}	0
1061	acetaminophen	{painkiller}	1
1110	sodium chloride	{salt}	1
1126	phenylalanine	{amino acid}	1
1137	hypertension	{high blood pressure}	1
1168	peyote	{mescaline mescalin mescal}	2
1177	chunnel	{Chunnel Tunnel}	1
1181	Qaaludes	{methaqualone}	2
1182	naproxen	{drug, anti-inflammatory}	2
1223	Milky Way	{galaxy}	1
1230	semolina	{flour}	1
1232	Ursa Major	{constellation}	1
1254	thyroid	*{thyroid gland}*	0
1271	ethics	{study, morality}	2
1282	propylene glycol	{antifreeze}	1
1283	panic disorder	{anxiety disorder}	1
1290	myopia	{nearsightedness}	1
1311	tsunami	{wave, tidal wave}	1
1320	earthquake	*{temblor}*	0
1328	ulcer	{ulceration}	1
1329	vertigo	{dizziness}	1
1352	schizophrenia	{mental illness}	1
1360	pediatricians	{baby doctor}	1
1364	capers	{pickle}	1

Table 6 Group 5: Best Found Hypernym Makes Good Definition

The final group of definitional questions, shown in Table 6, contains those where the ancestor proposed by our algorithm constitutes a useful definition of the question term by itself.[7] Not surprisingly, for this group of questions, our system returned the correct definition in the first position in the vast majority of cases, and as a result received a very high MRR score, as shown in Table 1.

4. Discussion

The issue of what constitutes a correct answer has raged in the TREC community since the first QA track in TREC8, and shows no sign of being settled. One particularly important but neglected issue is that of knowing who the questioner is. In everyday communication, people ask questions of each other, and in all cases the answers given are conditioned on the responder's knowledge of the questioner and suspicions of what they know, what they don't know and how much they are seeking to learn.

For argument's sake, one can postulate several different kinds of questioner. These might include: a child, an intelligent adult for whom English (or in the case of TREC10 #1255 "What does ciao mean", Italian) is or is not their primary language, or a student learning a new field (so he might well know other technical terms in the field). NIST has not asserted any user model. Unfortunately, it is not that easy to induce one from the judgment sets made available. It is particularly difficult to infer what level of specificity is required in an answer. For example, carbon dioxide is not a compound (according to NIST) yet nanotechnology is a science; diabetes is not a disorder yet acupuncture is a treatment, influenza is not a disease but poliomyelitis is.

Given a user description, it should be straightforward to determine the correct answer level; at least it should give rise to less haphazard specificity levels of correct answer. For instance, consider TREC10 #1266: "What are xerophytes". For all but botanists or landscape gardeners, the answer "plants" is probably sufficient, absent any context.

One approach that might be worth taking is to generate alternative answers based on the different user-model assumptions, and to assume a priori probabilities of these different models. These probabilities can be fixed, or (outside of TREC) determined by exterior processes. Within the TREC paradigm, however, one can possibly infer something about the questioner from the question itself. An average intelligent adult could very reasonably ask "What are xerophytes?", but maybe not so reasonably ask "What is the Milky Way?". Even the article in the question can convey meaning: "What is a thyroid?" might well be asked by a child, but "What is the thyroid?" might be asked by an anatomy professor of a medical student (i.e. the definite article here can convey tacit agreement of the domain, in this case the

[7] Here the effect of subject judgment comes into play. Those WordNet ancestors in italics were not considered correct answers by the NIST judges.

human body, which might be all that is needed to answer a child).

The difficulty with paying close attention to the question syntax is that if indeed the question is asked by a child or a non-native speaker, then conclusions based on correct grammaticality may be unreliable. "What is mold?" (TREC10 #1224) requires a very different answer from "What is a mold?", but only if presence or absence of the indefinite article can be trusted; if we knew the question came with a Russian accent, for example, we would have more information to work with! Answering the question properly requires identifying an appropriate user model. Doing this requires, in part, analysis of the question syntax. Drawing valid conclusions from the question syntax again requires a user model!

5. Conclusions and Future Work

We have broken down the 130 TREC10 definitional questions into five groups according to how useful an algorithm that seeks primarily to define a term by its WordNet class or genus can be. We have ideas about how to address each group, but the challenge is in identifying which situation is present for any particular question.

Our system had the worst performance with groups 1 and 2, when a term was not in WordNet or it had an entry but its WordNet parents were not useful for definitional purposes – in fact the latter case fared worse than the former because our system was distracted into thinking it had an answer. One possible solution is to manually explicitly identify these general hypernyms (property, cause, activity etc.) and to make our program try another approach if these are initially proposed.

The next-ranking groups (nos. 3 and 4) were those where the located WordNet ancestor was promising but not specific enough, and where our algorithm selected a non-optimal ancestor. The former problem can possibly be addressed by selecting (to add to the search) significant terms from the WordNet gloss in the hopes that they are differentiae of the genus. The latter problem can in individual cases be fixed by retuning the parameters in our lookup algorithm, but we don't want the successful cases (primarily in group 5) to start failing. It is unclear right now for how large a subset of groups 4 and 5 a successful parameter set can be found.

Group 5 fared very well, which gives us hope that WordNet will be useful in the future for a significant number of definitional questions – our groups 3-5 totalled two-thirds of the TREC10 set.

We have not had time to explore those cases in group 5 where we did not find the right answer, according to the NIST assessors, nor why the definitions in our group 3 were considered not specific enough. For many of these cases, arguments can be made that, depending on who asked the question, the right answer was found. A more complete analysis requires both a model of the user and of what constitutes a good answer to a question. We hope to pursue this line of inquiry in the near future.

References

[1] Brown, E.W. and Chong, H.A. "The Guru System in TREC-6." *Proceedings of TREC6*, Gaithersburg, MD, 1998.

[2] Byrd, R. and Ravin, Y. "Identifying and Extracting Relations in Text", *Proceedings of NLDB 99*, Klagenfurt, Austria, 1999.

[3] Chu-Carroll, J., Prager, J., Ravin, Y., Cesar, C. "A Hybrid Approach to Natural Language Web Queries", in preparation.

[4] Clarke, C.L.A., Cormack, G.V., Kisman, D.I.E., Lynam, T.R. Question Answering by Passage Selection. In *Proceedings of the 9th Text Retrieval Conference (TREC9)*, Gaithersburg, MD, to appear in 2001.

[5] Harabagiu, S., Moldovan, D., Pasca, M., Mihalcea, R., Surdeanu, M., Bunescu, R., Gîrju, R., Rus, V., Morarescu, P. FALCON: Boosting Knowledge for Answer Engines. In *Proceedings of the 9th Text Retrieval Conference (TREC9)*, Gaithersburg, MD, to appear in 2001.

[6] Hearst, M.A. "Automated Discovery of WordNet Relations" in *WordNet: an Electronic Lexical Database*, Christiane Fellbaum Ed, MIT Press, Cambridge MA, 1998.

[7] Hovy, H., Gerber, L., Hermjakob, U., Lin, Chin-Yew, Ravichandran, D. "Towards Semantic-Based Answer Pinpointing", *Proceedings of Human Language Technologies Conference*, pp. 339-345, San Diego CA, March 2001

[8] Miller, G. "WordNet: A Lexical Database for English", *Communications of the ACM* 38(11) pp 39-41, 1995

[9] Ittycheriah, A., Franz, M., Zhu, W-J., Ratnaparkhi, A., Mammone, R.J. IBM's Statistical Question Answering System, In *Proceedings of the 9th Text Retrieval Conference (TREC9)*, Gaithersburg, MD, to appear in 2001.

[10] Joho, H and Sanderson, M. "Retrieving Descriptive Phrases from Large amounts of Free Text", *Proceedings of CIKM, 2000.*

[11] Moldovan, D., Harabagiu, S., Pasca, M., Mihalcea, R., Girju, R., Goodrum, R. and Rus, V. The Structure and Performance of an Open-Domain Question Answering System. In *Proceedings of the Conference of the Association for Computational Linguistics* (ACL-2000), 563–570.

[12] Prager, J.M., Radev, D.R., Brown, E.W. and Coden, A.R. "The Use of Predictive Annotation for Question-Answering in TREC8", *Proceedings of TREC8*, Gaithersburg, MD, 2000.

[13] Prager, J.M., Brown, E.W., Coden, A.R. and Radev, D.R. "Question-Answering by Predictive Annotation", *Proceedings of SIGIR 2000*, pp. 184-191, Athens, Greece, 2000.

[14] Prager, J.M., Radev, D.R. and Czuba, K. "Answering What-Is Questions by Virtual Annotation." *Proceedings of Human Language Technologies Conference*, San Diego CA, pp. 26-30, March 2001

[15] Radev, D.R., Prager, J.M. and Samn, V. "Ranking Suspected Answers to Natural Language Questions using Predictive Annotation", *Proceedings of ANLP'00*, Seattle, WA, 2000.

[16] Rosch, E. et al. "Basic Objects in Natural Categories", *Cognitive Psychology* 8, 382-439, 1976.

[17] Srihari, R. and W. Li. 2000. A Question Answering System Supported by Information Extraction. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics* (NAACL-00), 166–172.

[18] Voorhees, E.M. and Tice, D.M. "Building a Question Answering Test Collection", *Proceedings of SIGIR 2000*, pp. 184-191, Athens, Greece, 2000.

[19] Wacholder, N., Ravin, Y. and Choi, M. "Disambiguation of Proper Names in Text", *Proceedings of ANLP'97*. Washington, DC, April 1997.

IBM's Statistical Question Answering System – TREC-10

Abraham Ittycheriah, Martin Franz, Salim Roukos
P.O.Box 218,
Yorktown Heights, NY 10598
{abei,franzm,roukos}@watson.ibm.com

Abstract

We describe herein the IBM Statistical Question Answering system for TREC-10 in detail. Based on experiences in TREC-9, we have adapted our system to deal with definition type questions and furthermore completed the trainability aspect of our question-answering system. The experiments performed in this evaluation confirmed our hypothesis that post-processing the IR engine results can achieve the same performance as incorporating query expansion terms into the retrieval engine.

1 Introduction

The TREC evaluations in question answering prompted many sites to develop technology to deal with open-domain question answering from real user queries. Our system focus is to create technology that can learn from question answer pairs sufficient rules and weights such that these can be used to find the answers to new questions. In TREC-9, we used a statistical algorithm for both answer tagging (predicting the class of the answer desired by a question), as well as named entity tagging (predicting the class of segments of text). Matching these predictions, as well as maximizing the overlap of question words to answer words yielded an answer to the question in our TREC-9 system. For TREC-10, we developed the following additional components:

- New and refined answer tag categories.
- Query expansion lists incorporated in answer selection.
- Focus expansion using *WordNet* (Miller, 1990).
- Dependency relationships using syntatic parsing.
- A maximum entropy formulation for answer selection (Ittycheriah, 2001).

These are described in the following sections and then we give some preliminary analysis of TREC-10 questions that were solved using these techniques as well as some crucial failures of our system. In the discussion below, we abuse the term TREC-9 meaning not the entire TREC-9 test but the first 500 questions which is the set of questions from the real TREC-9 test without the NIST reformulated questions.

2 Refining the Answer Tag Model

In our previous system, one of the predicted class is the unknown class, which we label as PHRASE. A comparison of the answer tags in three datasets is shown below in Fig. 1. Given the large number of questions being labelled as PHRASE, we set out for TREC-10 to increase the number of answer tags that we generated in the hopes of reducing the number of questions that were labelled PHRASE. Since both questions have to be annotated with the new answer tags as well as named entity material tagged similarly, it is a very expensive proposition to change the tag sets. We chose the following 31 tags as be-

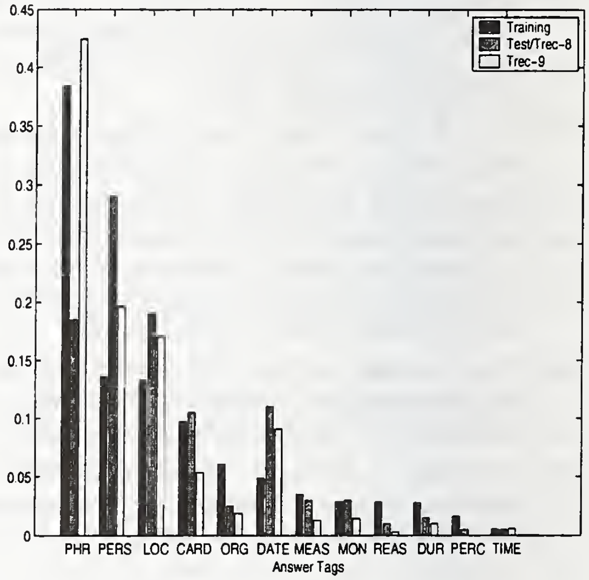


Figure 1: Histogram of Answer Tag Classes.

ing a reasonable expansion of the categories used in TREC-9. The 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

Despite the increased number of answer tag classes, the percentage of PHRASE labelled questions has increased in TREC-10 to be 56% (280 out of 500 questions). These questions are dominated by "what is" or "what are" (232 out of 500 questions) which are mostly definitional type questions. Thus, the TREC-10 test is very similar to the TREC-9 test in terms of answer tags and definitional questions.

3 Incorporating Query Expansion in Answer Selection

Our information retrieval subsystem uses the same two-pass approach as our TREC-9 system. In the first pass, we search an encyclopedia database. The highest scoring passages were then used to create expanded queries, applied in the second pass scoring of the TREC documents. The data pre-processing and relevance scoring techniques are similar to the ones applied in the previous TREC evaluations (Franz and Roukos, 1998), (Franz et al., 1999). The expanded queries are constructed using the local context analysis (LCA) technique (Xu and Croft, 1996).

Examining the words used in the query expansion for IR, it was noted that often the answer words occurred on that list. Quantifying this observation, we measured the number of words intersecting the answer patterns for the TREC-9 test and found that 185 questions out of 500 questions had at least one word of its answer as part of the query expansion list. However, in our normal setting for query expansion we expand each query by a set number of words. Typical queries in this domain are five words and we add twenty words through the LCA expansion. For TREC-9, out of 10K words on the query expansion list only 297 words are actually part of the answer strings. In the sentence scoring portion of our system, we add to the sentence score half of the IDF weight of the expanding word. The top sentences are reranked by a maximum entropy based answer selection model and within this model we incorporate knowledge about the query expansion as a binary feature indicating the presence or absence of such expanding words.

4 Focus Matching

Question focus was defined in (Moldovan and et. al., 1999), and here we modify and state it as the word or sequence of words which optionally occur following a question word, which serves to indicate the answer tag. This notion has been used in our answer tag modelling previously (Ittycheriah et al., 2001), but in addition we used this notion in TREC-10 to get a refined sense for the broad categories as well as provide a substitute for answer tags in the case of PHRASE type questions. If the question has a focus, then answers who have hypernym (parent) or hyponym (child) relationship in *WordNet* are boosted in score. For example, a question such as *What river in the US is known as Big Muddy?*, the focus is derived to be *river* and in *WordNet* we see that the Mississippi River has a child relationship with river. The distance of the focus word to the nearest content question word is measured as is done for the named entities also. The focus score for the sentence i is then computed as

$$S_{f,i} = (F + (3 - d_f) * D_f)$$

where F is the focus boost, d_f is the distance, and D_f the distance penalty for focus. At the best operating point, the focus boost is set to 10% of the total IDF weight of the question words and the distance penalty, D_f is set to 4.0. This score is added to the sentence score, which is described in (Ittycheriah et al., 2001).

5 Dependency Relationship Matching

Use of syntactic parsing has been used in information retrieval systems before, for example (Strzalkowski et al., 1997) shows an effective improvement in the precision rate using head-modifier pairs. Also, dependency structures on a syntactic parse is used in (Harabagiu and et. al., 2000) for deriving the logical form. The use of the dependency structure here will be to,

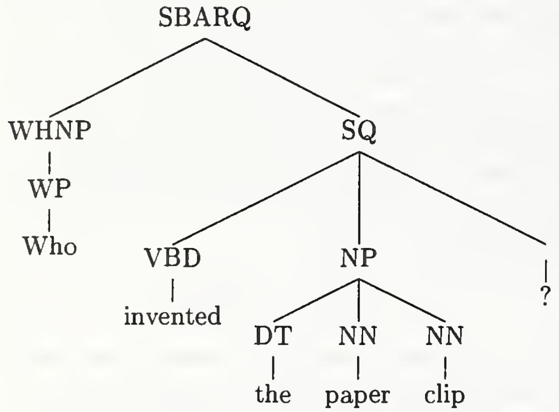
- constrain the match of branches
- to analyze what other extra words are in a dependent structure with the question words

- give credit to the proper named entity in a sentence

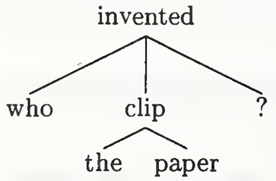
Using a parse structure gives the system the capability to score higher those answer candidates that have the words in a similar structure. Note though that the Cluster Words, Dispersion and Named Entity distance are performing similar functions though they are not explicit as using the parse structure. An example will motivate the use of the parse information.

5.1 Dependency Example

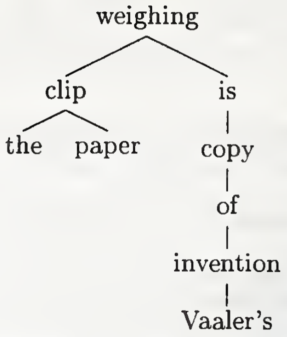
An example of a question in the TREC-9 corpus is, *Who invented the paper clip?* The parse for this question derived by the statistical parser is,



The dependency graph is,



The structure then reveals the requirements on the answer: specifically that the user is asking about a "paper clip" as opposed to "clips of paper". A strict *bag-of-words* technique assigns equal weight to both phrases. Given this dependency graph, it would be trivial now if the answer was essentially of the same form, but in this case the answer lies in the sentence *The paper clip, weighing a desk crushing 1,320 pounds, is a faithful copy of Norwegian Johan Vaaler's 1899 invention, said ...* . A portion of the dependency tree derived for this sentence is,



While the entire parse tree can not be aligned per se, we present a method of doing partial matches below. Also note, that in the ideal answer, "Johan Vaaler invented the paper clip", the named entity that is desired will have a dependency to a word in the question. This is not in general true but for questions dealing with the defined set of entities it holds.

Additionally, consider the answer *"Like the guy who invented the safety pin, or the guy who invented the paper clip", David says.* In this answer to the above question, the segment "invented the paper clip" gets full credit. However, "David" gets no credit, since the dependency structure shows no relationship to the invention of the paper clip. Without a dependency structure, it is difficult from the surface text to determine which invented should get credit and whether a named entity match should count or not.

6 Trainable Answer Selection

In question-answering, a classification viewpoint to the problem would be to find answers, a, that maximize the conditional probability $p(a|q)$. Formulation of this model would allow us to search the space of answers and find the answer to the given question. However, this model is not tractable currently as the answer space is large and training data insuffient to directly model this distribution. Instead, we model the distribution $p(c|a, q)$, which attempts to measure the c, 'correctness', of the answer and question. We then introduce a hidden variable representing the class of the answer, e, (answer tag/named entity) as follows,

$$\begin{aligned} p(c|q,a) &= \sum_e p(c,e|q,a) \\ &= \sum_e \frac{p(c,e,q,a)}{p(q,a)} \\ &= \sum_e \frac{p(c|e,q,a)p(e,q,a)}{p(q,a)} \\ &= \sum_e \frac{p(c|e,q,a)p(e|q,a)p(q,a)}{p(q,a)} \\ &= \sum_e p(c|e,q,a)p(e|q,a) \end{aligned} \qquad (1)$$

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.

The distribution $p(c|e, q, a)$ is modelled utilizing the maximum entropy framework described in (Berger et al., 1996). The thirty-one features used in the model are described fully in (Ittycheriah, 2001) but the broad categories of features are presented here also. One feature represents the document rank as returned by our IR engine, twelve features are used on the sentence corresponding to the answer and the remaining eighteen are features on answer candidate strings.

The training data for the answer selection model is drawn from questions 251-450 of the TREC-9 test and the 200 questions of the TREC-8 test. The last 50 questions of the TREC-9 questions are used for a validation test of the model and the first 250 questions are used as a real test. This setup of the data is primarily motivated so that all data used in both training and test are generally available for all research sites.

6.1 Sentence Features

The sentence features allow the model to validate the original sentence ranking via a simpler weighted sum of scores approach. These features include for example the matching word score, thesaurus (*WordNet*) match, dependency arc match score, LCA expanding word score, and the cluster match score.

6.2 Entity Features

These features on the answer candidates reflect the finding of the desired entity and also help to overcome failures in named entity marking. The features incorporate knowledge about finding the entity, the focus being present, and whether the answer candidate has a proper noun, digit or date. The Candidate DB Focus is a feature that fires when a word that occurs next to the question focus is found in the answer candidate. In the example above, the feature fires for Mississippi because elsewhere in the text the word occurs next to "River". The feature is most useful when an answer satisfies the focus somewhere in the text and then subsequently the answer is used without the focus.

6.3 Definition Features

Definitional questions request an elaboration on a concept or given a concept requires the term of the definition. These questions are largely outside the scope of named entity analysis and focus methods. The questions are simple to answer when there are only few instances of the term and the term is used primarily in the definitional context. However, this is often not the case and since the questions typically have only one major phrase that is being defined, there is very little the match type features can do. Using a dictionary resource such as WordNet can aid greatly in answering these type of questions. The collection of these features isolates various types of matches from *WordNet* glosses to items found during LCA expansion.

6.4 Linguistic Features

Is-Relationship This feature fires when the answer candidate is either the subject or object of a form of the verb "be" and has all the question words following.

Apposition This feature applies to answer candidates which are immediately preceded or followed by a comma and then a group of matching question words are found. This is similar in function to the *Comma_3_words_score* of the LASSO system (Moldovan and et. al., 1999) although in this case its even more restrictive in requiring all question words to be present.

Subject-Verb/Verb-Object When the question has non-stop verb (meaning important and uncommon), and the answer candidate is either in subject or object position, this feature fires.

These thirty-one features were examined for the answer selection problem and there is no feature that completely separates correct answers from incorrect for all questions. The features are mostly real valued (for example the matching IDF sum of question words) and has to be quantized into some bins. The bin widths are uniformly spaced between the maximum and minimum value for a feature. The IDF features are quantized into four bins. The non-IDF features, such as "Digit Score" are already quantized since they only test for the presence of a numeric quantity in the answer chunk. As an example of the quantization process, the following matrix shows the distribution of the feature Matching_Word_Score among correct (COR) and incorrect (INC) chunks. Note that the label match_0 only indicates that the answer was in the lowest bin, not that the number of words that matched was zero.

	match_0	match_1	match_2	match_3
COR	425	46	55	384
INC	1407	267	282	1035

The selected features come from considering these features individually and up to order 4 combinations. The features sorted by their weight and then choosing the top 10 and bottom 10 are are shown in Table 1. The MRR versus the number of features is shown in Table 2.

These results indicate that the best performance is at about 168 features and it represents a 12.9% improvement over the baseline ranking. A few of the features are worth examination. First, the future "1" indicates a correct answer. All features with weight greater than 1.0 are associated with the "correct" class (1). This is because the prior distribution is strongly weighted for incorrect answers and the model needs a lot of features to overcome the prior distribution. The first feature indicates "CARDINAL_CARD", which means that the question desires an answer with a number and that the answer candidate has a numeric word. The feature "miss_0_candent_1_PERSON_PERSON_candarmatne_1" is a complex feature that requires that the answer candidate not have any missing words, it must have the desired entity, the desired answer tag and entity found must be PERSON and that the entity has a parse link structure to a question word. Intuition says that this probably indicates a correct answer and indeed it weights it with a weight greater than 1 and thus boosts the score if the model is predicting a correct answer. At the other end of the weights, features like "LOCATION_PERSON" indicate that if the desired answer tag is a LOCATION and the answer candidate has a PERSON, then it probably is not a correct answer (it weights the score down if this feature fires). The ability to interpret these features is a strength of the maximum entropy approach. The weight in general can not be evaluated, except that large deviations from 1.0 indicate either positive correlation with the future (when greater than 1.0) and negative correlation with the future (when significantly less than 1.0).

7 TREC-10 Results

In this years evaluation, we used our three submissions to evaluate the effect of various amounts of query expansion. The results are displayed in Table 3. Essentially, the results seem to indicate that increasing the amount of query expansion in our IR engine results in poorer overall performance. These experiments were done to determine whether a basic search engine should be modified to improve question-answering and the results indicate that at least for our search engine (Franz et al., 1999), question answering can be performed as a post-processing of the IR results. These results represent a 34.5% improvement over our 50 byte results in TREC-9.

8 Development Set Analysis

In this section, we focus the analysis on the system ibmsqa01a. There were surprisingly 30 out of 500 questions which only one system answered correctly. Of these, 7 were answered by the ibmsqa01a system. In order to reserve a true test set for next years evaluation, we chose to look only at the first 200 questions. Two examples from this development portion where our system produced answers and there was no other systems with correct answers are shown below.

Q: What do you call a newborn kangaroo?
A: 960 Q0 AP891022-0031 2 4.6789 ibmsqa01a - inch - tall baby _ called a joey _ followed .

Q: How fast is alcohol absorbed?
A: 1065 Q0 SJMN91-06037052 3 8.6748 ibmsqa01a one hour to metabolize one ounce of alcohol .

There were however 37 questions that our system blundered on, by which I mean that we produced no answer when 10 systems were able to get the correct answer. An example of such a question is "What is caffeine?", where our system got the word "coffee" from the definition in *WordNet* and produced answers which contained the word. The correct answer, 'alkaloid', is also in the *WordNet* definition, but the system preferred answers with coffee. Another example of a question where the system failed was "How many liters in a gallon?" to which our system produced as the best sentence "There are 3.8 liters in a gallon.", but then during the answer extraction (50 byte), this answer was thrown away because answer tag we searched for was CARDINAL and the answer contained a MEASURE. This can be considered as a failure in the answer tag selection, but it could be corrected by the answer selection if we had sufficient examples in our training data where such a mapping was desired.

9 Conclusions and Future Work

Our statistical question answering system showed significant improvement over the year (34.5%). This improvement was largely dominated by the inclusion of query expansion in answer selection and modest improvements were obtained by using a statistical algorithm for answer selection. The trainability of the answer selection still suffers from lack of training material so for our next system we are attempting to increase the training set by an order of magnitude more questions.

History	Future	Weight
CARDINAL_CARD	1	2.67586
GEOLOGICALOBJ	1	2.04765
candmaxmat_2	1	2.04757
MEASURE_MEASURE	1	1.80068
arcmat_0_candnumglossexp_1	1	1.61033
canddate_1_candarmatne_1	1	1.54951
miss_0_candent_1_PERSON_PERSON_candarmatne_1	1	1.53813
clusterscore_1_prevscore_1_no_ne_match_PHRASE_X	1	1.52065
candfoc_1_candnumdefnexp_1	1	1.42604
match_0_miss_0_candnumdefnexp_1	1	1.40497
miss_0_candfoc_1_candarmatne_1_candnumdefnexp_1	0	0.62682
arcmat_0_ne_map_match	1	0.626688
exact_ne_match_candmaxmat_1	0	0.570918
no_ne_match_canddist_2	1	0.570399
arcmat_0_clusterscore_1_no_ne_match_candarmatne_1	1	0.560376
TITLE_WORK_X	1	0.516318
ne_map_match_docrank-1	1	0.505354
DATE_X	1	0.401585
PERSON_X	1	0.369949
LOCATION_PERSON	1	0.302307

Table 1: Maximum Entropy features selected for answer selection.

Number of Features	Baseline	48	72	96	120	144	168	192
TREC9 MRR (q450-500)	0.458	0.487	0.487	0.489	0.496	0.509	0.517	0.496

Table 2: Maximum Entropy Performance versus number of features.

		Strict		Lenient	
System	Description	MRR	Num Missed	MRR	Num Missed
A	No query expansion in IR, Ency query expansion in Answer Selection	0.390	218	0.403	212
B	Ency query expansion in IR and Answer Selection	0.390	220	0.403	215
C	Ency and *WordNet* query expansion in IR and Answer Selection	0.375	231	0.388	224

Table 3: Performance on TREC-10.

10 Acknowledgement

This work is supported 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.

M. Franz and S. Roukos. 1998. TREC-6 ad-hoc retrieval. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*. NIST Special Publication 500-240.

M. Franz, J. S. McCarley, and S. Roukos. 1999. Ad-hoc and multilingual information retrieval at ibm. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*. NIST Special Publication 500-242.

Sanda Harabagiu and et. al. 2000. Falcon: Boosting knowledge for answer engines. *TREC-9 Proceedings*, pages 50–59.

Abraham Ittycheriah, Martin Franz, Wei-Jing Zhu, Adwait Ratnaparki, and Richard Mammone. 2001. Question answering using maximum entropy components. *The Second Meeting of the North American Chapter of the Association of Computational Linguistics, Pittsburgh, PA*, pages 33–39.

Abraham Ittycheriah. 2001. *Trainable Question Answering Systems.* PhD Thesis, Department of Electrical and Computer Engineering, Rutgers - The State University of New Jersey.

G. Miller. 1990. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–244.

Dan Moldovan and et. al. 1999. LASSO: A tool for surfing the answer net. *TREC-8 Proceedings*, pages 65–73.

Tomek Strzalkowski, Fang Lin, Jose Perez-Carballo, and Jin Wang. 1997. Building effective queries in natural language information retrieval. *ANLP*, pages 299–306.

Jinxi Xu and W. Bruce Croft. 1996. Query expansion using local and global document analysis. In *Research and Development in Information Retrieval*, pages 4–11.

IIT at TREC-10

M. Aljlayl, S. Beitzel, E. Jensen
Information Retrieval Laboratory
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616
{aljlayl, beitzel, jensen } @ ir.iit.edu

A. Chowdhury
AOL Inc.
chowdhury@ir.iit.edu

D. Holmes
NCR Corporation
David.Holmes@WashingtonDC.NCR.COM

M. Lee, D. Grossman, O. Frieder
Information Retrieval Laboratory
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616
{lee, grossman, frieder} @ ir.iit.edu

Abstract

For TREC-10, we participated in the adhoc and manual web tracks and in both the site-finding and cross-lingual tracks. For the adhoc track, we did extensive calibrations and learned that combining similarity measures yields little improvement. This year, we focused on a single high-performance similarity measure. For site finding, we implemented several algorithms that did well on the data provided for calibration, but poorly on the real dataset. For the cross-lingual track, we calibrated on the monolingual collection, and developed new Arabic stemming algorithms as well as a novel dictionary-based means of cross-lingual retrieval. Our results in this track were quite promising, with seventeen of our queries performing at or above the median.

1 Introduction

For IIT at TREC-10, we focused on the adhoc tasks (both automatic and manual), the site finding task, and the Arabic cross-lingual tasks. For the adhoc tasks, our system is quite different from last year. We calibrated with different fusion approaches and found that a single similarity measure outperformed our other approaches. We also worked with the NetOwl entity tagger to improve our phrase recognition. In the manual track, we developed a new user interface to assist our manual user.

Our results for the Arabic cross-lingual track were quite promising. We developed a new stemmer and made use of a dictionary-based algorithm that requires the translation of the term to be equivalent when going from Arabic-English and from English-Arabic.

Finally, we participated in the web site finding track. We tested a variety of simple approaches, but unfortunately, our results were not very impressive. We are conducting failure analysis on this track to include in the final paper.

2 Adhoc

For TREC-10's ad-hoc task, we focused on effectiveness for short queries. We did a variety of calibrations after TREC-9 on the utility of fusion of various IR approaches. We found that when the stop word lists and parsers are kept constant and effective ranking strategies are used, essentially similar result sets occur for a variety of similarity measures and improvements in average precision due to fusion are negligible. We published this result [7], and for TREC-10, focused on a single similarity measure.

In this section, we briefly describe our query-processing techniques: the use of automatic statistical phrase weighting based on query length and the use of entity tagging for query terms. In the last section, we present our TREC 10 ad-hoc results including some of our results from fusion.

2.1 Query Processing

Many different strategies are used to improve the overall effectiveness of an IR system. Several examples are automatic term weighting [1, 2] and relevance feedback [3]. Phrases are frequently suggested as a means for improving the precision of an IR system. Prior research with phrases has shown that weighting phrases as importantly as terms can cause query drift [5] and a reduction in precision. To reduce query drift, static weighting factors are applied to a phrase reducing the contribution of importance to a documents ranking. These static weighting factors were shown to yield slight improvements in effectiveness [4, 5]. This year we applied two techniques to improve phrase processing. The first is an automatic phrase-weighting algorithm based on the query length and the second is entity tagging using SRA's NetOwl tagger to determine what phrases to use for search.

2.2 Automatic Statistical Phrases Weighting Based on Query Length

Statistical phrases are frequently identified at index time by identifying two term pairs that occur at least FIX THIS →X times and do not cross stop words or punctuation. Twenty-five is commonly used as a threshold for the number of documents a phrase must occur in before it is considered a statistical phrase [5].

While the use of phrases is a precision enhancing technique, their naïve usage generally reduces IR effectiveness. When multiple phrases are evaluated for a given query, the likelihood of query drift increases. This drift is caused by phrases overemphasizing a given document that does not contain a breadth of the attributes but only a highly weighted phrase. For an example query of "oil company law suits", the phrases: "oil company", "company law" and "law suits" will overemphasize documents not containing all the terms or phrases and cause nonrelevant documents to receive a higher ranking. This overemphasis causes query drift and the precision of a system decreases. To correct this, we introduce a damping factor of *(exp(-1*delta*queryLength)* and apply it to the actual contribution any phrases can supply to a given document. In Equation 1 the complete weighting for a phrase is given.

$$\sum 1+\left(\frac{1+\ln(1+\ln(tf))}{(.8+.2*(docsize/avgdocsize))}*\exp(-1*\mathrm{delta}*\mathrm{queryLength})\right)*nidf*qtf$$

Equation 1: Phrase Ranking Algorithm

Where:

- tf = frequency of occurrences of the term in the document
- qtf = frequency of occurrences of the term in the query
- $docsize$ = document length
- $avgdoclength$ = average document length
- N = is the number of documents in the collection
- n = is the number of documents containing the word
- $nidf$ = log(N+1/n)

Our hypothesis is that as the number of phrases increase for a query, the likelihood of query drift due to a highly weighted phrase increases. Thus, by adaptively weighting phrases based on query length, we can improve precision by reducing the likelihood of drift. We ran tuning experiments with the TREC 6, 7 and 8 short (title only) queries. We measured the effectiveness of the various runs with no phrases and phrases with various static weights and dynamic weights.

By keeping the phrase weight set to one (equivalent to the weight given to terms) our average precision is reduced by almost 5%. Other researchers have experienced this same result [4, 5]. By reducing our phrase weight by a factor of .5 and .25 our effectiveness improves. While other groups have chosen a fixed static weight of 0.5, short queries continue to improve to 0.25. Table 1 shows the average precision for phrase weights of 1, .5, and .25. Our adaptive phrase weighting enables us to avoid tuning for phrases. A dynamic weighting based on query length determines the likelihood that the phrase will contribute to the weight. Our dynamic approach yields an improvement of 12% over the statically tuned approach on average for the 150 queries. All IIT runs this year use the given phrase weighting approached described above.

	No Phr	Pwt - 1	Pwt - .5	Pwt - .25	Pwt - Sig	No->.25	No->Sig
T6	22.37%	21.02%	22.59%	23.03%	23.13%	2.95%	3.40%
T7	17.57%	15.51%	16.94%	17.68%	17.73%	0.63%	0.91%
T8	23.85%	24.09%	24.47%	24.58%	24.60%	3.06%	3.14%
Avg	21.26%	20.21%	21.33%	21.76%	21.82%	2.21%	2.48%

Table 1: Phrase Weighting Evaluation Runs (Short Queries)

2.3 Ad-Hoc TREC 10 Experiments

Our overall results for Trec-10 Ad Hoc experiments are summarized in the following chart.

Above Median	At Median,	Below Median
32	1	17

For all queries, we used our new weighted statistical phrase processing. In addition, for indexing, we used a modified porter stemmer and conflation class stemming system. This year's baseline title only experiment was iit01t. For our submitted run, we used a modified pivoted document length ranking strategy. We used Rocchio positive feedback using 15 terms from the top 10 documents selected in pass one and each new query term was given a factor of .25. In addition, we used the TREC disks 4-5 for collection enrichment with Rocchio positive feedback of 15 terms from the top 10 documents and a weighting of 0.15. Our run with feedback and collection enrichment is shown in Figure 1 below.

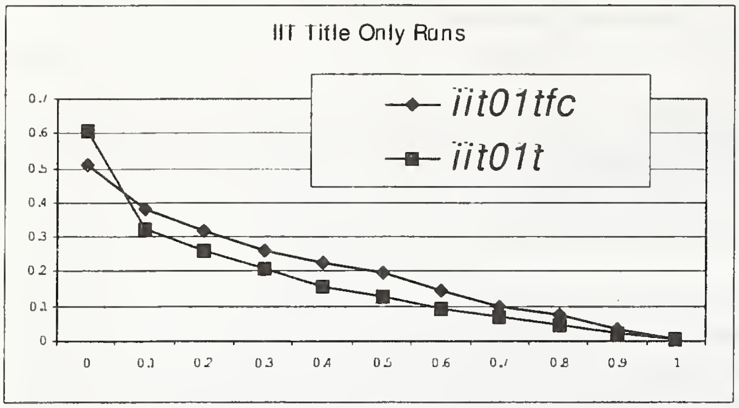


Figure I: Title only runs

2.4 Query Entity Tagging

We also tested the impact of using an entity tagger over statistical phrases. Tagging a large document collection is difficult with existing entity-taggers because they are not designed for scalability. We were able to tag queries very quickly. The idea was to take the entities tagged in the query and derive two-term phrases from these entities. Hence, a query with "federal housing authority" that has this tagged as a single entity would result in the phrases "federal housing" and "housing authority" to be derived from this tag.

We encountered several problems with this approach. Many queries are not long enough for entity taggers to accurately tag the query terms. Worse, not all queries contain entities that provide useful knowledge of which phrases to use for query processing. To further examine our strategy we used the description of the query instead of only the short titles. Only five of the fifty queries contained entities that were tagged by the NetOwl tagger that could be used for query processing. The five queries and their tags are shown in Table 3. When an entity was encountered, all terms within it were combined as phrases. For query 505 "Edmund Hillary" is identified as a useful phrase, for query 510, "J. Robert Oppenheimer" is found, and for query 527 "Booker T. Washington" is identified as a single phrase. Finally, query 538 has "Federal Housing Authority". Because our index includes only two term phrases, we generate two term phrases from these entities. Future work will focus on tagging the entities in the corpus for indexing. That way, Washington as a name will be distinguished from Washington as a place in both the queries and the index and can be used as a filter.

QUERY 505: WHO IS/WAS <PERSON TYPE="PERSON" FIRSTNAME="EDMUND" LASTNAME="HILLARY" GENDER="MALE">EDMUND HILLARY</PERSON>?
Query 510: Find biographical data on <PERSON TYPE="PERSON" FIRSTNAME="J. ROBERT" LASTNAME="OPPENHEIMER" GENDER="MALE">J. Robert Oppenheimer</PERSON>.
Query 515: What did <PERSON TYPE="PERSON" FIRSTNAME="ALEXANDER GRAHAM" LASTNAME="BELL" GENDER="MALE">Alexander Graham Bell</PERSON> invent?
Query 527: What biographical data is available on <PERSON TYPE="PERSON" FIRSTNAME="BOOKER T." LASTNAME="WASHINGTON" GENDER="MALE">Booker T. Washington</PERSON>?
Query 538: Find documents describing the <ENTITY TYPE="ENTITY" SUBTYPE="GOVERNMENT">Federal Housing Administration</ENTITY> (<ENTITY TYPE="ENTITY" SUBTYPE="GOVERNMENT">FHA</ENTITY>): when and why it was originally established and its current mission.

Table 2: Entity Tagged Queries

2.5 Summary

For TREC-10's ad-hoc task, we focused on effectiveness for short queries for the web track. This year we focused on query processing techniques and fusion approaches. Our initial results are both positive and negative in nature with an overall strong performance in the adhoc title-only task Thirty-two queries of fifty were judged over the median.

3 Manual Task

For the manual WEB Track, IIT expanded upon work from prior years. Our overall results are summarized in the following chart.

Above Median	At Median	Below Median
33	3	14

Research focused on the use of concepts and manual relevance feedback. Additionally, a new user interface was developed. As with previous years, we implemented required and scoring concepts. All fifty topics had at least one required *concept*. A *concept* is represented as a set of words from which a document must contain at least one word. Eighteen topics contained two required concepts (documents must contain at least one entry from each list. Forty-six topics have scoring concepts, or concepts that contribute to relevance but do not identify new documents. Table 3 summarizes our experiments related to concepts. While the use of multiple required concepts only provided a modest boost to average precision, the probability of achieving the best average precision doubled. The median average precision for all teams was 0.1665 for our topics with two required concepts, while the median was 0.1997 for topics where we used one topic, indicating the two concept topics were somewhat more difficult.

Required Concepts	Number of Queries in Set	Avg Precision	Best	At or Above Median, not best	Below Median
1	32	0.3226	7	17	8
2	18	0.3499	8	5	5

Table 3: Average Precision for Manual Queries

We also tested the effect of manual relevance feedback. Manual relevance feedback involved reading some number of documents and selectively modifying queries based upon what was read. To do this, we split the topics into three groups. For the most "top" group, we read at least 100 documents per topic, with a maximum of 156. For the middle group we read between 50 and 99 documents. Finally, we read from zero to 49 documents for the group with minimal relevance feedback. We reviewed a little under 10% of returned documents. Table 4 summarizes the results for manual relevance feedback. It can be seen that reading numerous documents had an impact on whether or not we had the best query.

Documents Read	Number of Queries in Set	Avg Precision	Best Avg Precision	At or Above Median, not best	Below Median
100+	10	0.4714	7	2	1
50-99	25	0.3187	4	15	6
0-49	15	0.2628	4	5	6

Table 4 Manual Relevance Feedback Results

Final results were re-ranked based upon user assessment. User assessed "Relevant" documents contained all elements of topic, "Probably Relevant" contained most elements, or loosely addressed all elements. Documents assessed "Probably not relevant" contained some reference to the topic but did not seem related, while "Not Relevant" were completely unrelated. Table 5 below shows our in-house assessments of the result documents.

User Assessment	Documents	Ranking Adjustment
Relevant	598	Ranked above all other documents returned
Probably Relevant	523	Relevance score boosted by 0.25
Probably not Relevant	612	Relevance score lowered by 0.5
Not Relevant	1678	Relevance score lowered by 0.9

Table 5 Relevance Assessments from our Manual User

4 Homepage Finding

This year our group participated in the new site finding task. For a baseline run, we indexed the title terms from the document collection and ran an initial query pass using our basic adhoc retrieval strategy. In addition, the source URL's for each result document were cleaned to remove extraneous words and characters so they would adhere to a typical URL format. After having retrieved the results from our initial query pass, we used three techniques to augment and improve the result set: TAND, Co-occurrence Boosting, URL-folding.

4.1 TAND Initial Results at Thirty Percent

The results from the initial query pass were TAND'ed. In order for a candidate result document to remain in the result set, it had to contain a minimum of thirty percent of the query terms. This technique was used as a coarse-grained filter, eliminating result documents that had little chance of being relevant. We arrived at thirty percent and all other thresholds by calibrating with the training site-finding set.

4.2 Boosting on Result Co-Occurrence

Along with our primary title-only index, we created several other indexes that were used for a form of collection enrichment. These included:

- ODP Descriptions – We crawled the hierarchy of the Open Directory Project (www.dmoz.org) and created an index of the description terms for each entry.
- ODP Anchor Text – An index of the anchor text used for hyperlinks in the Open Directory Project
- First-100 – An index of the first one hundred terms from each document in the WT10G.

After the TAND'ing of the result sets from the initial query pass was complete, we ran a query pass against each of these three indexes, and used the following algorithm to "boost" results in the initial result set:

- For the top thirty results from the ODP description query, we checked the URL for the result document in question against the result set from our initial query pass.
  - If it was present in the initial query pass, the score for the document in the initial result set was increased by 85%
  - If it was not present in the initial query pass, but a document with the same URL was confirmed to exist in the WT10g collection, that document was added to the initial result set with the unmodified weight from the ODP Description result set.
- This process was repeated for the two additional indexes in the following order, with the following parameters:
  - ODP Anchor Text: Examined the top sixty results and boosted matches by 50%
  - First-100: Examined the top sixty results and boosted matches by 60%

TAND'ing and Boosting improved our baseline mean reciprocal rank by approximately 70%. It should be noted that the order in which the boosting indexes were queried is very important, as potential results could have been boosted multiple times depending on which source located them first.

The order in which the boosting indexes were queried, and the various boosting factors and number of results examined were determined experimentally by performing a large number of calibrations using the supplied training data for the Homepage finding task. Essentially, the numbers describe the measure of confidence we placed in the ability of each source to yield relevant results. We found that the ODP indexes, potentially due to the large amount of human oversight and interaction, were trustworthy. By contrast, the index of the first one hundred terms was shown to be less likely to contain highly relevant results, probably due to the presence of large quantities of "noise" information that is often present in the first terms of a web page, such as advertisements, etc.

4.3 Folding

The final technique we used on the boosted result set was our URL-folding algorithm. The idea here is to combine results from the same site in the ranked list so as to order them in a reasonable way. We refer to pages on a web site in terms of *parent-child* relationships. A parent page is shallower in the site hierarchy (e.g.; ir.iit.edu) while a child page is deeper (e.g.; ir.iit.edu/researchers). Folding took place as follows:

a. Parent occurs higher in the result set than child: child is removed from result set and parent's score is increased
b. Child occurs higher than parent: parent score is increased, but child is left in its original position of the result set.

Relevance score modifications were performed for each parent according to the following equation:

$$S_p = S_p + \ln\left(\sum S_c\right)$$

Equation 2: Parent Weight Incrementation

After experimenting with this scheme, we found a paradox: Many parent pages had too many children above them in the rankings, but increasing the increments by which parents were weighted caused parents with many children to be ranked too highly. To provide finer-grained tuning of how parents had their ranks increased, we added a final step to our algorithm that occurred after all folding had been completed. In this step, we moved parent pages that had unfolded child pages of within 35% of the parent's score just above those unfolded children in the result ranking. We also guaranteed that parent pages had at most three unfolded children above them in the ranking, regardless of their relevance.

After attending TREC, we performed some failure analysis on our techniques, in an effort to discover why there was such a large disparity between our performance on the training queries, and our performance on the supplied topics. This failure analysis revealed some deficiencies in our query and document parsers, and also confirmed that there is a high degree of overlap in the improvements observed from our boosting and folding techniques.

Our experimental results for both the training data and the actual homepage topics for each approach are shown in Table 6. The improvements resulting from our post-conference failure analysis are also included. All values express the mean reciprocal rank over the query set.

Query Set	Baseline	Baseline + Boosting	Baseline + Folding (iit01st)	Baseline + Boost + Fold (iit01stb)
Training Data	.590	.725	.670	.880
Homepage Topics	.253	.503	.559	.578
Topics - Improved	.373	.519	.561	.664

Table 6 Results of Site Finding Task (MRR)

5 Arabic Monolingual and Cross-lingual Track

For the Cross-Lingual Arabic Information retrieval, our automatic effort concentrated on the two categories; English-Arabic Cross-Language Information Retrieval (CLIR) and monolingual information retrieval. For the English-Arabic CLIR we used two types of dictionary-based query translation: Machine-Readable Dictionary (MRD) and Machine Translation (MT). The First-Match (FM) technique is used for term selection from a given entry in the MRD [8].

5.1 Monolingual

For the monolingual run, we used two stemming algorithms. The first algorithm is root-based, and second is light stemming. In the root-based algorithm, the main aim is to detect the root of the given word. When no root is detected, the algorithm retains the given word intact. The root-based algorithm is aggressive. For example, the root of *office*, *library*, *book*, and *write* is the same, thus, the root-based algorithm places these in the same conflation class. Accordingly, a light-stemming algorithm is developed. It is not as aggressive as the root-based algorithm. The idea of this technique is to strip out the most common affixes to the Arabic words. For example, it returns the plural, dual to their singular form except for irregular pluralization.

Our monolingual run is described in Table 7. This run did reasonably well, with 21 queries above the median, 1 at the median and three below.

Average Precision								
Best	**Median**	**Worst**	**iit01mlr**	**Above**	**At**	**Below**	**Best**	**Worse**
0.5118	0.2516	0.0216	0.4288	21	1	3	3	0

Table 7 Monolingual run Using Light Stemming

5.2 English-Arabic Cross-Language information Retrieval

We conducted our experiments by using two approaches for query translation. The first approach is the Machine-Readable Dictionary (MRD). The second approach is Machine Translation (MT). In MRD realm, we use the first match in the bilingual dictionary as the candidate translation of the source query term. This approach ignores many noise terms introduced by the MRD. Al-Mawrid English-Arabic is used for the translation process [9].

In MT realm, the translation was performed on every field of the topic individually. We performed our experiment by using a commercial MT system product. It is called Al-Mutarjim Al-Arabey. It is developed by ATA Software Technology Ltd [10]. The post-translation expansion technique is used to de-emphasize the extraneous terms that are introduced to the source query after translation.

Our cross-lingual run is described in Table 8. Our run has 17 queries above the median, zero at the median and eight below. There are 3 queries where our run is the best.

Average Precision								
Best	Median	Worst	iit01xma	Above	At	Below	Best	Worse
0.5623	0.1701	0.0001	0.3119	17	0	8	3	0

Table 8 CLIR result using Mutarjim Al-Arabey MT system

REFERENCES

[1] Salton G., C. Yang and A. Wong. "A vector-space model for information retrieval", *Comm. of the ACM*, 18, 1975.

[2] A. Singhal, C. Buckley, and M. Mitra, "Pivoted Document Length Normalization", ACM-SIGIR *Conference on Research and Development in Information Retrieval*, 1996.

[3] J. Rocchio. "Relevance Feedback in Information Retrieval. *Smart System - Experiments in Automatic Document Processing*", pages 313--323. Prentice Hall, Englewood Cliffs, NJ, 1971.

[4] A. Turpin and A. Moffat. "Statistical Phrases for Vector-Space Information Retrieval", ACM-SIGIR, 1999: 309-310.

[5] M. Mitra, C. Buckley, A. Singhal, and C. Cardie. "An Analysis of Statistical and Syntactic Phrases", *Fifth RIAO Conference, Computer-Assisted Information Searching On the Internet*, 1997.

[6] TREC at NIST (National Institute of Standards and Technology) trec.nist.gov

[7] A. Chowdhury, D. Grossman, O. Frieder, C. McCabe, "Analyses of Multiple-Evidence Combinations for Retrieval Strategies", ACM-SIGIR, September 2001.

[8] M. Aljlayl and O. Frieder. "Effective Arabic-English Cross-Language Information Retrieval via Machine Readable Dictionaries and Machine Translation," *ACM Tenth Conference on Information and Knowledge Managemen (CIKM),* Atlanta, Georgia, November 2001.

[9] Dar El-Ilm Lilmalayin, http://www.malayin.com/

[10] http://www.atasoft.com/products/mutarjim_v2.htm

Multi-timescale video shot-change detection

Marcus J Pickering and Stefan M Rüger
Department of Computing
Imperial College of Science, Technology and Medicine
180 Queen's Gate, London SW7 2BZ, England
m.pickering@doc.ic.ac.uk

Abstract

We describe our shot-boundary detection experiments for the TREC-10 video track, using a multi-timescale shot-change detection algorithm.

1 Introduction

The shot change detection system is based on that proposed by Pye et al [1] as part of a scene-change detection method which also took into account audio cues.

The algorithm aims to detect and distinguish different types of breaks in video by looking at differences between frames across a range of timescales. Looking at a wider range of frames than just those that are consecutive enables the detection of gradual changes such as fades and dissolves, while rejecting transients such as those caused by camera flashes.

Influenced by Zhang's twin comparison method [2], we added functionality to detect the start and end times of gradual changes to fulfil the requirements for the TREC submission.

In the remainder of this paper, we describe the shot-change detection system in detail, and present the results of the TREC evaluation.

2 System Description

The video segmentation algorithm is broadly based on the colour histogram method, which is extended 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 each corresponding block in two images 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.

Video	Known Trans	Mean Recall	Recall	Mean Prec	Prec
ahfl.mpg	63	0.961	0.936	0.919	0.921
anni009.mpg	38	0.870	0.789	0.735	0.697
bor03.mpg	230	0.856	0.982	0.920	0.953
bor08.mpg	379	0.920	0.873	0.894	0.945
bor17.mpg	127	0.905	0.952	0.843	0.975
nad28.mpg	181	0.917	0.939	0.853	0.762
nad31.mpg	187	0.892	0.866	0.834	0.900
nad33.mpg	189	0.951	0.952	0.882	0.918
nad53.mpg	83	0.962	0.951	0.876	0.797
senses111.mpg	292	0.902	0.989	0.909	1.000
ydh1.mpg	69	0.961	0.971	0.839	0.881
Weighted means		0.912	0.932	0.879	0.916

Table 1: Results of shot-boundary detection task for cuts in all files with greater than 100 overall transitions. The recall and precision values for our system are shown next to the respective means across all systems. Recall is the proportion of the known shot boundaries retrieved by the system and precision is the proportion of boundaries retrieved by the system which were judged to be correct. The bottom line shows the column mean for each of the statistics, with each file's contribution weighted by the number of transitions in that file.

A peak is defined as a value of d_n which is greater than a pre-defined threshold and is greater than the 16 preceding and 16 following values of d_n. A shot break is declared if there are near-coincident peaks of d_{16} and d_8. An additional coincident peak of d_2 suggests a cut, otherwise the break is classified as a gradual transition.

The algorithm thus far detects the presence of cuts or gradual changes, but gives no indication of the start and finish points of the gradual changes. We therefore employ a method similar to that described by Zhang [2] in which a lower threshold 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 it is greater than the threshold it is marked as a potential start of a transition. If, on examination of successive frames, the d_4 difference falls below the threshold 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 the d_4 change first falls below the threshold again, following the shot change. The d_4 timescale is used because it is thought to be fine enough to accurately pinpoint 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.

3 Results

The results show that our system performed slightly better than average overall. The breakdown of detected breaks into cuts (Table 1) and gradual transitions (Table 2) shows that, as for other systems, performance is considerably poorer for gradual transitions than it is for simple cuts. However, one of our best performances, in both precision and recall relative to the average, was

Video	Known Trans	Mean Recall	Recall	Mean Prec	Prec
ahf1.mpg	44	0.683	0.681	0.700	0.625
anni009.mpg	65	0.501	0.523	0.669	0.809
bor03.mpg	11	0.660	0.545	0.283	0.166
bor08.mpg	151	0.633	0.741	0.794	0.741
bor10.mpg	150	0.687	0.866	0.743	0.866
bor12.mpg	136	0.556	0.625	0.705	0.825
bor17.mpg	119	0.511	0.697	0.678	0.584
nad28.mpg	116	0.603	0.543	0.555	0.588
nad31.mpg	55	0.478	0.418	0.436	0.353
nad33.mpg	26	0.535	0.500	0.389	0.206
nad53.mpg	76	0.596	0.631	0.575	0.761
senses111.mpg	16	0.336	0.187	0.298	0.068
ydh1.mpg	52	0.492	0.423	0.620	0.536
Weighted means		0.581	0.641	0.653	0.674

Table 2: Results of shot-boundary detection task for gradual transitions in all files with greater than 100 overall transitions.

for "bor12.mpg" for which all breaks were gradual changes.

On closer examination of the test videos, it becomes clear that the relatively poor performance of the system on gradual transitions is due in large part to the difficulty in distinguishing object or camera motion from gradual transitions. One of the lowest results for precision was obtained for the video "senses111.mpg", and examination of the video shows that almost all of the false positives were caused by the camera following the subject as he moved across a highly contrasting background, with the rest being caused by object motion. A look at the missed transitions shows that they were almost all situations in which one technical drawing was dissolved into another containing identical colours. To detect this would have required an extremely low threshold which would have brought with it a whole raft of new false positives.

One of the poorest recall rates for the results shown was for "anni009.mpg". Here the problem seems to be caused by a confusion between cuts and gradual changes - virtually all of the falsely declared cuts corresponded to missed gradual transitions, which suggests that the thresholds were inappropriately set for this recording.

Another cause of reduced performance was that our detection of transition start and stop times did not match with those of the reference. In "bor12.mpg", for example, a large number of the gradual transitions appearing in the list of inserted breaks correspond to longer transitions appearing in the list of deleted transitions. This may point to below-optimum threshold setting, but TREC's determination of the reference itself was a subjective process, therefore the disparity is not necessarily due to an inherent flaw in our system.

4 Conclusions

While our system's performance was, on balance, slightly better than average, there were some obvious problems. It is clear that some discriminant is needed in addition to colour. If an object could be tracked across a supposed shot boundary, the boundary could be discounted, for example. Other discriminants such as texture and shape may also be helpful.

There was evidence that the empirically determined thresholds were not optimal in all cases, suggesting that some form of automatic threshold adjustment may be required. However, this would probably require two passes over the data.

Acknowledgements: This work was partially supported by EPSRC, UK, and AT&T Research Labs, Cambridge, UK.

References

[1] Pye D, Hollinghurst NJ, Mills TJ, Wood KR. Audio-Visual Segmentation for Content-Based Retrieval. 5th International Conference on Spoken Language Processing, Sydney, Australia, Dec 1998.

[2] Zhang HJ, Kankanhalli A, Smoliar SW. Automatic Partitioning of Full Motion Video. Multimedia Systems vol 1, 10-28, Jan 1993.

Link-based Approaches for Text Retrieval

Julien Gevrey and Stefan M Rüger
Department of Computing
Imperial College of Science, Technology and Medicine
180 Queen's Gate, London SW7 2BZ, England
gevreyjulien@hotmail.com and s.rueger@doc.ic.ac.uk

Abstract. We assess a family of ranking mechanisms for search engines based on linkage analysis using a carefully engineered subset of the World Wide Web, `WT10g` (Bailey, Craswell and Hawking 2001), and a set of relevance judgements for 50 different queries from Trec-9 to evaluate the performance of several link-based ranking techniques.

Among these link-based algorithms, Kleinberg's `HITS` and Larry Page and Sergey Brin's `PageRank` are assessed. Link analysis seems to yield poor results in Trec's Web Ad Hoc Task. We suggest some alternative algorithms which reuse both text-based search similarity measures and linkage analysis. Although these algorithms yield better results, improving text-only search recall-precision curves in the Web Ad Hoc Task remains elusive; only a certain category of queries seems to benefit from linkage analysis. Among these queries, homepage searches may be good candidates.

1 HITS

"*What other web pages find useful is likely to be useful to me as well.*" This approach is also known as Kleinberg's method (Kleinberg 1998; Chakrabarti, Dom, Gibson, Kleinberg, Raghavan and Rajagopalan 1998; Kleinberg, Kumar, Raghavan, Rajagopalan and Tomkins 1999) and differs drastically from traditional search engines whose ranking method is mainly based on the frequency of matching words. Documents that are pointed to by many other documents are called *authorities* and are ranked highly. Documents that point to many documents related to the query-topic are called *hubs* and may also be of interest. Good authorities are those that are pointed to by good hubs and good hubs point to good authorities: this is the mutually reinforcing relationship at the core of Kleinberg's `HITS` algorithm. `HITS` comprises two phases (Kleinberg 1998):

Phase 1 provides a small subgraph G of the web that is relatively focused on the query topic —it has many relevant pages, and strong authorities. This is done by taking the most highly ranked items of a text-based search (say, the top 200) as root set. This set is then expanded by adding, for each page in the root set, all its incoming links and a fraction of its out-going links. The set of pages thus derived is known as the base set.

Phase 2 consists of analysing the link structure of G to compute hubs and authorities in a mutually reinforcing relationship. This calculation is an iterative process. At step n, the hub value of a page is computed as the sum of the authority values (computed at step $n-1$) of its incoming links, and the authority value of a page is computed as the sum of the hub values (computed at step $n-1$) of its incoming links, ie,

$$a_{n+1}(u) = \sum_{(v,u) \in G} h_n(v)$$

and

$$h_{n+1}(u) = \sum_{(v,u) \in G} a_n(v),$$

where $a_n(u)$ denotes the authority value computed for page u at step n, $h_n(u)$ denotes the hub value computed for page u at step n and G denotes the graph whose nodes are web pages and whose edges are links from one page to another: $(v, u) \in G$ iff page v points to u.

2 PageRank

Larry Page and Sergey Brin's algorithm (1998, see also Page, Brin, Motwani and Winograd 1998) is said to be deployed in their successful search-engine `Google` (`http://www.google.com`). Like Kleinberg's algorithm, it focuses on the hyperlink structure of web pages.

Intuitively, we solve the recursive definition of authority: a page is authoritative if authoritative pages link to it. At each step of the recursion, a page v with outdegree N_v and authority value a_v pointing to some page u will confer on page u a fraction a_v/N_v of its own authority value. For each page in the collection,

$$a_{n+1}(u) = \sum_{(v,u) \in G} \frac{a_n(v)}{N_v},$$

where $a_n(u)$ denotes the authority value computed for page u at step n and G denotes the graph made from links between pages.

Like `HITS`, `PageRank` relies on an eigenvector calculation: eventually, the importance of each page reaches a limit, which happens to be its component in the principal eigenvector of the matrix corresponding to the above transform (the `HITS` matrix is the adjacency matrix of G, whereas the PageRank matrix is a weighted adjacency matrix where 1's are replaced by $1/N_v$'s).

3 Average and Sim

To add other link-based ranking approaches to our experiments we also suggest our own link-based algorithms. These are not fixed-point algorithms like `HITS` and `PageRank`, since they only comprise a single iteration. Our belief is that if the computation of authorities only takes a few steps, the information about the initial ranking is neither lost nor diluted by too much iteration. Furthermore, we want to allow pages to confer some authority on pages they point to.

The underlying idea of our algorithms is to combine the similarity measures obtained by text-only search with linkage analysis. Indeed, `HITS` and `PageRank` work on a root set (top results of a text-based analysis) and then assign to each page the same authority value for the link analysis to work on (because the fixed-point algorithm converges to the principal eigenvector of a given matrix no matter what the initial vector is). This results in losing part of the information (ranking and similarity measures) obtained via text analysis. Our algorithms, called Average and Sim, reuse similarity measures along with linkage analysis.

Average. The authority value of page p is the average over similarity measures of all incoming links:

$$\text{authority}(p) = \frac{1}{|\{q|q \rightarrow p\}|} \sum_{q \rightarrow p} \text{similarity}(q)$$

By assigning to p the average value of similarity measures of all pages pointing to p, we mean that we believe more in what others say about p than in what p tells about itself. We use the average instead of a simple sum, because our experiments with Trec-9 Web Track queries have shown that average performs far better. This may be due to the fact that the average lays more emphasis on the quality than on the quantity of incoming links as summing would do. Another advantage of computing the average is that results nearly become root-set size independent.

Sim. The authority value of page p is the similarity measure of page p plus the average over all similarity measures of incoming links (similarity value + authority value conferred by pages pointing to p):

$$\text{authority}(p) = \text{similarity}(p) + \frac{1}{|\{q|q \rightarrow p\}|} \sum_{q \rightarrow p} \text{similarity}(q)$$

The idea is much the same as what is presented for Average. With Sim, we take into account both what the page tells us about itself and what other pages have to say. The formula can be decomposed as follows: one term (similarity measure) for text-analysis of page p, one term (average similarity measure over all incoming links) for the authority that other pages confer to p. Owing to the use of the average, both terms are of the same order so that summing up these two terms makes sense.

4 Results Table

Table 1 presents the average precision of several algorithms for the fifty Trec-9 queries. The text-only baseline for linkage analysis is that of Managing Gigabytes: MG (Witten, Moffat and Bell 1999).

From these experiments with Trec-9 data, we have shown that linkage analysis does not improve the average precision of content-only search. The influence of several factors on linkage analysis has been studied carefully. We have shown that, for HITS and PageRank algorithms, the root set should be of medium size (500 pages or so), so that it contains enough relevant pages and not too many irrelevant items. As for the number of iterations, experiments show that it is not necessary to make HITS and PageRank converge; the best results are obtained with only a few iterations. A new approach that may be of interest would be to try to combine linkage analysis with the reuse of similarity measures, since these methods yield better average results than HITS and PageRank.

These first unsuccessful attempts to improve text-only search results by linkage analysis in Trec's Web Ad Hoc Task correspond to other results, such as of Singhal and Kaszkiel (2000), Gurrin and Smeaton (2000b), and Savoy and Rasolofo (2000). They also find disappointing

algorithm	root set size	number of iterations	root set expansion	reuse of similarity measures	average precision
HITS	200	40	yes	no	0.0115
HITS	500	40	yes	no	0.0055
HITS	1000	40	yes	no	0.0057
HITS	200	40	no	no	0.0276
HITS	500	40	no	no	0.0216
HITS	1000	40	no	no	0.0215
PageRank	300	40	no	no	0.0285
PageRank	400	40	no	no	0.0297
PageRank	600	40	no	no	0.0232
PageRank	1000	40	no	no	0.0214
PageRank	1000	2	no	no	0.0278
PageRank	1000	10	no	no	0.0299
Average	3000	1	no	yes	0.0506
Sim	3000	1	no	yes	0.0504
MG (baseline)	-	-	-	-	0.0770

Table 1: Link Analysis Results for the previous Trec-9 queries

average precisions for link-based methods. For instance, Gurrin and Smeaton (2000b) have carried out link-based experiments that made a distinction between structural links (that separate documents within a particular domain, exist to aid the user in navigating within a domain, and consequently are not seen as a source of authority judgements) and functional links (that link documents in different domains, and can be seen mostly as links from a source document to a target document that contains similar and, to the author's opinion, useful information). More emphasis was laid on functional links since they are thought to be the most authoritative ones. What Gurrin and Smeaton found in (2000a) and (2000b) is that their link-based method did not bring any improvement over text-only search. However, their conclusion is that these poor results are not necessarily due to their method. As they point out, the WT10g dataset may not be relevant to assess search-engines' performance. Indeed, they found while experimenting on WT10g that, in all, approximately 2% of the links were functional, while a large 98% were structural links, so that the lack of functional links seriously hampers their experiments. Other criticisms have been uttered by Google's co-founders Larry Page and Sergey Brin, who write in (1998) that they do not believe that using a small set such as WT10g (compared to the real WWW) allows us to evaluate how a search-engine would perform while working on the real World Wide Web.

Since Average and Sim are the link-based methods that yield the best link-based results in our experiments, we decided to run these two algorithms for the Web Ad Hoc Task submission of Trec 2001 (icadhoc1 and icadhoc2). Text-only search results were submitted as baseline in icadhoc3. Table 2 displays the results after evaluation – they are of the same quality and quantity as the ones for previous year's queries in Table 1.

Are there query subsets which are likely to benefit from link-analysis? Our experiments in the Web Ad Hoc Task have shown that some queries are good candidates for link analysis; the

algorithm	average precision
Average (icadhoc1)	0.0537
Sim (icadhoc2)	0.0458
MG (baseline icadhoc3)	0.0883

Table 2: Link Analysis Results for the unseen Trec 2001 queries

retrieval performance for these queries being consistently improved by link-based methods no matter what algorithm was deployed. This may be due to the fact that the underlying hyperlink structure is particularly adapted. In this respect, the study of web communities, as Kleinberg does in (Gibson, Kleinberg and Raghavan 1998), may be a means of determining topics and communities for which link-structure is suitable for link-based retrieval techniques. However on average, given the Trec-9 queries, this improvement fails to show.

5 Home Page Finding

An anchor-text algorithm: Anchor. The underlying idea is to consider web pages as hubs for a given topic. MG is used to search an "anchor-text collection" built by indexing the anchor-texts of each page in WT10g. The top-ranked pages of the anchor-text collection can be thought of as being good hubs for the query, since they contain many anchor-texts related to the query. This allows one to compute hub values for each page in the collection. The authority value is then computed by summing up hub values of all incoming links:

$$\text{authority}(p) = \sum_{q \to p} \text{hubvalue}(q)$$

Only 17 pages were retrieved before rank 100 using Anchor, and the average rank over 17 found homepages is 23.82.

This is a very poor result compared to the 57 homepages found before rank 100 by text-only search (MG), and the average rank over these 57 found homepages is 16.47.

Rank Merging. On average, the Anchor algorithm performs far worse than text-only search. However, it helps retrieve more efficiently home pages that are ranked poorly (below rank 100) by text-based search. This disjointedness of behaviour is a good reason to suggest a merging of the ranked lists for an overall better result. Indeed, rank merging can improve text-only search in the Home Page Finding Task: with a particular merging heuristics we managed to retrieve 61 homepages with an average rank of 17.59.

To obtain this result, we merged MG's and Anchor's ranking lists as follows: the first seven retrieved items from MG and the first seven retrieved items from Anchor come first (interleaved), followed by 35 items from MG and the rest is completed with items retrieved by Anchor. This heuristics is based on the observation that the top few results from MG and Anchor are often good. The 35 following items are from MG's ranking list since on the whole MG performs better than Anchor and retrieves home pages before the top-40 ranked pages (if it is to retrieve them before rank 100). The last 51 items come from Anchor's ranking list because Anchor may help bring interesting pages into the top 100 that are not retrieved by MG.

These experiments show that link-based methods can help improve content-only search in home page finding. However, we have not relied on `Anchor` only to obtain this result; a rank-merging technique combining both text-based results and anchor-text-based results had to be deployed. Also, the Rank-Merging heuristics was chosen so that it improves the behaviour on previous year's Trec queries. It had to be seen with this year's queries whether this heuristics is generically helpful.

Hence, we submitted the link-based `Rank Merging` for Trec 2001's Home Page Finding submission (`ichp1`) and text-only search (`MG`) results as baseline (`ichp2`). It turns out that the baseline algorithm fares better in terms of the Trec evaluations than Rank-Merging (average reciprocal rank over 145 topics 0.237 vs 0.208).

6 Conclusion

In the Web Ad Hoc Task, we have shown that `HITS` and `PageRank` alone can not improve text-only performance. We have studied the influence of the expansion process, the root set size and the number of iterations on `HITS` and `PageRank`. We found that — in the context of Trec's `WT10g` — expansion is not efficient, that the root set should be of medium size to contain enough relevant documents without having too many irrelevant pages and that a small number of iterations is often better than convergence. We suggested two algorithms, `Sim` and `Average`, which combine reuse of similarity measures of text-only search with linkage analysis, and yielded better results than `HITS` and `PageRank`. However these results are still below `MG`'s text-only search results.

The Home Page Finding Task's experiments illustrate how anchor-text can be efficiently used to retrieve home pages. The relevance of anchor-text in the Home Page Finding Task may be due to the fact that there is little ambiguity in labelling a link that points to a home page; all one has to do is to name the entity, individual or organisation one wants to refer to. By deploying a rank-merging technique, we have seen some anecdotal (but no general) evidence of improvement through link-analysis.

References

P Bailey, N Craswell and D Hawking (2001). Engineering a multi-purpose test collection for web retrieval experiments. In *Notebook Text Retrieval conf 2001.* NIST.

S Brin and L Page (1998). Anatomy of a large-scale hypertextual web search engine. In *Proc 7th WWW conf.*

S Chakrabarti, B Dom, D Gibson, J Kleinberg, P Raghavan and S. Rajagopalan (1998). Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proc 7th WWW conf.*

D Gibson, J Kleinberg and P Raghavan (1998). Inferring web communities from link topology. In *Proc 9th ACM conf on Hypertext and Hypermedia.*

C Gurrin and A F Smeaton (2000a). A connectivity analysis approach to increasing precision in retrieval from hyperlinked documents. In *Proc 8th Text Retrieval conf*, pp 357–366. NIST.

C Gurrin and A F Smeaton (2000b). Dublin city university experiments in connectivity analysis for trec-9. In *Notebook 9th Text Retrieval conf*, pp 190ff. NIST.

J Kleinberg (1998). Authoritative sources in a hyperlinked environment. In *Proc 9th ACM-SIAM Symposium on Discrete Algorithms.*

J Kleinberg, R Kumar, P Raghavan, S Rajagopalan and A S Tomkins (1999). The web as a graph: measurements, models and methods. In *Proc 5th Annual Intl Conf Computing and Combinatorics.*

L Page, S Brin, R Motwani and T Winograd (1998). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford, Santa Barbara. http://www-db.stanford.edu/~backrub/pageranksub.ps.

J Savoy and Y Rasolofo (2000). Report on the trec-9 experiment: Link-based retrieval and distributed collections. In *Notebook 9th Text Retrieval conf*, pp 472ff. NIST.

A Singhal and M Kaszkiel (2000). At&t at trec-9. In *Notebook 9th Text Retrieval conf*, pp 134ff. NIST.

I H Witten, A Moffat and T C Bell (1999). *Managing Gigabytes.* Morgan Kaufmann.

Acknowledgements: This work was partially supported by EPSRC, UK.

Applying Support Vector Machines to the TREC-2001 Batch Filtering and Routing Tasks

David D. Lewis
Independent Consultant
858 W. Armitage Ave., #296
Chicago, IL 60614
Dave@DavidDLewis.com
www.daviddlewis.com

1. Introduction

My goal for TREC-2001 was simple: submit some runs (so that I could attend the conference), spend the minimum time necessary (since I've been busy this year with a large client project), and get respectable results (marketing!). The TREC batch filtering task was the obvious choice, since this year it was purely and simply a text categorization task.

2. Learning Algorithm

Given the large training set available for batch filtering, choosing a supervised learning algorithm that would make effective use of this data was critical. The support vector machine approach (SVM) to training linear classifiers has outperformed competing approaches in a number of recent text categorization studies, particularly for categories with substantial numbers of positive training examples. SVMs require little or no feature selection, since they avoid overfitting by optimizing a margin-based criterion rather than one based on number of features. This minimizes the complexity of the software and processing. Finally, Thorsten Joachims has made publicly available an efficient implementation of SVMs, *SVM_Light* [Joachims 1999]:

http://www.joachims.org/svm_light/

SVM_Light allows training of both linear and, via kernels, nonlinear classifiers. I used linear classifiers in all cases. Indeed, I left all *SVM_Light* options that affect learning at their default values except *-j*, which controls the relative weight of positive and negative training examples in computing the margin-based loss criterion that SVM's optimize.

I modified *SVM_Light* to accept a comment before each example specifying a document ID, and to output during classification records containing score, predicted class, true class (if present in the test data), and document ID.

3. Tuning the Weighting of Positive and Negative Examples

My experiments focused on the relative weighting of positive and negative training examples. This was due to two problems I anticipated with using SVMs:

Problem 1. Past text categorization experiments have suggested that SVMs are less dominant over competing algorithms on categories with very few positive training examples. A plausible explanation is that the orientation of the learned hyperplane is being determined almost completely by the negative examples. In some machine learning tasks, the positive and negative classes are equally coherent, and a classifier fit to either will produce good effectiveness on the binary classification problem. This is rarely true in text categorization, however. The positive class is typically a coherent subset (e.g. "Retail Sales") of all possible documents, but the negative class is the less well-defined "everything else". Therefore, telling *SVM_Light* to pay more attention to positive examples for low frequency classes seemed like a good idea.

Problem 2. *SVM_Light* by default optimizes a margin-based loss measure which gives equal weight to positive and negative examples. It has been proven that optimizing this measure will tend to lead to low error rate; error rate also gives equal weight to positive and negative examples. However, the TREC batch filtering task used two effectiveness measures, T10SU and T10F, which give unequal weights to positive and negative examples and, moreover, were likely to require two different classifiers to optimize. (See the TREC-2001 filtering track report in this volume for more on these measures.) This again suggested paying attention to the weighting of positive and negative examples.

A typical approach to Problem 2 would be to train using *SVM_Light*'s usual criterion, producing a linear model with a threshold of 0. In a second phase, one would search for a new threshold value that optimizes the TREC effectiveness measure on the training set, while leaving the rest of the parameters unchanged [Lewis, et al 1996]. Zhang & Oles recently used this approach with SVMs [Zhang & Oles, 2001]. This approach assumes, however, that the optimal orientation of the hyperplane is the same for all effectiveness measures, something which is not at all clear. Further, it does nothing about Problem 1. I therefore took the opposite approach, which was to leave the threshold at 0, and try to force the fitting process for the other parameters to adapt to the TREC filtering measures.

Happily, *SVM_Light* has a parameter that controls the relative weight of positive and negative examples in its loss function. Since the T10SU measure corresponds to a weighting of positive examples to be 2 times more important than negative examples, an obvious approach would be to use that same relative weighting in training, at least in producing classifiers for the T10SU measure. I rejected that approach for two reasons:

1. While it has been proven that equally weighting positive and negative examples in SVM Light's loss function leads to high accuracy (i.e. high utility with an equal weighting on positive and negative examples), this result has not been shown to carry

through to unequal weightings of positive and negative examples. It seemed possible, indeed likely, that the scaling factors for the two measures would be different. In any case, as with most computational learning theory results, those for SVMs are too loose to constrain parameter settings tightly.

2. Something had to be done about the F-measure, which does not correspond to any simple relative weighting of positive to negative examples. (Indeed, the F-measure can't be optimized by any predetermined threshold [Lewis 1995], but I ignored this problem for TREC-2001.)

I therefore took a brute force approach. I did multiple training runs for each topic with relative weightings of positive to negative examples of 0.4, 0.6, 0.8, 0.9, 1.0, 2.0, 4.0, and 8.0.

This gave me 8 classifiers for each topic, from which I needed to choose a single classifier for each of the two effectiveness measures. I considered three methods:

Method 1. Evaluate on training data: Using each classifier to classify the training data, comparing those classifications with the true labels, computing the effectiveness on the appropriate measure, and picking the classifier with best effectiveness. The obvious problem with this approach is overfitting: the effectiveness estimates will be too optimistic. If the estimates were *systematically* too optimistic, then the choice of classifier would not be affected. However, that seemed too much to hope for, particularly with a nonlinear effectiveness measure such as T10F.

Method 2. Leave-one-out cross-validation: In cross-validation, one breaks the training data into *k* subsets. A classifier is trained on the union of *k-1* of the subsets, and evaluated on the *k*th subset. The process is repeated *k* times, using each of the subsets as the validation subset once. One then combines the results from the validation subsets to get an overall estimate of the effectiveness of the training procedure. The most extreme and most accurate version of cross-validation is *leave-one-out cross validation (LOOCV)*, i.e. doing *n*-fold cross validation when there are *n* training examples. Cross validation can be used to choose a parameter setting by making a cross-validated estimate of effectiveness at several values of the parameter, choosing the parameter value with highest estimated effectiveness, and then doing a final training run with all data using the chosen parameter setting. (Or one can train using all training data on all choices of parameter setting in advance, and then use cross-validation to pick the best of the already trained classifiers, as I did.)

Method 3. xi-alpha estimation: *SVM_Light* incorporates a highly efficient approximation to LOOCV called *xi-alpha estimation* [Joachims 2000].

It is important to stress that none of these three methods make any use of the test data.

Given the computational expense of Method 2, I first investigated Methods 1 and 3. I had high hopes for the xi-alpha estimate but found its predictions of effectiveness seemed both unrealistically pessimistic, and varied with the example weighting parameter in ways that seemed intuitively wrong. On the other hand, the estimates of Method 1 seemed unrealistically optimistic in many cases, as well as disagreeing strongly with the Method 3 estimates.

I therefore used the more expensive but accurate Method 2. SVM_Light includes support for LOOCV which, in the case of TREC 2001 batch filtering, meant 23,307-fold cross validation. Using this to choose among 8 values of a weighting parameter in theory meant training 8 × 23,307 classifiers, each on 23,306 examples, for each of 84 categories.

Fortunately, the properties of the SVM algorithm are such that many of the results of LOOCV folds can be predicted from a run on the full training data, without actually doing the training on the subset. *SVM_Light* incorporates an optimization that prunes away the folds that do not need to be explicitly run, and it meant that typically only a few hundred to a few thousand of the LOOCV folds were actually run for each setting, rather than 23,307 folds. In addition, I used a slightly aggressive version of pruning (the options -x 1 and -o 1) known to work well on text classification problems [Joachims, 2000] instead of the exactly correct version of pruning (options -x 1 and -o 2). Still, several weeks of computing time on a 700MHz PC were required to generate the results for 8 parameter settings and 84 categories.

A minor complexity was that *SVM_Light* output LOOCV and xi-alpha estimates of recall, precision, and error, but I really wanted estimates of T10F and T10U. I therefore wrote code to work backwards from the estimates that were printed (to only 4 digits of accuracy) to the actual contingency table entries, taking into account knowledge of the number of training examples and the number of positive examples for a topic. The code then computed estimates for T10F and T10U from the LOOCV contingency table entries.

These LOOCV-based effectiveness predictions were used to choose two sets of 84 classifiers, one set submitted for T10SU (run *DLewis01bfUa*) and one for T10F (run *DLewis01bfFa*).

I also used these two sets of classifiers to rank the test documents and submitted rankings of the top 1000 documents for each topic (runs *DLewis01rUa* and *DLewis01rFa*) for the routing evaluation.

4. Text Representation

SVMs, like most approaches in both machine learning and IR, require text to be reduced to vectors of numeric feature values.

Our text processing was minimal, consisting of downcasing the text, replacing numbers and punctuation with whitespace, and breaking the text into tokens at whitespace. No stemming was used. I discarded words on the SMART stoplist.

One innovation was motivated by the robustness of SVMs to large feature sets. It is common in IR to give additional prominence in a document representation to words that occur in the document title, for instance by counting them twice. It seems likely that this is a good strategy for some words but a poor one for others. To allow the learning algorithm to choose, I created two sets of features:

1. A set of binary features corresponding to the presence or absence of each word in the title.

2. A set of features for the total number of words in both the title and body of the document. Log TF weighting was applied to this feature set.

These two sets of features were combined into a single feature set used to represent documents. With respect to this feature set, there are linear models that give a variety of ranges of prominence to title words vs. words in the document body.

No IDF weighting or other corpus weighting was used. I did, however, apply cosine normalization to the feature vectors, as the default settings of *SVM_Light* are designed with this in mind.

5. Results

See the Appendix to these proceedings for the raw data. Comparing each of my runs to the other 18 submitted runs as evaluated by the measure I submitted that run for, I found:

- *DLewis01bfUa*, my T10SU run, had greater than or equal to median T10SU on 82 of 84 topics, and equaled the maximum T10SU score on 61 of 84 topics.

- *DLewis01bfFa*, my T10F run, had greater than or equal to median T10F on 81 of 84 topics, and equaled the maximum T10F score on 49 of 84 topics.

- *DLewis01rFa,* my T10F run as submitted for the routing evaluation, had greater than or equal to median SAP (simple average precision) on 83 of 84 topics, and equaled the maximum SAP score on 61 of 84 topics.

What I had intended to be quick-and-dirty runs using OPS (other people's software) were unusually dominant in a relatively mature TREC track. I believe the most important factors in this performance were:

1) The good fit of the SVM approach to text classification problems,

2) Exploring a range of relative weightings of positive and negative examples as a uniform way of addressing both the scaling and thresholding problems, and

3) Use of the computationally expensive but very accurate LOO approach for choosing a distinct relative weighting for each topic.

Future work will test the relative importance of these factors versus, for example, choice of text representation.

6. Afterword: High Frequency Topics for Dinner? Yum!

Prior to the submission of TREC Filtering runs, I made two dinner bets on the outcome of the routing evaluation. The outcome of one of these has been determined as of the writing of this paper. Here's the history:

1. Avi Arampatzis wrote (15-August-2001) to the TREC filtering mailing list, worrying that using only the top 1000 docs in the routing evaluation wouldn't be meaningful, because there were too many positive test documents.

2. As part of the discussion of Avi's obvservation, I wrote: "While I have not looked at the test data labels, I'll go out on a limb and predict that many groups will have have [sic] test set precision @ 1000 over 90% for a nontrivial number of topics. That suggests that any interesting differences between systems will only kick [sic] among documents well below rank 1000..."

4. Chris Buckley wrote "I'd be surprised with P @ 1000 of over 90% for any topics except those that are defined by a single keyword. That's a comment about reliability of the target categorization, not on system performance. Ie, the system may find 950 documents that should be in the category, but only 850 of those were actually assigned the category."

5. I wrote Chris off the list betting dinner that some system would get P @ 1000 of over 90% for some topic that was not defined by a single keyword. We discussed a bit how "single keyword" would defined and he accepted. (Basically, if Chris can write a single word query that gets P @ 1000 of 90% or more, the topic doesn't count.)

6. Separately, Paul Kantor wrote me and the list that "I will buy you a dinner if any system gets 90% @ 1000 for any topic." These were much looser terms than I'd already proposed to Chris, so I happily accepted.

7. Paul conceded on the list on September 6, 2001, after the preliminary results were released and several groups reported 90% @ 1000 results on 30 or so of the topics. I had a nice dinner with Paul at TREC 2001.

8. I am eagerly awaiting the result of Chris Buckley's single word queries to see who buys dinner, perhaps in Finland at SIGIR 2002.

Precision of 90% at 1000 documents seems to conflict with that fact that interindexer consistency rates are often 60% or lower [Cleverdon, 1991]. If two people can't agree more often than that, how can a machine do better? However, interindexer consistency is measuring agreement on an entire collection of documents. (In practice it is typically estimated by sampling techniques.) I made the above bets based on a suspicion that high scoring documents are different from collections as a whole. That is, if a well-trained statistical classifier is very confident a document belongs to a class, then it must be an "easy" document that almost any human indexer would assign to the class. The interindexer consistency, and thus the possible machine/indexer consistency, would be much higher than average. Since some routing topics in TREC 2001 were clearly going to have 10's of thousands of relevant documents, the top 1000 would consist of documents with very high scores.

In addition, at the time I made the bets I already knew that *SVM_Light*'s LOOCV estimates of precision on the *entire* training set were above 90% for some topics. Since LOOCV estimates are (almost) statistically unbiased, I was confident that results on the *entire* test set would be similar, and that on the top 1000 would be even better. As it turned out, some groups had precision of 100% on their top 1000 test documents for some topics.

On a serious note, it's clear that Avi Arampatzis' original worry was on the mark: evaluating routing runs on only the top 1000 documents meant that there was very little distinction among systems for high frequency topics. I will go further, and suggest that any ranking-oriented evaluation is of questionable interest when one has both substantial training data and thousands of relevant documents.

References

[Cleverdon 1991] C. Cleverdon. The Significance of the Cranfield Tests on Index Languages. In A. Bookstein, Y. Chiaramella, G. Salton, and V. V. Raghavan, editors, *SIGIR '91: Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 3-12, New York, 1991. Association for Computing Machinery.

[Joachims 1999] T. Joachims. Making Large-Scale SVM Learning Practical. In *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf and C. Burges and A. Smola (ed.), MIT Press, 1999.

[Joachims 2000] T. Joachims. Estimating the Generalization performance of an SVM Efficiently. *International Conference on Machine Learning (ICML)*, 2000.

[Lewis 1995] D. Lewis. Evaluating and optimizing autonomous text classification systems. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *SIGIR '95: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 246-254, New York, 1995. Association for Computing Machinery.

[Lewis, et al 1996] D. Lewis, R. Schapire, J. Callan, and R. Papka. Training algorithms for linear text classifiers. In Hans-Peter Frei, Donna Harman, Peter Schauble, and Ross Wilkinson, editors, *SIGIR '96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 298-306, Konstanz, 1996. Hartung-Gorre Verlag.

[Zhang and Oles, 2001] T. Zhang and F. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, v. 4, pp. 5--31, 2001.

Patterns of Potential Answer Expressions as Clues to the Right Answers

M. M. Soubbotin
InsightSoft-M, Moscow, Russia
MSoubbotin@insight.com.ru

Abstract

The core of our question-answering mechanism is searching for predefined patterns of textual expressions that may be interpreted as answers to certain types of questions. The presence of such patterns in analyzed answer-string candidates may provide evidence of the right answer. The answer-string candidates are created by cutting up relatively-large source documents passages containing the query terms or their synonyms/substitutes.
indicative patterns.
The specificity of our approach is:
- placing the use of indicative patterns in the core of the QA approach;
- aiming at the comprehensive and systematic use of such indicators;
- defining various structural types of the indicative patterns, including nontrivial and sophisticated ones;
- developing accessory techniques that ensure effective performance of the approach.
We believe that the use of indicative patterns for question answering can be considered as a special case of the more general approach to text information retrieval that contrasts with linguistics-oriented methodology.

Introduction

We decided to participate in the TREC-10 Question Answering track with a purpose to test certain specific features of the text processing technology we are developing in the framework of our CrossReader project. This technology is aimed at presenting to a user the needed information directly, i.e. instead of documents, or sources containing potentially relevant information. The query-relevant sentences or short passages are extracted from the processed documents and judiciously arranged; so, new full texts emerge that are focused precisely on the user's subject (Subbotin, 1993; Gilyarevskii, 1993; Perez, 2001).

The latest version of this technology - the TextRoller system - uses not only key words, but also positive and negative patterns for choosing and arranging text items. For the TREC-10 Question Answering task we have developed a variant of our basic technology that searches for candidate answers using key words (from the question text) and chooses the most probable answer using patterns. The participation at TREC-10 was a test for some basic mechanisms of our technology. Now, after this test was successfully passed, these mechanisms will be implemented in the new TextRoller versions.

Basic Features of the Applied Approach

It seems that many systems participating at TREC QA track represent the question (or its reformulations) as a set of entities and relations between them in order to compare these entities and relations with those of candidate answers texts; the answer candidate that correlates at the highest degree with the question text gets the highest score. By contrast, our QA system checks the answer candidates for the presence of certain predefined indicators (patterns) to which scores were assigned beforehand, i.e. independently of the question text analysis. Candidate snippets containing the highest-scored indicators are chosen as final answers.

It is obvious from the above that the applied approach does not require NLP or knowledge-based analysis of the question text. This text is considered as just a string consisting of various substrings. These are used, first of all, for composing queries helping to retrieve passages containing answer candidates. If present in candidate answers texts, they are considered as a condition of applicability of a given indicative pattern for a given question, but they do not influence the score of the pattern (as said above, it is predefined beforehand).

The efficiency of this approach depends on the quantity and diversification of predefined indicative patterns as well as on the recall of passages containing candidate answers.

We could not rely on the presence of predefined patterns in the texts of candidate answers for every question. If case of neither pattern was found, the system used the more common way to choose among candidate answers basing on lexical similarity between the question and an answer snippet. From 289 answer strings that were correct responses 193 did contain the patterns. Non-matching any patterns, but containing question (query) terms were 64. Other (containing minor indicators, such as capitalized words, or randomly selected) - 32.

To some extent, many QA-Track participants (at TRECs 8 and 9) had used what we call the indicative patterns. The specificity of our approach is:

- placing the use of indicative patterns in the core of the QA approach;
- aiming at the comprehensive and systematic use of such indicators;
- defining various structural types of the indicative patterns, including nontrivial and sophisticated ones;
- developing accessory techniques that ensure effective performance of the approach.

In (Sanda Harabagiu et al., 2000) the term "definition patterns" was introduced as "associated with questions that inquire about definitions". This kind of patterns was widely used by our QA system, although in many cases they were effective in combination with some additional indicators (see section "How patterns work"). It is also noteworthy that we did not confine the use of these patterns to questions inquiring about definitions. We assume, in general, that there should not be one-to-one correspondence between a given pattern and a question type. The same pattern can be applicable in answering many types of questions (getting a different score for each question type).

The Library of Indicative Patterns

The indicative patterns used by our QA system are sequences or combinations of certain string elements, such as letters, punctuation marks, spaces, tokens (such as "&", "%", or "$"), digits, and words/phrases that are accumulated in special lists (see Fig. 1).

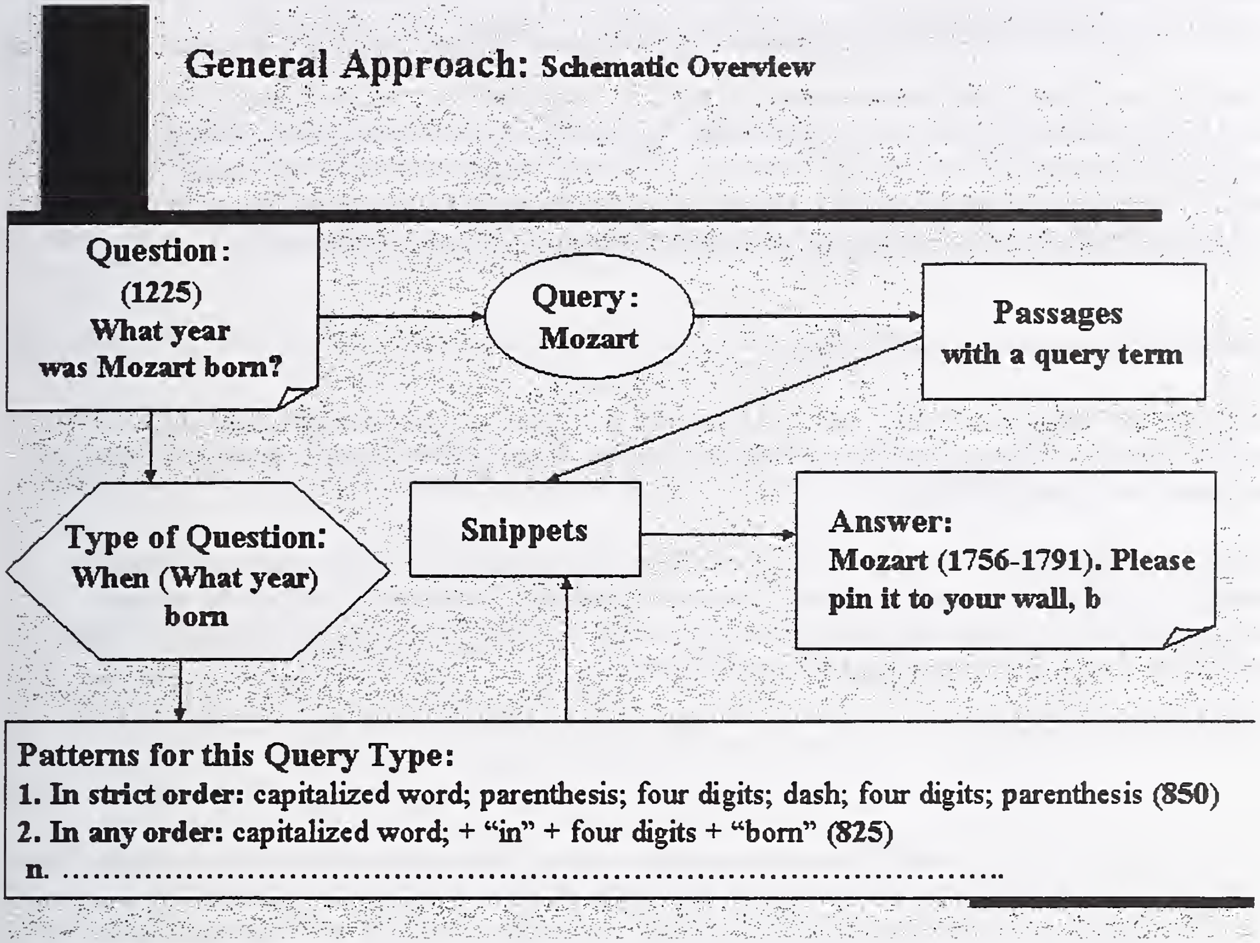

General Approach: Schematic Overview
Question: (1225) What year was Mozart born?
Query: Mozart
Passages with a query term
Type of Question: When (What year) born
Snippets
Answer: Mozart (1756-1791). Please pin it to your wall, b
Patterns for this Query Type:
1. In strict order: capitalized word; parenthesis; four digits; dash; four digits; parenthesis (850)
2. In any order: capitalized word; + "in" + four digits + "born" (825)
n. ....

Fig. 1. The general approach

The way we defined indicative patterns is totally heuristic and inductive.

At the initial stage the indicative patterns lists are accumulated basing on expressions that can be interpreted as answers to the questions of a definite type. For example: "Milan, Italy" present in any text passage can be considered (completely independently from the whole sense of the passage) as an answer to the question "Where is Milan". So, a pattern for the "Where" question type may be created: "city name; comma; country name". The string "Mozart (1756-1791)" contains answers to the questions about Mozart's birth and death dates, allowing construction of the pattern: "a person's name; parenthesis; four digits; dash; four digits; parenthesis ". We studied texts systematically with the purpose of identifying expressions that may serve as models for answer patterns. Some patterns components can be used for searching more complex structure patterns. The validity of a pattern for a given question type (and its score) can be tested in large text corpora.

The library of patterns can never be complete.

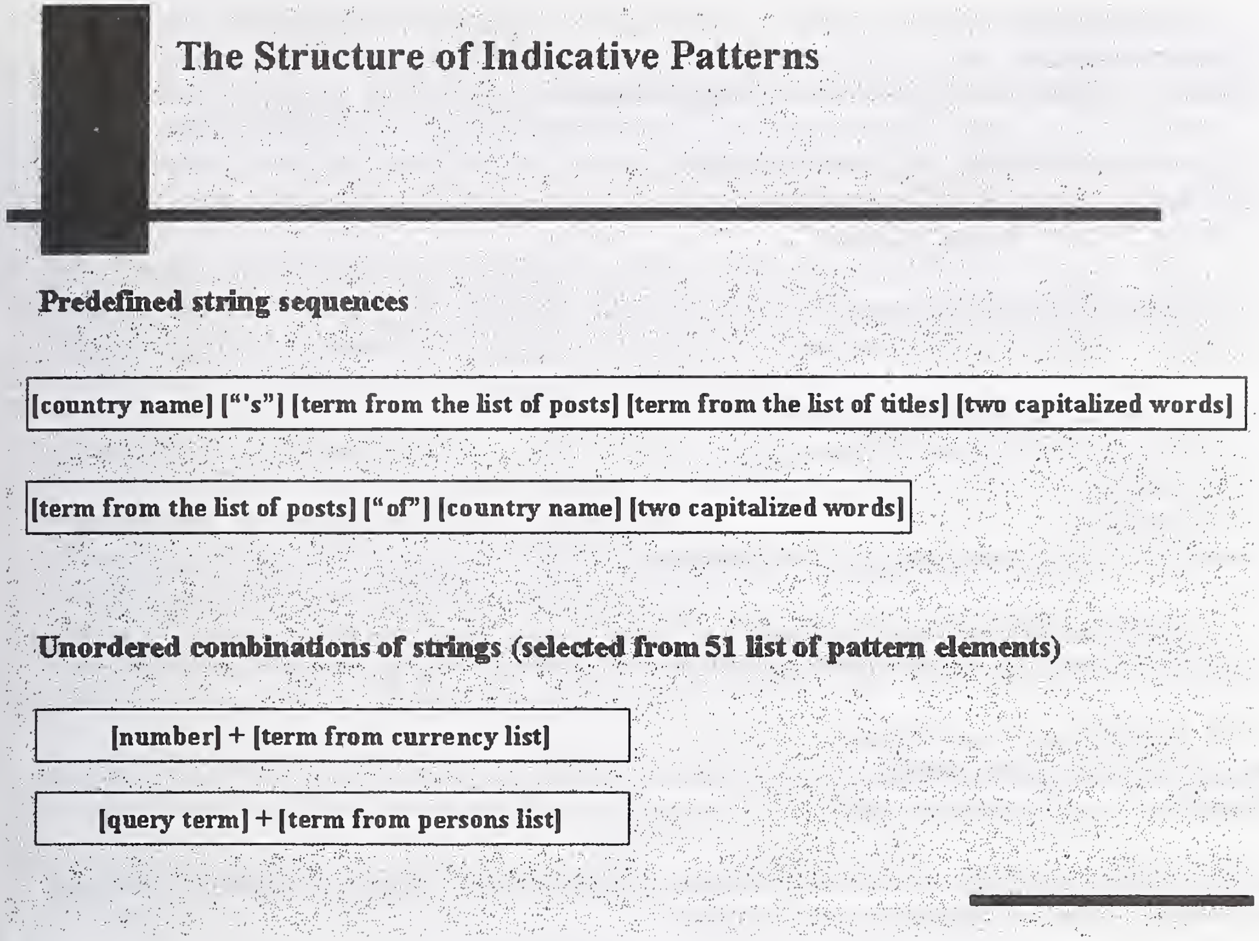
Identifying patterns while studying text corpora is a research field by itself, accumulating special knowledge on cooccurencies of text elements (characters, strings, as well as definite classes of characters and strings). So, it can be found that the string "Mr. " at a certain frequency level precedes one or two capitalized words, and the string "Jr." follows such words, etc.

Thus, we can accumulate the knowledge on "typical" combinations and correlations of strings that correspond to personal names, to a persons age, to locations, dates, activities, etc. This requires the use of sophisticated tools and special methods. This knowledge area can become important not only for QA, but also for other text retrieval tasks. For example, we use such methodology for extracting and ordering of sentences resulting in a coherent description of the requested subject.

The Structure of Indicative Patterns

A pattern may include a constant part and a variable part. The latter can be represented by a query term or even an unknown term (the answer word/phrase proper that occupies a definite position in the sequence of pattern elements).

We distinguish between two pattern categories: the first represents a complete structure while the second is a composite structure of specific pattern elements (see above). For TREC-10 we had prepared 51 lists of various patterns elements; for each question category 5 - 15 of such lists were applied for recognition of potential answers (see Fig. 2).



The Structure of Indicative Patterns

Predefined string sequences

[country name] ["'s"] [term from the list of posts] [term from the list of titles] [two capitalized words]

[term from the list of posts] ["of"] [country name] [two capitalized words]

Unordered combinations of strings (selected from 51 list of pattern elements)

[number] + [term from currency list]

[query term] + [term from persons list]

Fig. 2. Structure of the patterns

Usually, patterns with more sophisticated internal structure are more indicative of the answer. So, for answers to the question type "Who is the President (Prime Minister, etc.) of a given country" we found various combinations of elements that can be present in an answer expression: words that signify the name of a country, the post a person occupies, a proper name, a title, punctuation marks, etc. Let us denote countries by "a", posts by "b", proper names (first and last) by "w", titles (e.g. "His Excellency") by "e". The presence of combinations "abeww"; "ewwdb,a", "b,aeww" in an analyzed string indicates a correct answer to this question type with high probability.

The validity of certain simple structure patterns (e.g. the "definitions patterns") is dependent on the presence of additional positive and negative indicators (see below).

We distinguish between 6 basic definition patterns.

Below, these patterns are represented as sequences of their constituent elements. We will denote the primary query word present in the "snippet" with A, and the supposed "actual answer" with X; the pattern elements are divided by a semicolon. We consider two subtypes with the same structure, but where A and X occupy the inverse positions, as belonging to same pattern type.

1. <A; is/are;[a/an/the]; X>
<X; is/are;[a/an/the]; A>
Example: "Michigan's state flower is the apple blossom".

2. <A; comma; [a/an/the]; X; [comma/period]>
<X; comma; [a/an/the]; A; [comma/period]>
Example: "Moulin Rouge, a cabaret ".

3. <A; [comma]; or; X; [comma]>
Example: "shaman, or tribal magician,".

4. <A; [comma]; [also] called; X [comma]>
< X; [comma]; [also] called; A [comma]>
<X; is called; A>
<A; is called; X>
Example: "naturally occurring gas called methane".

5. <X, dash; A; [dash] A; dash; X; [dash]>
Example: "nepotism - hiring relatives for the better jobs".

6. <X; parenthesis-; A; parenthesis >
<A; parenthesis; X; parenthesis >
Example: "myopia (nearsightedness)".

As said above, these patterns were used not only for answering the "definition questions", but also for "Who-", "Where-", and other question types.

The expressions matching a pattern often show no structural similarity with the question text (see, for instance, the example in Fig 1). A lot of expressions in text corpora convey information that can be interpreted as answering a certain question without any special intention to do it (e.g. the standard beginning of agency news: "Milan, Italy..." answers the question "Where is Milan?".

How Patterns Work

Presence of certain patterns in the snippet-candidate serves as an almost guaranteed indication of the right answer (see Fig. 1). Their high score lets to choose the answer string with confidence. Lower score patterns cannot guarantee the correct response as they can be present in a number of candidate answer strings both correct and wrong. For some of such patterns we used additional indicators of validity. When there are several candidates with the same pattern, the system checks the text of candidate answers (and their surrounding) for presence of such additional indicators.

This is the case, in particular, for "definition patterns". Among the additional indicators for them there are such as absence of an article, presence of a stop-list word or other word lists.

Some recently-emerged text-processing techniques claim for using patterns while identifying relevant content. The best known is wrapper induction, an information-extraction technique that is considered an alternative to NLP-based methods (Kushmerick, 2000; Kushmerick, 1999; Adams, 2001). Wrappers demonstrate that extensive linguistic knowledge is not necessary for successful IE. For example, research on a collection of email conference announcements shows that speaker's names are often prefixed by "Who" and many names begin with the title "Dr." (Fraitag, 2000).

However, wrapper induction in its present-day form is resource specific, it extracts information from particular Web sites. It uses specific features related to the document formats rather than the ways information is commonly presented in written texts.

Overview of the Process Flow

Preconditions for effective use of the method are:
- Detailed categorization of question types (for example, we distinguish between nine "Who"-question types ("Who-Post"; "Who-Author", etc);
- The great variety of patterns for each type (for "Who-Author"-type we have 23 patterns);
- A sufficiently large number of candidate answers to each question (usually we get several hundreds or thousands of candidate snippets).

Multiple overlapping answer-string candidates ("snippets") are created by cutting source documents passages containing the query terms or their substitutes (see Fig. 3).

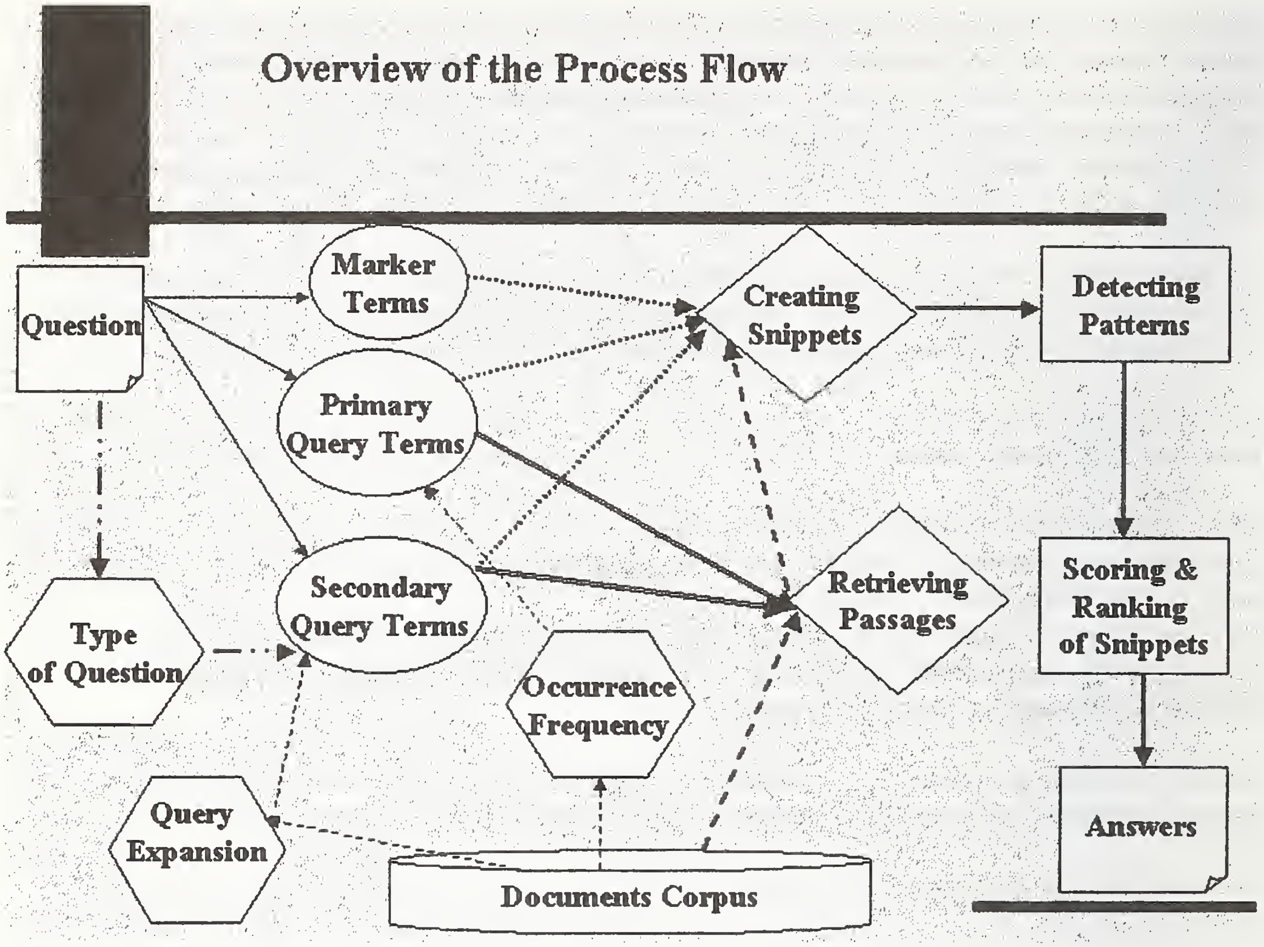


Fig. 3. Overview of the process flow

Using of specific question words (in contrast to common words) as query terms ensures in most cases that the question subject is addressed in the source passages.

In the literature we find approaches attempting to distinguish between the main (primary) and additional (secondary) query words. In (Sneiders, 1998) this distinction is discussed as applied to searching for answers to FAQs, where the answers are represented as sentences. Primary keywords are the words that convey the essence of the sentence. They cannot be ignored. Secondary keywords are the less-relevant words for a particular sentence. They help to convey the meaning of the sentence but can be omitted without changing the essence of the meaning.

We accept this distinction by assuming that the primary terms are question-specific words and are almost inevitably present in the passage that treats the same subject as the question. We use certain criteria of the specificity, including the minimal occurrence in the documents corpus (for example, "mortarboard" in the question "Where on the body is a mortarboard worn").

In some question categories, primary query words do not convey the question subject completely, requiring secondary searching terms. Such terms are, for example, the words signifying a certain post in questions of the type "Who is (the) X of (the) Y?", where X is a post, and Y is the name of a country, company, organization, etc.

For some question types the secondary query terms should be supplemented by their related words. For this purpose we use a query-expansion technique. Our expansion module extracts query-related terms not from the full documents, but from the short relevant text passages.

The retrieved passages are cut into 50-byte snippets. They are cut around the query words, as well as around other question words that have not served as query terms (these are denoted as "markers" in Fig. 1).

All the snippets are analyzed to identify patterns that are indicative of a potential answer (as described above).

The Results and the Perspectives of the Approach

Our mean reciprocal rank (strict): 0.676; mean reciprocal rank (lenient): 0.686. System was confident for 372/492 (75 %) of the questions. Of those, 289/372 (77 %) were correct responses. Two thirds of correct answer strings were obtained using patterns thus proving the feasibility of the applied approach.

We believe that the use of indicative patterns for question answering can be considered as a special case of the more-general approach to text information retrieval that contrasts with the linguistics-oriented methodology.

Generally, text documents contain information that is included not intentionally, but due to its indirect interconnections to what the author directly conveys. This implicit information can be addressed systematically by a set of patterns. We are conducting investigations that will allow us to develop appropriate tools for this.

Aiming at the practical implementation of indicative patterns approach, we are currently developing advanced versions of our TextRoller technology that uses both query terms and patterns while assembling new "full texts" from appropriate passages of the processed documents. The patterns are used not only for choosing passages, but also for ensuring their judicious arrangement, as well as domain specificity and readability of the constructed text.

Bibliography

Adams Katherine C.. The Web as a Database. New Extraction Technologies & Content Management. Online, March/April 2001, pp. 28 - 32

Fraitag, Dayne. Two Approaches to Learning for Information Extraction. Talk at UC, San Diego (San Diego, CA) on October 16, 2000

Gilyarevskii R., M. Subbotin. Russian Experience in hypertext: automatic compiling of coherent texts. Journal of the American Society for Information Science, 1993, v.44, 4, pp. 185-193

Harabagiu Sanda, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Girju, Vasile Rus and Paul Morarescu. Falcon : Boosting Knowledge for Answer Engines. Proceedings of the Text Retrieval Conference (TREC-9), 2000

Kushmerick, Nicolas. Cleaning the Web. IEEE Intelligent Systems, vol. 14, No. 2 (1999): http://www.cs.ucd.ie/staff/nick/home/research/download/kushmerick-ieeeis99.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

Perez, Ernest. Finding Needle in the Textstacks. Online, September/October 2001

Sneiders, E.. [The Development of an FAQ Answering System].
Information Systems in the WWW Environment. IFIP TC8/WG8.1 Working Conference, 15-17 July 1998, Beijing, China. Published by Chapman & Hall on behalf of IFIP, pp. 298-319

Subbotin M., D. Subbotin. INTELTEXT: Producing Coherent Linear Texts While Navigating in Large Non-Hierarchical Hypertexts. Lecture Notes in Computer Science, N 753, Springer-Verlag, 1993, pp. 281-289

Acknowledgments: The authors are thankful to Mr. Daniel Kamman for help with preparation of the English version of this article.

Mercure and MercureFiltre applied for Web and Filtering tasks at TREC-10

M. Boughanem, C. Chrisment, M. Tmar

IRIT-SIG
Campus Univ. Toulouse III
118, Route de Narbonne
F-31062 Toulouse Cedex 4
Email: trec@irit.fr

1 Summary

The tests performed for TREC-10 focus on the Filtering (adaptive, batch and routing) tracks and web tracks. The runs are based on Mercure system for web, routing and batch tracks, and MercureFiltre for adaptive track.

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 [2] [3].

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 two layers are symmetric and their weights are based on the $tf - idf$ measure inspired from the OKAPI [5] and SMART term weightings.

- the query-term (at stage s) links are weighted as follows:

$$q_{ui}^{(s)} = \begin{cases} \frac{nq_u * qtf_{ui}}{nq_u - qtf_{ui}} \; if \; (nq_u > qtf_{ui}) \\ qtf_{ui} \; otherwise \end{cases} \qquad (1)$$

Notations:
$q_{ui}^{(s)}$: 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 q_u,
nq_u: query length (number of terms) of q_u,

- the term-document link weights are expressed by:

$$d_{ij} = \frac{tf_{ij} * \left(h_1 + h_2 * log\left(\frac{N}{n_i}\right)\right)}{h_3 + h_4 * \frac{dl_j}{\Delta d} + h_5 * tf_{ij}} \tag{2}$$

Notations:
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,
Δ_d: average document length.

The query evaluation is based on spreading activation. Each node computes an input and spreads an output signal. The query modification is based on relevance back-propagation. It consists in spreading backward the document relevance from the document layer to the query layer [2].

2.1 Query evaluation

A query is evaluated using the spreading activation process described as follows:

1. The query u is the input of the network. Each node from the term layer computes an input value from this initial query: $In\,(t_i) = q_{ui}^{(s)}$ and then an activation value: $Out\,(t_i) = g\,(In\,(t_i))$ where g is the identity function.

2. These signals are propagated forwards through the network from the term layer to the document layer. Each document node computes an input: $In\,(d_j) = \sum_{i=1}^{T} Out\,(t_i) * d_{ij}$ and then an activation, $Out\,(d_j) = g\,(In\,(d_j))$.

   The set of retrieved documents, $Output_u\,(Out\,(d_1)\,, Out\,(d_2)\,, \ldots, Out\,(d_N))$ is then ranked in a decreasing order of the activation value.

2.1.1 Query modification based on relevance back-propagation

The top retrieved documents are judged and a *relevance value* corresponding to the user preference is assigned to each document (positive for relevant documents, negative for non-relevant documents and null for non-judged documents). These values are used to compute the *DesiredOutput* vector.

$$DesiredOutput = (rel_1, rel_2, \ldots, rel_N) \begin{cases} rel_j = \frac{Coef\_rel}{Nb\_rel}\, if\ D_j\ is\ relevant \\ rel_j = \frac{Coef\_Nrel}{Nb\_Nrel} otherwise \end{cases} \tag{3}$$

1. This output is considered as an "input" in the back-spreading process and is presented to the document layer. Each document node computes an input value, $In(d_j) = rel_j$ and then an activation signal, $Out(d_j) = g(In(d_j))$.
2. This activation is back-spread to the term layer. Each term node computes an input value, $In(t_i) = \sum_{j=1}^{N} (d_{ji} * Out(d_j))$ and then an output signal, $Out(t_i) = g(In(t_i))$.
3. Finally, the new query-term links corresponding to the new query are computed as follows: $Q_u^{(s+1)} = \left(q_{u1}^{(s+1)}, q_{u2}^{(s+1)}, \ldots, q_{uT}^{(s+1)}\right)$ with $q_{ui}^{(s+1)} = M_a * q_{ui}^{(s)} + M_b * Out(t_i)$

   Notations:
   T: the total number of indexing terms,
   N: the total number of documents,
   $q_{ui}^{(s)}$: the weight of the term t_i in the query u at the stage s,
   t_i: the term t_i,
   d_j: the document d_j,
   d_{ij}: the weight of the link between the term t_i and the document d_j,
   $doclen_j$: document length in words (without stop words),
   tf_{ij}: the term frequency of t_i in the document d_j,
   n_i: the number of documents containing term t_i,
   qtf: query term frequency,
   nq: query length (number of terms),
   M_a and M_b: tuned and determined by a series of experiments and set to $M_a = 2$ and $M_b = 0.75$,
   $Coef\_rel$ ($Coef\_Nrel$): user preference positive value for relevant document and negative value for non relevant document.

3 Batch and Routing Experiments

The batch and routing experiments were performed using Mercure system. The profiles were learned using the same learning algorithm as before: the relevance back-propagation. The relevance value assigned to each document was used as user judgement. It corresponds to $Coef\_rel$ in the relevance back-propagation algorithm.
The filtering algorithm starts with an initial query, built from all the topic parts, and its relevance judgements corresponding to all documents with items down to 26150. A pool of queries based on the learning method was then selected. All the remaining documents (with items up to and including 26150) were used as test data.

3.1 Batch Filtering

The profiles in the batch task are learned using the relevance back-propagation method.
The TREC standard output file of each query was analyzed to build an output file containing:

$$< topic >< func >< value >< thresh >< rank >< prec >< recall >< method >$$

As it has been done in [4]. The document activation weights which maximizes the utility function were found and selected as thresholds. Then the queries corresponding to these thresholds were selected and tested on a set of test documents. The documents weighted by values higher than the threshold were selected for the corresponding query.

We build profiles using relevance back-propagation on the training dataset, then we apply them on the test dataset.

The following algorithm is used:

```
For each profile $P$
  1. evaluate $P$ on the training dataset
  2. select top 10000, let $result_0$ be the obtained document ranked list
  3. $i = 1$
  4. repeat
       (a) build a new profile $P_i$ by relevance back-propagation
       (b) evaluate $P_i$ on the training dataset
       (c) select top 10000, let $result_i$ be the obtained document ranked list
       (d) inc $i$
     until $i = max\_iteration$
  5. for each $r \in \{1 \ldots 10000\}$, $i \in \{0, 1, \ldots max\_iteration\}$
       (a) $result_{ir}$ contains top $r$ documents from $result_i$
       (b) evaluate $result_{ir}$ using $T9U$ utility
  6. select profile $P_i$ such as $\exists r \in \{0, 1 \ldots 10000\}$ where $result_{ir}$ gives the best (max) $T9U$
  7. apply $P_i$ on the test dataset, let $test\_result_i$ be the obtained document ranked list
  8. select documents in $test\_result_i$ having their $rsv$ at least equal to the $rsv$ of the $r^{th}$
     document
  9. submit this list
```

In the experiments, we carried out relevance twice. We found that this number of iterations was enough to learn the profile.

We computed the utility on the top 10000 documents only as this set is likely to contain most of the relevant documents.

3.1.1 Batch filtering results

Table 1 lists the comparative batch results.

TREC batch filtering				
Evaluation measure	$= max$	$\geq median$	$< median$	Avg
F-Beta	1	24	59	0.392
SetPrecision	10	36	38	0.631
SetRecall	1	20	63	0.204
T10SU	0	28	56	0.181

Table 1: Comparative batch filtering results

3.2 Routing track

3.2.1 Description of the experiment

We experiment routing using a similar method then in the batch filtering track, the queries having the best average precision in the training dataset were selected as routing queries, and applied on the test documents.

3.2.2 Routing results

Table 2 shows the routing results at average uninterpolated precision.

TREC Routing				
$= max$	$\geq median$	$< median$	$AvgP$	Precision at 1000
2	47	35	0.092	0.5594

Table 2: Comparative routing results at average uninterpolated precision

The average number of relevant documents per profile is largely great than 1000. The evaluation function used this year is the average uninterpolated precision, it consists on computing the precision at each relevant document at its position in the ranked list, adding these numbers up and divide by the total number of relevant documents. Relevant documents which do not appear in the top 1000 receive a precision score of zero. Non relevant documents appearing in the top of the ranked have a large influence on the average uninterpolated precision.

4 Adaptive track

We experiment this year an adaptive filtering process [1] using MercureFiltre. MercureFiltre is inspired from Mercure, It includes three elements: the profile, the document and the dissemination threshold. The profile and the document are represented by a set of weighted terms. The adaptation concerns the profile and the dissemination threshold.

4.1 System initialisation

The user profile is represented as follows:

$$p^{(0)} = \left(\left(tp_1, w_1^{(0)}\right), \left(tp_2, w_2^{(0)}\right) \ldots \left(tp_n, w_n^{(0)}\right)\right) \quad (4)$$

where tp_i is a term and $w_i^{(0)}$ is its weight in the initial profile (at $t = 0$), the term-profile weight is noted $w_i^{(t)}$ where t represents the instant when the system makes the last profile update. Initially, the term-profile weight is computed as follows:

$$w_i^{(0)} = \frac{tfp_i}{max_j\,(tfp_j)} \quad (5)$$

The formula 5 seems to be abusively simplistic, but at the beginning of the filtering process, no information is known but a set of terms and their occurrences in the initial user profile. However, this weight will be adjusted by learning.

4.1.1 Filtering incoming documents

Each term belonging to a document arriving at time t is weighted as follows:

$$d_i^{(t)} = \frac{tf_i^{(t)}}{h_3 + h_4 * \frac{dl^{(t)}}{\Delta l^{(t)}} + tf_i^{(t)}} * log\left(\frac{N^{(t)}}{n_i^{(t)}} + 1\right) \quad (6)$$

Notations:
$tf_i^{(t)}$: appearance frequency of the term 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 of $d^{(t)}$ (number of 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

The weighting formula 6 is a form of Okapi term-profile weight used in Mercure.
$N^{(t)}$, $n_i^{(t)}$ and $\Delta l^{(t)}$ are collection based parameters that are computed on the cumulative documents filtered at t.

A relevance value noted $rsv\left(d^{(t)}, p^{(t)}\right)$ is then computed corresponding to the document and the profile, as follows:

$$rsv\left(d^{(t)}, p^{(t)}\right) = \sum_{t_i \in d^{(t)}, tp_j \in p^{(t)} \text{ and } t_i = tp_j} d_i^{(t)} * w_j^{(t)} \quad (7)$$

The binary decision rule used for document filtering is the following:

$$\begin{cases} if\ rsv\left(d^{(t)}, p^{(t)}\right) \geq threshold^{(t)}\ accept\ the\ document \\ otherwise\ reject\ the\ document \end{cases}$$

4.1.2 Adaptive learning

The learning process is processed at each selected relevant document. Learning consists on adapting the user representation: update term-profile weight, add new terms and remove some non significant terms.
We assume that an interesting term must appear frequently in relevant documents, and appear a little in non relevant documents. When a relevant document is selected, each term-profile weight increases the more:

- it appears frequently in this document
- it appears in many relevant documents ($\frac{r_i^{(t)}}{R^{(t)}}$ is near 1)
- it appears in a few non relevant documents ($\frac{s_i^{(t)}}{N^{(t)}}$ is near 0)

Notations:
$r_i^{(t)}$: the number of relevant documents containing the term t_i until the time t
$R^{(t)}$: the total number of relevant documents until the time t
$s_i^{(t)}$ the number of non relevant documents containing the term t_i until the time t.

The learning process we adopted is a kind of relevance reinforcement process. The problem is assimilated to a linear equation resolution. The equation to resolve is to force obtaining a *rsv* value β to a relevant document. Each solution of this system is a set of weights affected to the document terms. The constraints are that the weight of each term in the profile divided by its importance according to the same profile is constant. The system is then the following:

$$\begin{cases} \sum_{t_i \in d^{(t)}, tp_i \in p^{(t)} \ and \ t_i = tp_j} d_i^{(t)} * w_j^{(t)} = \beta \\ \frac{w_i^{(t)}}{f\left(d_i^{(t)}, r_i^{(t)}, s_i^{(t)}\right)} = \frac{w_j^{(t)}}{f\left(d_j^{(t)}, r_j^{(t)}, s_j^{(t)}\right)} \ \forall \left(t_i, t_j\right) \in {d^{(t)}}^2 \end{cases} \tag{8}$$

However, the *rsv* depends on the document length. Let α be the *rsv* value wished for a relevant document with length dl_m, α and dl_m are constants. When $dl^{(t)}$ increases β increases but $\frac{\beta}{dl^{(t)}}$ decreases, so β is logistically proportional to the document length:

$$\beta = \frac{\alpha}{log\left(dl_m\right)} * log\left(dl^{(t)}\right) \tag{9}$$

For these experiments, $\alpha = 20$ and $dl_m = 100$.
The solution of the equation system 8 is a set of "provisional" weights, for each term appearing in the document, the provisional weight solution of the system 8 is the following:

$$\forall i, \ pw_i^{(t)} = \frac{\beta f\left(d_i^{(t)}, r_i^{(t)}, s_i^{(t)}\right)}{\sum_j f\left(d_j^{(t)}, r_j^{(t)}, s_j^{(t)}\right) * d_j^{(t)}} \tag{10}$$

The function f is proportional to the term importance. The function f used for these experiments is the following:

$$\forall i, f\left(d_i^{(t)}, r_i^{(t)}, s_i^{(t)}\right) = d_i^{(t)} * \frac{exp\left(\gamma * \frac{r_i^{(t)}}{R^{(t)}}\right) * \left(1 - exp\left(\gamma\left(\frac{s_i^{(t)}}{S^{(t)}} - 1\right)\right)\right)}{exp\left(\gamma\right)} \tag{11}$$

where $R^{(t)}$ is the number of relevant documents and $S^{(t)}$ is the number of non relevant documents until the time t. *gamma* is set to 3.

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:

$$w_i^{(t+1)} = w_i^{(t)} + log\left(1 + pw_i^{(t)}\right) \tag{12}$$

We add 1 to $pw_i^{(t)}$ to avoid adding negative value to the term-profile weight.

4.1.3 Threshold setting and updating

For each topic, the dissemination threshold should be set and updated [6]. To set the intial threshold, we consider for this experiment that first 40 incoming documents are non relevant, the threshold is initialized with the average of the top 10 rsv values computed for these documents.

Threshold updating is made periodically at each 10 selected documents as follows:

$$\begin{align}
Threshold^{(t+1)} &= Threshold^{(t)} * A_1 * A_2 * A_3 \tag{13} \\
A_1 &= -exp\left(-\alpha_c \frac{S^{(t+1)}}{N^{(t+1)}}\right) + 2 \tag{14} \\
A_2 &= \frac{1}{\beta_c \frac{R^{(t+1)}}{R^{(t+1)} + S^{(t+1)}}} + 1 \tag{15} \\
A_3 &= exp\left(-\frac{NT^{(t+1)}}{\gamma_c}\right) + 1 \tag{16}
\end{align}$$

Notations:
α_c, β_c and γ_c are corrector parameters,
$R^{(t+1)}$ is the number of selected relevant documents between instants t and $t + 1$,
$S^{(t+1)}$ is the number of selected non relevant documents between instants t and $t + 1$,
$N^{(t+1)}$ is the number of incoming documents between instants t and $t + 1$,
$NT^{(t+1)}$ is the number of all incoming documents until instant $t + 1$.

A_1 enables to increase the threshold if the proportion of selected non relevant documents ($\frac{S}{N}$) is high, A_2 enables to decrease the threshold if the proportion of relevant rejected documents is assumed to be relatively high ($\frac{R}{R+S}$), A_3 enables to increase the threshold less quikly when the number of evaluated documents (N) is assumed to be enough for learning.

4.1.4 Adaptive results

Table 3 lists the adaptive results.

TREC adaptive filtering	
Evaluation	Avg
F-Beta	0.0192
SetPrecision	0.3865
SetRcall	0.0196
T10SU	0.0297

Table 3: Adaptive filtering results

Unfortunately, we have had no time to submit our results.

5 Web Track Experiment

This year experiment is based on Mercure sample search. Two runs were submitted to be integrated on the pool:

1. Merxtd: simple search using the title and the descirption fields

2. Merxt: simple search using the title field

Table 4 shows the comparative web results:

Type	Run	average precision	=max	≥ median	< median
Merxtd	title + description simple search	0.1729	2	18	30
Merxt	title simple search	0.1438	0	26	24

Table 4: Web results - 50 queries

References

[1] N. J. Belkin, W. B. Croft, *Information retrieval and information filtering: two sides of the same coin?*, Communications of the ACM 35(12), pages 29-38, 1992.

[2] M. Boughanem, C. Chrisment & C. Soule-Dupuy, *Query modification based on relevance back-propagation in Adhoc environment*, Information Processing and Management, 35(1999), pages 121-139, April 1999.

[3] M. Boughanem, T. Dkaki, J. Mothe & C. Soule-Dupuy, *Mercure at TREC-7*, Proceedings of the 7th International Conference on Text REtrieval TREC-7, E. M. Voorhees and D.K. Harman (Ed.), NIST SP 500-242, pages 413-418, November 1998.

[4] S. Walker, S. E. Robertson, M. Boughanem, G. J. F. Jones, K. Sparck Jones, *Okapi at TREC-6 automatic and ad hoc, VLC, routing, filtering and QSDR*, Proceedings of the 6th International Conference on Text REtrieval TREC-6, D.K. Harman (Ed.), NIST SP 500-240, pages 125-136, November 1997.

[5] S. E. Robertson, S. Walker *Okapi/Keenbow at TREC-8* In Proceedings of the TREC-8 Conference, National Institute of Standards and Technology, pages 151-161, November 2000.

[6] S. E. Robertson, S. Walker, *Threshold setting in adaptive filtering*, Journal of documentation 56, pages 312-331, 2000.

Multilingual Question/Answering: the DIOGENE System

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 the DIOGENE question/answering system developed at ITC-Irst. The system is based on a rather standard architecture which includes three components for question processing, search and answer extraction. Linguistic processing strongly relies on MULTIWORDNET, an extended version of the English WORDNET. The system has been designed to address two promising directions: multilingual question/answering and question/answering on the Web. The results obtained in the TREC-10 main task will be presented and discussed.

1 Introduction

In the last few years, under the promotion of the TREC-8 (Voorhees and Tice 1999) and TREC-9 (Voorhees 2000) competitions, there has been a large interest in developing QA systems. Being our first participation at TREC-QA we adopted, for the DIOGENE system, a rather common architecture with three basic components: a *question processing* component, based on several linguistic processors and resources including WORDNET; a *search component*, based on information retrieval techniques; and an *answer processing* component, which exploits the similarity between the question and the documents to identify the correct answer.

In order to identify which modules are appropriated both previous experiences in QA and modules already available at ITC-Irst have been considered. Among the proposals that we have found of interest for the linguistic analysis, we have considered the taxonomy of question types implemented in LASSO (Moldovan et al. 1999) for identifying both the question type and the answer type; the term extraction capabilities developed for the QALC system (Ferret et al. 2000); and the use of lexical knowledge contained in WORDNET for the retrieval of semantic relations, as proposed in (Moldovan et al. 2000).

In addition, during the design phase of the system, two directions for future developments have been taken into consideration: *multilingual QA* and *QA on the Web*. Multilinguality is a crucial aspect when the language of the search question and the language of the text collection are different. We experimented with an Italian/English scenario, where the question can be posed either in English or Italian, the search is performed either in English or Italian, and the answer is given in the language of the question. In the paper we will discuss the solutions adopted for multilinguality, even if they are currently out of the scope of the TREC competition.

As for QA on the Web, this perspective raises a number of specific issues, the most evident being that the implication of an answer with respect to its question for a Web user is generally weaker than for controlled text collections, where human judges apply rigid tests. A reason for this is that the relevance of a Web document relies on several factors. For instance, a retrieved document could not include the answer in itself, but could nevertheless provide links to other documents useful to find the answer. Elements that provide implicit knowledge are the document structure, hypertextual links, multimedia co-reference, and generally, a strong use of contextual information.

This paper is structured as follows. Section 2 introduces the overall architecture of the system. Section 3 addresses the linguistic analysis of the question, including word sense disambiguation, answer type identification and query expansion. Section 4 describes the searching modalities. Section 5 presents the answer extraction component, including paragraph filtering, named entities recognition and answer identification. Section 6 illustrates the TREC results, with a discussion on strengths and weaknesses of DIOGENE.

2 Architecture

DIOGENE relies on a rather standard architecture based on three basic components (see Figure 1): a *question processing* component, a *search component* and an *answer processing* component.

The ***question processing*** component is in charge of the linguistic analysis of input questions that can be formulated either in English or in Italian. At this step of the process, we confront the multilinguality problem using language specific resources. The analysis is performed sequentially by the following modules.

- ***Tokenization and pos tagging***. First the question is tokenized and words are disambiguated with their lexical category by means of statistical part of speech taggers. The Treetagger developed at the University of Stuttgart (Schmid, 1994) is in charge of the disambiguation of English words. Italian questions are processed by a part of speech tagger developed at ITC-Irst.
- ***Multiwords recognition***. About five thousand multiwords (i.e. collocations, compounds and complex terms) have been automatically extracted from different resources and are recognized by pattern matching rules. English multiwords have been extracted from WORDNET (Fellbaum, 1998); Italian multiwords have been extracted from a monolingual Italian dictionary (Disc, 1997)
- ***Word sense disambiguation***. This module, described in Section 3.1, disambiguates words in the query with respect to their senses. MULTIWORDNET (Pianta et al., 2002) was adopted as a sense repository. Word sense disambiguation is crucial for providing reasonable keyword expansions and correct translations between the two languages.
- ***Answer type identification***. The answer type for a question represents the entity to be searched as answer. This module relies on a taxonomy of answer types and a pattern matching rule system, both described in Section 3.2.

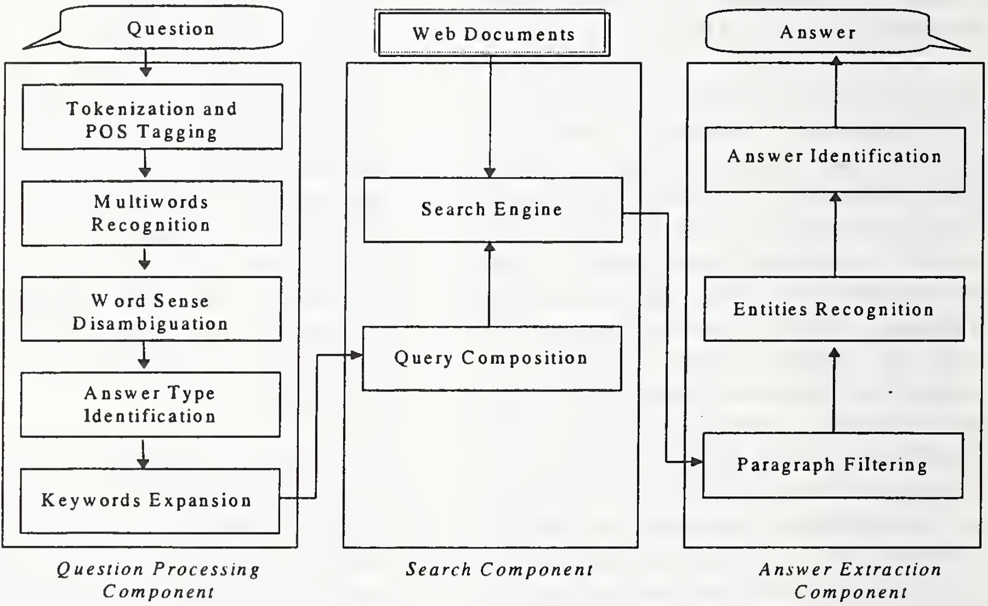


Figure 1. System Architecture.

- ***Keywords expansion***. Two kinds of expansions, described in Section 3.3, are carried out: word synonyms and morphological derivations. Also in this case, multilinguality is guaranteed by the use of MULTIWORDNET.

The ***search*** component first composes the question keywords and their lexical expansions and then performs the document retrieval. For the participation at TREC-10 the ZPrise search engine, described in section 4, has been used.

The ***answer extraction*** component implements a paragraph filtering module that extracts text paragraphs from the top scored retrieved documents. This is done by maximizing the number of keywords and expansions produced in the question processing phase within a window of a fixed length of words. The output is composed by the text paragraphs that should contain the answer to the question. Then, a *named entities recognition* module identifies the entities in the text paragraphs corresponding to the answer type category. We are using an adaptation of *Learning*-PINOCCHIO (Ciravegna, 2000), which makes use of learning algorithms to recognize named entities, such as persons, organizations, locations, measures and dates. Finally, the *answer identification* module highlights the portion of text containing the answer to the question which is then presented to the user.

3 Linguistic Expansions

3.1 Semantic Disambiguation

The identification of the correct sense of a word in a question is necessary to add either synonyms or translations for that word without the risk of introducing disturbing elements in the search query. There are two crucial questions to address: first, a repository of word senses has to be identified; second, it is important to develop a disambiguation technique able to cope with the specificity of questions, particularly with the availability of a limited context (i.e. few words).

As for sense repository we have adopted MULTIWORDNET (Pianta et *al.* 2002), a multilingual lexical database including information about English and Italian words. MULTIWORDNET is an extension of English WORDNET (Fellbaum, 1998), a semantic network of English words grouped into synonym sets called synsets. Words and synsets are linked by means of various relations, distinguished into semantic relations, which link concepts, and lexical relations, which link individual words. The main lexical relations represented in WORDNET are synonymy and antonymy, while hyponymy, hypernymy, meronymy, entailment and conceptual opposition are the main semantic relations that link the synsets. MULTIWORDNET has been developed keeping as much as possible of the semantic relations available in the English WORDNET: Italian synsets have been created in correspondence with English synsets, importing lexical and semantic relations from the corresponding English synsets. The Italian part of MULTIWORDNET currently covers about 40,000 lemmas, completely aligned with the English WORDNET 1.6 (i.e. with correspondences to English senses).

As far as word disambiguation is concerned we have applied Word Domain Disambiguation (WDD), a technique already experimented for the disambiguation of short news (Magnini, Strapparava 2000), and further extended by adding domain frequency information. Word Domain Disambiguation is a variant of Word Sense Disambiguation (WSD) where for each word in a text a *domain* label, (among those allowed by the word) has to be chosen instead of a *sense* label. Domain labels, such as MEDICINE and ARCHITECTURE, provide a natural way to establish semantic relations among word senses, grouping them into homogeneous clusters. In MULTIWORDNET the synsets have been annotated with one or more domain labels selected from a set of about two hundred labels hierarchically organized (see (Magnini, Cavaglià, 2000) for the annotation methodology and for the evaluation of the resource).

The WDD algorithm works in two steps. First, for each content word in the query and for each sense of the word, the corresponding domain labels in MULTIWORDNET are collected with a score determined by the frequency of the label among the senses of the word. Let us consider as example an ambiguous query from the TREC corpus. In "*What is the brightest star visible from Earth*?" the situation after the first step of the WDD algorithm is represented in Figure 2.

star	star#1: celestial body	Astronomy
	star#2: an actor who play	Art
bright	bright #1: brilliant shining	Physics
	bright #2: popular glorious	Factotum
	bright #3: promising auspicious	Factotum
visible	visible #1: conspicuous obvious	Physics
	visible #2: visible seeable	Astronomy
earth	earth #1: Earth world globe	Astronomy
	earth #2: estate landed_estate ...	Economy
	earth #3: clay	Geology
	earth #4: soil dirt	Geology
	earth #5: ground_earth	Electricity
	earth #6: dry_land solid_ground ...	Geography
	earth #7: land ground soil	Geography
	earth #8: earth ground	Geology

Figure 2. Word Domain Disambiguation.

At the second step, all the possible tuples of domain labels for each word are scored by means of a similarity function, and the best tuple is selected. The similarity between two domains is computed according to the probability of the two domains to co-occur within a text. This information has been computed over several balanced corpora, both for English (i.e. the Brown corpus, the LOB corpus and the Reuters news corpus) and for Italian (i.e. the Elsnet corpus and a large collection of newspaper news). In our example, the algorithm selects Astronomy for "star", Physics for "bright", Astronomy for "visible" and Astronomy for "earth", which correspond to our intuition about the involved word senses.

Results obtained over the 200 TREC questions, previously manually annotated with the correct domain label for each keyword, are very encouraging, showing a limited loss in accuracy with respect to WDD over longer texts, where larger pieces of context are available for disambiguation.

3.2 Answer Type Identification

The answer type for a question represents the entity to be searched as answer. In order to extract this information, a taxonomy of *answer types* was manually defined starting from the 200 TREC-8 questions. The taxonomy includes categories such as "LOCATION", "PERSON", "TIME-PERIOD", "MEASURE" and "GENERIC". Then, each category is associated with a set of rules that check different features of the question; in particular a rule may detect the presence of a particular word occurrence, of words of a given part of speech, and of words belonging to a given semantic category. For instance, the rule described in (1) matches any question starting with "quale" ("*what*"), whose first noun, if any, is a person.

```
(1)  RULENAME: WHICH-WHO
        TEST: ["which"   [¬NOUN]*
        [NOUN:person-p]ȷ  +]
       OUTPUT:  ["PERSON"  J]
```

Rule (1) matches questions like "*What famous communist leader died in Mexico City?*" because the first noun encountered after "what" (i.e. "leader") satisfies the person-p constraint. The same rule does not match the question "*What large U.S. city had the highest murder rate for 1988?*", because "city" does not satisfy the person-p predicate. Semantic predicates (e.g. location-p, person-p, time-p, etc.) are defined on the MultiWordNet taxonomy, already described in Section 3.1. Each predicate checks if the sense of a word referred in a rule is subsumed by at least one high level synset manually defined for that predicate. As an example, for person-p, we identified synsets like person#1 ("human being") and group#1 ("any number of entities considered as a unit"). Then the predicate is satisfied if leader#1 is subsumed by at least one of these synsets. ·

While rules are mostly language dependent, semantic predicates defined on MultiWordNet are reusable for English. For instance, rule (2) is the corresponding rule for Italian:

(2) RULENAME: QUALE-CHI
TEST: ["quale" [¬NOUN]*
[NOUN:person-p]$_J$ +]
OUTPUT: ["PERSON" J]

The output of a rule gives two pieces of information: the category of the answer type and the focus of the question (Moldovan et *al.* 1999), i.e. the word that expresses the answer type in the question. In the examples above, the answer type is the category PERSON, while the focus is the word "leader". This information will be used to retrieve the correct answer to the question in the documents retrieved by the search engine. In particular, knowing the category of the entity we are looking for (i.e. a PERSON) the focus will be used to determine if any "candidate answer" we find in a document (i.e. person names) is an appropriate instantiation of that category. This can be done by accessing the MultiWordNet taxonomy and checking if the candidate answer is a synonym of the focus or is subsumed by the focus.

Currently we use about 90 answer type rules for Italian and about 70 for English. They have been checked on the TREC corpus resulting respectively in a 93% accuracy and 91%. Failures are due mainly to pos-tagging and disambiguation errors.

3.3 Keyword Expansion

At this step 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.

Morphological derivation. The approach adopted is answer oriented, in that it considers the expansions with the higher probability to appear in the answer. For instance, given the question "*Who invented the electric light?*", five expansions are automatically generated for the basic keyword "invent": the past participle masculine "inventato", because this is the actual form of the lemma; the past participle female "inventata", because the direct object of the verb is female; the past indicative "inventò", because in Italian it can substitute the past participle; the noun "inventore", because it is the nominalization of the subject of the verb; finally, the noun "invenzione", because it is the nominalization of the object of the verb. Derivations have been automatically extracted from an Italian monolingual dictionary (Disc, 1997).

Synonyms. The approach adopted for disambiguation, i.e. word domain disambiguation, is in line with this assumption: domains allow the clustering of related WordNet senses. Once a domain label for a word is selected by the disambiguation algorithm, synonyms are collected from all the MultiWordNet synsets for that word belonging to the selected domain. For instance, given the morphological expansions described above for the verb "invent", a number of synonyms extracted from MultiWordNet are added, including *discover* and *discoverer*.

3.4 Definition questions

In the TREC10 main-task there are several definition questions, whose answer requires a definition of the focus. For these questions a number of specific patterns have been defined, which consider the following features:

- definition questions typically begin with "Who" or "What";
- they confine to the pattern "Who/What be sNP?", where "be" stands for the different possible forms of the verb "to be" and sNP is a simple noun phrase, which consists of a noun or adjective + noun (eventually preceded by article). For example: "Who was Galileo?" or "What is an atom?".
- Anther specific pattern for definition question is "What does ABBREV stand for?" (e.g. "What does NASA stand for?")

The peculiarity of these questions is that the keywords themselves do not provide enough information for extracting the proper answer. For the above mentioned questions the keywords are just "Galileo","atom" and "NASA".

By restricting ourselves to simple NPs consisting of noun or adjective + noun, we avoid the improper coverage of non-definition questions. For example the *what-where* questions described in (Moldovan et *al.* 1999) are not considered definitions because they usually use complex NPs (e.g. "What is the capital of Uruguay?").

In the case of a definition question, DIOGENE makes use of the WORDNET glosses to expand the focus of the question. The intuition behind the WORDNET gloss expansion is that because the expected answer for a definition question will be a definition of the question focus, it is reasonable to expect that it contains words that also appear in the WORDNET gloss of the focus (i.e. its definition). For instance, in the case of "Who is Galileo?" the related keywords and multiwords extracted from the gloss of "Galileo" are "astronomer", "Italian", "mathematician", and "refracting telescope". The gloss keywords are used to extend the query and they are also considered during answer extraction.

The gloss keywords extraction takes into account the gloss nouns, because we consider nouns more informative than verbs, adjectives and adverbs. All the other parts of speech are considered only if they begin with a capital letter (in the Galileo gloss – "Italian"), because usually they denote names.

4. Search Component

At this moment DIOGENE makes use of the ZPrise (Dimmick, 1998, Downey, 1999) search engine developed by the Retrieval Group at NIST. In particular, we used PRISE 2.0, which is part of the Z39.50/PRISE2.0 package. ZPrise is based on vector techniques and supports term feedback; on the other hand it does not support Boolean search and has restricted phrase search capabilities.

Query composition is performed after keywords are extracted from the question and every word is supplied with an expansion list containing its word form and its expansions. Let's see how the query composition looks in the case of the question "Who is the inventor of the electric light?" After part-of-speech tagging and multiword extraction, the following keywords (key-phrases) are taken: "electric light" and "inventor". The lexical expansion produces the following expansion list for every key-phrase: "electric light", "electric light bulb", "incandescent lamp", and also morphological derivatives for these nouns: for "inventor" the expansion is "inventor" "artificer" "discoverer" "inventors" "artificers" "discoverers". For definition questions, gloss keywords are also extracted.

Previous experiments on search modalities (Magnini and Prevete, 2000) have proven the advantage of a Cartesian product search modality, in which a Boolean expression based on the Cartesian product of the expansion lists is generated. Because of the lack of Boolean expression support in ZPrise we realised the AND expressions as a single ZPrise query and the OR expressions were transformed into multi-line queries. Given the weak support of phrases in ZPrise we had to decompose multiword units into separate words, which diminishes the precision of the search. In addition, in the process of query composition no more than ten keywords for a question are taken into account because ZPrise slows down significantly when a big number of keywords are presented in one query line. Keywords are scored taking into account their part of speech (e.g. nouns score better then verbs), their polysemy (i.e. polysemous words score worse that less polysemous) and the first character of the word.

The speed of the search engine turned out to be a set-back during the time of the TREC-10 questions processing. To speed up the performance we divided the document collection into four sub-collections and created a separate index for every sub-collection. This made possible a parallel search with four instances of ZPrise running on four different computers simultaneously, while every instance processes its own sub-index. After that the results are unified using UNIX shell scripts specifically developed for this purpose. This strategy significantly improved the time for the search. The method of parallel work was also used in the final stage of the question processing.

5 Answer Extraction Component

5.1 Paragraph Filtering

A paragraph filtering module extracts text paragraphs from the top scored retrieved documents. This is done maximizing the number of keywords and expansions produced in the question processing phase, within a window of a fixed length of words. The output is composed by text paragraphs that should contain the answer to the question. Paragraph filtering can be tuned with three parameters: (i) the length of the paragraphs to be extracted, which was set at 200 words; (ii) the percentage of keywords and expansions that need to be present in the paragraph; (iii) a list, eventually empty, of obligatory keywords, whose presence is necessary.

5.2 Named Entities Recognition

Once the relevant paragraphs have been selected, the named entities recognition module identifies, among possible candidate answers, the entities that match the answer type category (i.e. PERSON, ORGANIZATION, LOCATION, MEASURE and DATE). The task is performed by *Learning*-PINOCCHIO (Ciravegna, 2000), a system for adaptive information extraction developed at ITC-Irst. *Learning*-PINOCCHIO learns template filling rules that insert SGML tags into texts without any other user intervention than corpus tagging.

During the training phase the system induces a large set of rules by bottom-up generalization of instances in a tagged training corpus where texts have been manually marked with SGML tags locating the information to be extracted (in our case the tags *<person>*, *<\person>*, *<organization>* *<\organization>*, *<location>* *<\location>*, *<measure>*, *<\measure>*, *<date>* and *<\date>* have been used). Rule induction is also supported by dictionaries, such as repositories of person's first and last names, geographical and organization names extracted from gazetteers, online resources and WORDNET.

The training corpus used (~400Kb) was randomly extracted from the TIPSTER collection, part of the TREC material at our disposal. As expected, system performances are highly affected by the dimension, format and domain of the training corpus: documents contained in the TIPSTER collection proved to be rather heterogeneous, hampering the construction of a representative training corpus. Results obtained on the training corpus vary among categories, ranging from 74.5% precision and 82% recall for the category DATE, to 60% precision and 57% recall for the ORGANIZATION category.

5.3 Answer Identification

Answer identification is performed after named entities are recognised by *Learning*-PINOCCHIO. In case the answer type of the question is consistent with the type of an entity *c* in a given paragraph *p*, the entity is accepted as a candidate answer. Then, the candidate entity is scored considering the presence of relevant keywords in a fifty byte interval around it. More formally, given:

- $t_1, t_2, \ldots, t_m$ is the sequence of tokens composing the paragraph *p*, with *i* denoting the position of the i-th token t_i.
- *k*, *s* and *e* are functions such that *c* is composed of *k(c)* tokens, *s(c)* is the position of the first token and *e(c)* is the position of the last token of *c* in *p*.
- *kl(p)* is the list of keywords belonging to *p*.
- *len* is the *kl* length
- member(i,p) is a function which gives as output 1 if the token t_i is a *kl(p)* member, otherwise 0.

We define the *left key density* and the *right key density* of the answer candidate *c* in paragraph *p*, indicated respectively LKD(c,p) and RKD(c,p), using the following pair of functions:

$$\mathrm{LKD(c,p)} = \frac{1}{len} \sum_{i=1}^{s(c)-1} \frac{\sqrt{member(i,p)*\alpha}}{s(c)-i}$$

$$\mathrm{RKD(c,p)} = \frac{1}{len} \sum_{i=e(c)+1}^{m} \frac{\sqrt{member(i,p)*\alpha}}{i-e(c)}$$

Where α is a tuned parameter.

In order to build a 50 byte answer for a candidate entity *c*, we initially consider a default string A composed of the words in *c* (an entity can be composed of more than one word, for instance "Bill Clinton" is an entity whose type is PERSON). If A is composed of less than 50 characters other characters are appended to A both on its left and on its right. The number of characters added to the left is defined by LKD(c,p), while the number of characters added to the right is defined by RKD(c,p).

6 Results and Discussion

Being our first experience with a QA system and given the high number of different modules assembled in DIOGENE, we did not expect high performance. The system correctly answered just 10% of the questions of the TREC main task. A manual analysis over a small set of questions was carried out to evaluate problems related to single modules. Four modules were considered: answer type identification, search engine, paragraph filtering and named-entities recognition. Each module was evaluated in term of its error rate, given a small amount of correct inputs. The estimated error rate for the answer type module was 11%, which is comparable with the performances calculated at training time (see Section 3.2). ZPrise produced a very high error rate (53%), which means that for less than half of the questions the search engine retrieved a document containing the answer. The main reason is that filters (i.e. no more than 10 documents for a question and no document with length exceeding 2000 words) applied to ZPrise output are actually too restrictive. They were implemented to take the processing time under control, but their effects were underestimated. As for paragraph filtering, its estimated error rate is around 40%, which also indicates that the techniques for paragraph extraction can be significantly improved. Finally, the estimated error rate for *Learning*-Pinocchio was around 60%, which was mainly due to the low homogeneity between training and test corpus.

7. Conclusion and Future Work

The system described in this paper participated to the main task of the TREC-10 competition. Even though it relies on a rather standard architecture, DIOGENE deals with two important issues related to QA: multilingual QA and QA on the Web. For these purposes, some modules already available in ITC-Irst have been used. The results obtained by the whole system and the performance of each module have been described in the paper.

Another crucial issue for the future is the automatic evaluation of a QA system. The basic idea is to develop an evaluation methodology that does not rely on the human judgment of thousands of answers. Although there is some recent work in this direction (Breck et al. 2000), the approach we are testing considers the Web as the main information source to evaluate the relevance of an answer. Our work is based on the assumption that if a certain answer is relevant with respect to a given question, then there should be many documents containing keywords extracted both from the question and from the answer. Moreover, these documents should be semantically similar documents, i.e. they should maximize the overlapping of semantic features, such as the set of domains labels (see Section 3.1) extracted from each text.

References

Breck, E.J., Burger, J.D., Ferro, L., Hirschman, L., House, D., Light, M., Mani, I.: How to Evaluate Your Question Answering System Every Day …and Still Get Real Work Done. Proceedings of LREC-2000, Second International Conference on Language Resources and Evaluation (2000) 1495-1500.

Ciravegna, F.: Learning to Tag for Information Extraction from Text. In Ciravegna, F., Basili, R., Gaizauskas, R. (Eds.) ECAI Workshop on Machine Learning for Information Extraction, Berlin (2000).

Dimmick D., O'Brien G., Over P. and Rogers W.: Guide to Z39.50/PRISE2.0: Its Installation, Use and Modification.

Gaithersburg, USA, unpublished resource, (1998).

Downey L., Tice D.: A Usability Case Study Using TREC and ZPrise , Information Processing and Management, 35 (5), 589-603 , (1999).

Disc. Dizionario Italiano Sabatini Coletti. Giunti, Firenze (1997)

Fellbaum, C.: WORDNET, An Electronic Lexical Database. The MIT Press (1998).

Ferret, O., Grau, B., Hurault-Plantet, M., Illouz, G., Jacquemin, C., Masson, N., Lecuyer, P.: QALC – The Question Answering System of LIMSI-CNR. Proceedings of the Ninth Text Retrieval Conference (TREC-9), Gaithersburg, MD. (2000).

Harabagiu, S., Pasca, M., Maiorano, S.: Experiments with Open-Domain Textual Question Answering. Proceedings of Coling-2000 (2000a).

Magnini, B., Cavaglià, G.: Integrating Subject Field Codes into WORDNET. Proceedings of LREC-2000, Second International Conference on Language Resources and Evaluation (2000) 1413-1418.

Magnini, B., Prevete, R.: Exploiting Lexical Expansions and Boolean Compositions for Web Querying. Proceedings of the ACL workshop on Recent Advances in Natural Language Processing and Information Retrieval, Hong Kong (2000) 13-21.

Magnini, B., Strapparava, C.: Experiments in Word Domain Disambiguation for Parallel Texts. Proceedings of the ACL workshop on Word Senses and Multilinguality, Hong Kong (2000) 27-33.

Moldovan, D., Harabagiu, S., Pasca, M., Mihalcea, R., Goodrum, R., Girju, R., Rus, V.: LASSO: A Tool for Surfing Answer Net. Proceedings of the Eight Text Retrieval Conference (TREC-8) (1999) 65-74.

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).

Pianta E., Bentivogli L., Girardi C.: MultiWordNet: Developing an Aligned Multilingual Database. Proceedings of the 1st International Global WordNet Conference, Mysore, India (2002)

Schmid, H.: Probabilistic Part-Of-Speech Tagging Using Decision Trees. Proceedings of the International Conference on New Methods in Language Processing (1994).

Voorhees, E.M., Tice, D.M.: The TREC-8 Question Answering Track Evaluation. Proceedings of the Eight Text Retrieval Conference (TREC-8) (1999).

Voorhees, E.M.: Overview of the TREC9 Question Answering Track. Proceedings of the Ninth Text Retrieval Conference (TREC-9) (2000).

JHU/APL at TREC 2001: Experiments in Filtering and in Arabic, Video, and Web Retrieval

James Mayfield, Paul McNamee, Cash Costello, Christine Piatko, and Amit Banerjee
Research and Technology Development Center
Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Road, Laurel, Maryland 20723-6099 USA
{mayfield, mcnamee, costello, piatko, banerjee}@jhuapl.edu

Overview

The outsider might wonder whether, in its tenth year, the Text Retrieval Conference would be a moribund workshop encouraging little innovation and undertaking few new challenges, or whether fresh research problems would continue to be addressed. We feel strongly that it is the later that is true; our group at the Johns Hopkins University Applied Physics Laboratory (JHU/APL) participated in four tracks at this year's conference, three of which presented us with new and interesting problems. For the first time we participated in the filtering track, and we submitted official results for both the batch and routing subtasks. This year, a first attempt was made to hold a content-based video retrieval track at TREC, and we developed a new suite of tools for image analysis and multimedia retrieval. Finally, though not a stranger to cross-language text retrieval, we made a first attempt at Arabic language retrieval while emphasizing a language-neutral approach that has worked well in other languages. Thus, our team found several challenges to face this year, and this paper mainly reports our initial findings.

We also made a last-minute (really a last 36 hour) effort to participate in the web retrieval track. We unearthed a year-old index and the software that we used for the web task at TREC-9, and very quickly produced some official submissions. Our main interest in the home-page finding task was to submit content-only runs that could serve as a simple baseline to which other group's sophisticated hyperlink-influenced approaches might be compared. We simply did not have the time to seriously investigate the more complex problems being examined by the web track; however, we wanted to be good TREC citizens and contribute to the document pools.

All of our text-based investigations were based on the Hopkins Automated Information Retriever for Combing Unstructured Text, or HAIRCUT system. HAIRCUT is a Java-based tool developed internally at JHU/APL that was first used to compare tokenization methods during TREC-6. HAIRCUT benefits from a basic design decision to support flexibility throughout the system. For example, the software supports words, stemmed words, character n-grams, and multiword phrases as indexing terms. And, several methods for computing document similarity are supported, though we recently have relied on probabilistic methods based on statistical language modeling techniques.

In general, we have seen better performance using language models than when using cosine-based vector scoring. In our experiments we used a linguistically motivated probabilistic model. Hiemstra and de Vries describe this 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 [15]. 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 expressed as

$$Sim(q,d) = \prod_{t=terms} \left(\alpha \cdot f(t,d) + (1-\alpha) \cdot df(t)\right)^{f(t,q)}$$

Equation 1. Similarity calculation.

where *(1-α)* is the probability that a query word is generated by a generic language model, and α is the probability that it is generated by a document-specific model. *df(t)* denotes the relative document frequency of term *t*.

We conducted all of our work on a set of four Sun Microsystems workstations that are shared among our department (80 physicists, chemists, engineers, and about 25 computer scientists). Two of the machines are 4-node Sun Microsystems Ultra Enterprise 450 servers with 2.5 and 4.0 GB of physical memory, respectively; the other two machines are Sun Ultra-2 workstations with 1.25 of RAM. This cluster has 200GB of dedicated, networked disk space for use in our retrieval work.

Filtering Track

We participated in both the routing and batch tasks for the filtering track. We did not use any of the hierarchy information available with the Reuters categories for either task.

Routing Task

Our goal for the routing task was to evaluate the use of a statistical language model for routing. We submitted two runs, one based on a character n-gram (*n=6*) index (*apl10frn*) and one based on a stem index (*apl10frs*) using a derivative version of the SMART stemmer. We also created an unofficial word-based run (*apl10frw*). We simulated routing, using a modified version of HAIRCUT system to score indexed test documents using training index statistics – the statistical language model described above was used for scoring. We formed queries using 60 terms per topic that were selected from the positive batch qrels documents. Term selection was accomplished using mutual information based difference statistics with respect to the August 96 training data.

We were pleased with our official results for our first participation in this task. We were excited to participate in the "routing bet" discussion and we can report that we have 28 queries (exactly 1/3 of the queries) with ≥ 0.9 precision at 1000 docs in both our official runs. The closeness of the results indicates the choice of terms is not critical.

	Avg. prec.	# bests	# ≥ median (84 topics)
apl10frn	0.121	4	70
apl10frs	0.104	4	56
apl10frw	0.113	unofficial run	

Table 1. APL Routing Results

Batch Task

Our goal for the batch task was to evaluate the effectiveness of Support Vector Machines (SVMs) on the new Reuters data set [22]. SVMs are used to create classifiers from a set of labeled training data. SVMs find 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.

For the batch task, we sought to explore the effects of different parameter choices on learning with this Reuters collection. We were interested in the use of tf/idf weighted vectors vs. per-topic binary vectors; the use of radial basis function (RBF) vs. linear kernels in the SVMs; score thresholds on resulting classifier scores; and training skew factors to incur less error on positive examples. This follows earlier work in text batch filtering on the original smaller Reuters collection [2] [6]. We used the SVM-light package (version 3.50, by Thorsten Joachims [19]) to create classifiers based on the training data for classification of the test data. We used a reduced feature space for both batch submissions. For all runs, we normalized document vectors to unit length.

Our post-submission results show: tf/idf training-derived features were better than topic-specific binary ones; RBF kernels were slightly better than linear kernels; aggressive score thresholding hurt our tf/idf runs, while it helped improve our binary runs; fixed skew was not as good as the per-topic skew developed by others in the track.

Batch Using Linear SVMs with Binary Vectors

For the submitted run *apl10fbsvml* we used 200 terms derived on a per-topic basis to create binary term vectors for each document (our implementation actually created a different document vector for each topic). The terms were selected from each topic's positive qrels documents, using mutual-information-like difference statistics with respect to the August 96 training sample. Given *n* positive training documents for a topic, we randomly chose *n* potentially negative examples from the full training index, and threw away any that were actually positive. We created linear SVMs, weighting positive and negative training examples equally (-j 1 flag in SVM-light). *J* is a cost or skew factor, by which training errors on positive examples outweigh errors on negative examples (see [5]).

We then used the score of the test document using the topic SVM to decide whether to return the document. In experiments reported in the literature, SVMs scores are normally thresholded above zero. However, we had observed many training errors close to zero; many negative examples were misclassifed with a small positive score. We thus experimented with setting higher score thresholds. We debated using a small epsilon to threshold the score, but decided to try to find the "best" scores per topic automatically to maximize the 2R+ -N+ measure for the training data. While the overall approach did not work all that well, thresholding did salvage something out of these particular vectors. Unofficial runs using a zero threshold did worse, for both *j=1* and *j=5* (runs *BINLIN skew1* and *BINLIN skew5* in Table 4 below).

We do not know why this approach did not succeed. We considered trying different values of *j* to weight positive and negative examples differently. Perhaps more negative training data or a greater number of terms would improve the technique. Finally, our main intuition is that binary features are probably not appropriate for this Reuters dataset.

Batch Using RBF SVMs with TFIDF Vectors

For the submitted run *apl10fbsvmr* we used a reduced term space of 2000 terms to create all the test and training document vectors, based on all the training data. The terms were selected using the top 2000 stems by document frequency in the training set. Stems were produced using a derivative of the SMART stemmer and stopwords were not removed. We created tf/idf weighted vectors for each document and each vector was normalized to unit length. Given *n* positive training documents for a topic, we randomly chose *4n* potentially negative examples from the training index, and threw away any that were actually positive. We then trained radial basis function SVMs (using the *-t 2 -g 1* flags in SVM-light), weighting positive and negative training examples equally *(-j 1* flag in SVM-light). Using thresholds higher than zero to classify the test documents, as we did with linear kernels, proved to be a big mistake. It hurt performance significantly. Set precision was good, but set recall was terrible.

We redid this run using the same RBF models with zero as the score threshold (*RBF skew1*), and are much happier with the results. We also did some runs using weighted RBF models with *j=5* (*RBF skew5*) and similarly tried linear kernels (*LIN skew1* and *LIN skew5*). These post-hoc experiments confirm that SVMs can work well for the batch task, using either radial basis functions or linear separators with tf/idf weighted vectors normalized to unit length.

We expect there are many per-topic optimizations (such as the leave-one-out cross-validation on training data Dave Lewis used to find optimal *j* weights per topic [8]) that could dramatically improve these initial findings.

Results

	T10SU	Fbeta	SetPrec	SetRecall
apl10fbsvml	0.115	0.292	0.303	0.627
apl10fbsvmr	0.081	0.154	0.380	0.054

Table 2. Official Batch Submissions.

	T10SU	Fbeta	SetPrec	SetRecall
RBF skew1	0.283	0.459	0.546	0.437
RBF skew5	0.254	0.430	0.442	0.525
LIN skew1	0.234	0.413	0.400	0.601
LIN skew5	0.157	0.341	0.318	0.689

Table 3. Unofficial (post hoc) batch runs, unified tf/idf weighted term space.

	T10SU	Fbeta	SetPrec	SetRecall
BINLIN skew1	0.030	0.132	0.113	0.835
BINLIN skew5	0.009	0.085	0.071	0.895

Table 4. Unofficial (post hoc) batch runs, per-topic binary term space.

Summary of Batch Filtering Results

Chart 1 summarizes the results of our batch filtering experiments. SVMs with RBF kernels on TFIDF vectors and no thresholding works well, and could have performed above median compared to other official batch results. Thresholding above zero hurt for RBF SVMs on TFIDF vectors (*RBF skew1* vs. *apl10fbsvmr*). However thresholding improved a poor baseline result of linear SVMs on binary vectors (*apl10fbsvml* vs. *BINLIN skew1*).

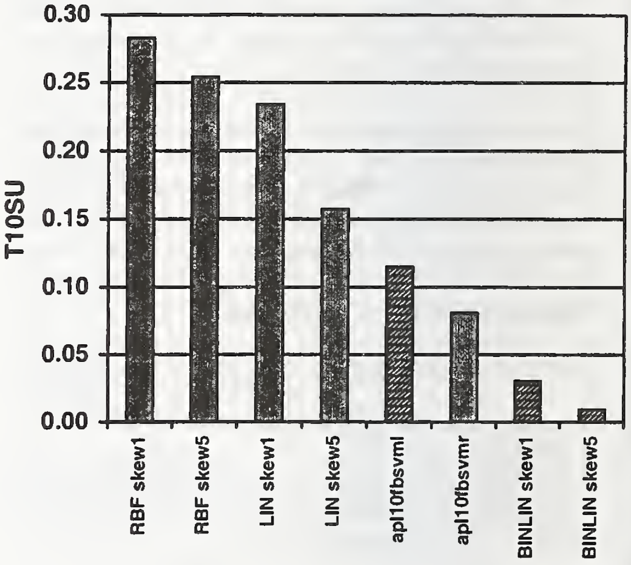


Chart 1. SVMs with RBF kernels on TFIDF vectors work well for batch filtering with the T10SU metric.

Video Retrieval

The video track was a new addition to TREC this year. It consisted of three tasks: shot boundary detection, known-item search and general search. The data set was eleven hours of mostly documentary video from the Open Video Project at University of North Carolina Chapel Hill and the NIST Digital Video Collection. JHU/APL did not have any previous experience with video or image retrieval so participation in this track was a valuable learning experience. A significant amount of time had to be devoted to developing the software infrastructure needed to process MPEG video, create an index, and parse queries. This led to a philosophy of "simple is best."

For the shot boundary detection task, we experimented with using color histograms, luminance, and the raw image gradient of frames to locate hard cuts and gradual transitions. Hard cuts were identified using an ad hoc global threshold on the color histogram intersection of consecutive frames [16]. With gradual transitions, possible dissolves and fades were first detected by looking for abrupt changes in the average luminance of frames. These possible gradual transitions were then evaluated by analyzing the change in the image gradient. Each frame was divided into eight-by-eight blocks. If a large percentage of the blocks had changes in the image gradient greater than some threshold, the presence of a dissolve or fade was confirmed. The same technique was used to locate the start and end of each gradual transition. This approach was based on the work of Zabih, Miller, and Mai [17], the major differences being that we did not perform motion compensation and used raw image gradients rather than edges. This resulted in a method that was less computationally expensive than the typical edge entering and exiting method. The method did not perform well in the evaluation, but we did not have sufficient time to experiment with different variations and thresholds. The only interaction between the algorithm for detecting hard cuts and those for detecting gradual transitions was the hard cut algorithm taking precedence if a cut and a transition were detected in close proximity. A summary of the results for shot boundary detection is shown in Table 5.

	Total videos	# ≥ median	# best
Cuts-prec	15	12	8
Cuts-recall	15	4	1
Graduals-prec	17	0	0
Graduals-recall	17	4	2

Table 5. Shot boundary detection results

Because of limited time and experience, our approach to video retrieval was to treat a video as a series of still images. We made no attempt to exploit the extra information available with video and not with images, such as the audio track and object motion. The experiments we performed focused on using color histograms and image texture features. Each video was first decomposed into shots using the shot boundary detection algorithms described above. The middle frame was used as the key frame to represent all the content of the frames in the shot. This is not a complete representation, but it fit with the emphasis on simplicity for the sake of expediency. In fact, the index files for the 6.3 GB data set comprised only 31 MB altogether, less than 1% the size of the source data. A key frame was described by a vector that contained color and texture features. Similarly, each query was also represented by one or more of these vectors. For a description of the vectors, see Table 6.

Keyframes	Dimensions	Color features	Texture features
7391	272	256	16

Table 6. Description of video index and vector features

When processing queries, any text or audio was completely ignored. If a video example was provided for a query, just the middle frame was extracted as an image example. A weighted distance measure was used for evaluation with the key frames ranked by minimum distance to the set of query examples. The weights were chosen so that the texture and color features made approximately the same contribution to the distance measure even though there were fewer texture measures. The texture features were calculated using a texture descriptor proposed by Manjunath [9]. It creates a multiresolution decomposition using a Gabor filter bank. We used code available from the Image Processing and Vision Research Lab at the University of California, Santa Barbara [18] to calculate these features.

While our results from the known item task were close to the median, the results from the general search were significantly below average. We have not had time to completely investigate this disparity. One explanation for this is that the general information need queries depend more on the text description of the query than on the image or video examples. Since we discarded this information when parsing the query, we were at a disadvantage when trying to retrieve relevant video clips for general searches. The three queries on which we were above the median would support this hypothesis; the text descriptions were short with little information contained in them. "Other shots of city scapes," which is the text description of a query where we were above the median, is a good example. In the known item task, the queries we scored the best on asked about objects that have a strong color component: "Scenes with a yellow boat" or "Other examples of the surface of the planet Mars." This result agrees with the strong emphasis we placed on color in the representation of video data.

Arabic Language Retrieval

The Cross-Language Retrieval task at TREC 2001 consisted of bilingual retrieval of Arabic newspaper articles given either English or French topic statements. Monolingual submissions were also accepted using the manually produced Arabic translations of the topics.

The apparent necessity of having quality translation resources available for use in a CLIR system has often been expressed. For example, at the first CLEF workshop, Anne Diekema gave a provocative talk, suggesting that CLIR evaluation was essentially just evaluation of translation resources [1]. We spent several days searching the Web for extant collections of parallel corpora or bilingual dictionaries that would be helpful for translating to Arabic, with no real success. We finally found one newspaper that published mappable, parallel content in both Arabic and English, only to discover that the Arabic stories were available only as images (a practice that stems from the historic lack of standards and software for displaying Arabic text). Downloading that GIF files, OCRing them, and building a parallel collection was beyond our means.

Unable to discover or acquire significant translation resources, we relied exclusively on two on-line machine translation systems, Ajeeb [20] and Almisbar [21]. Recently, Kraaij showed how translation probabilities can be incorporated nicely into a language model for cross-language text retrieval, and he demonstrated the efficacy of this combination at the CLEF-2001 workshop [7]. However, since we simply used machine translation for query translation we did not have access to translation probabilities that are available when dictionaries and corpus-based approaches are used. All of our work was with fully automated retrieval.

This was JHU/APL's first experience with Arabic document processing and we learned quite a lot from the experience. We had no personnel who could read Arabic. This however, did not dampen our enthusiasm for the task in the slightest. Over the last several years, our team at APL has participated in multiple CLIR evaluations, where large document collections in Chinese, Dutch, English, French, German, Italian, Japanese, and Spanish were searched [10] [11] [12] [13] [14]. While these higher-density languages tend to have many resources available for linguistic analysis and automated translation, these languages are diverse, and use numerous character sets and character encodings. Our approach for combating the inherent scalability issues presented by working with numerous languages has been to focus on simple, language-neutral approaches to text processing. Counterintuitively, we have not found that sophisticated, linguistically-rich approaches demonstrate an appreciable performance advantage over the knowledge-light methods we espouse.

One example of a language-neutral technique is the use of overlapping character n-grams. We have found that n-grams work well in many languages and a pseudo linguistic normalization occurs in agglutinative languages such as Dutch and German [11] N-grams are more widely used for retrieval in Asian languages; we recently showed that 3-grams perform on par with 2-grams in unsegmented Japanese text [12], which is not the case with Chinese [14]. Our use of 6-grams for indexing Arabic was not founded on linguistic principles or empirical evidence – we simply guessed that it would be a good choice as it has been in many other alphabetic languages. In retrospect, shorter n-grams have proven to work better with Arabic. In addition to examining the choice of words or n-grams as indexing terms, we experimented with eliminating or replacing certain Arabic characters that did not appear in a list of 28 letters that we had available. Thus we built four different indexes; summary information about each is shown in Table 7.

	# terms	index size
words	571,798	372 MB
words - morph	539,979	351 MB
6-grams	6,784,129	2513 MB
6-grams - morph	6,081,618	2427 MB

Table 7. Index statistics for the 869 MB, 384K article TREC-2001 Arabic collection.

Our submissions were produced by combining multiple base runs using different combinations of the topic statement fields, and different methods for morphological normalization, tokenization, query expansion, and translation. One monolingual run, three bilingual runs from English topics, and one cross-language run using the French topics were submitted. For our monolingual Arabic run, *apl10ca1*, we relied on eight constituent runs

- 2 query formats: TD and TDN
- 2 choices for relevance feedback (yes or no)
- 2 tokenization alternatives, words and 6-grams
- 1 normalization approach, character elimination was used

Thus, eight different base runs were created, and merged together to produce *apl10ca1*. See [13] for details of the merging strategy.

Apl10ce1, was our first bilingual run using the English topics. We used the exact same approach as *apl10ca1*, but had two methods for translating the topics:

- 2 translation systems (Ajeeb and Almisbar)

Thus sixteen different base runs were combined to produce the submitted run.

Our second and third English bilingual runs only made use of the TD topic fields and used either words, or 6-grams as indexing terms. The second run, *apl10ce2* used eight base runs:

- 1 query format: TD
- 2 choices for relevance feedback (yes or no)
- 1 tokenization alternative: 6-grams
- 2 normalization approaches, character elimination was used, or not
- 2 translation systems (Ajeeb and Almisbar)

The third English bilingual run, *apl10ce3*, was just like *apl10ce2*, except that words were used in place of n-grams.

Finally, we submitted one run using the French topic statements, *apl10cfl*. The base runs for this used:

- 1 query format: TDN
- 2 choices for relevance feedback (yes or no)
- 2 tokenization alternatives: words and 6-grams
- 1 normalization approach, the character elimination was used
- 2 translation systems (Ajeeb and Almisbar) from English to Arabic
- 1 translation system for French to English (Systran)

Thus, when using the French queries, we first translated to English using the Systran product, and then translated to Arabic using one of the two online systems (Ajeeb/Almisbar). Interestingly, this second layer of translation did not seem to cause much loss in retrieval effectiveness. This may be due to the generally high performance of the Systran English/French module.

Official results

An overview of APL's five official runs for the Arabic track are shown in Table 8 below.

	MAP	Recall (4122)	# best	# ≥ median	% mono
apl10cal	0.3064	2669	3	17	100 %
apl10cel	0.2891	2819	1	22	94.4
apl10ce2	0.2250	2593	0	16	73.4
apl10ce3	0.1914	2350	0	15	62.5
apl10cfl	0.2415	2574	0	20	78.8

Table 8. Official results for Arabic runs (25 topics)

We note that run apl10cel (bilingual English to Arabic) achieved 94.4% of the monolingual baseline observed in *apl10cal*. As yet, we are unable to ascertain whether this is do in part to our particular approach to retrieval, or is more a factor of the quality of the machine translation software we relied on.

Since the conference workshop in November, we have found better bilingual performance using n-grams of length four instead of the longer six-grams. This yielded an improvement in average precision from 0.2891 (*apl10cel*) to 0.3350. But our monolingual baseline also improved when 4-grams were used, from 0.3064 (*apl10cal*) to 0.3588. Thus, the relative bilingual performance drops insignificantly to 93.4%.

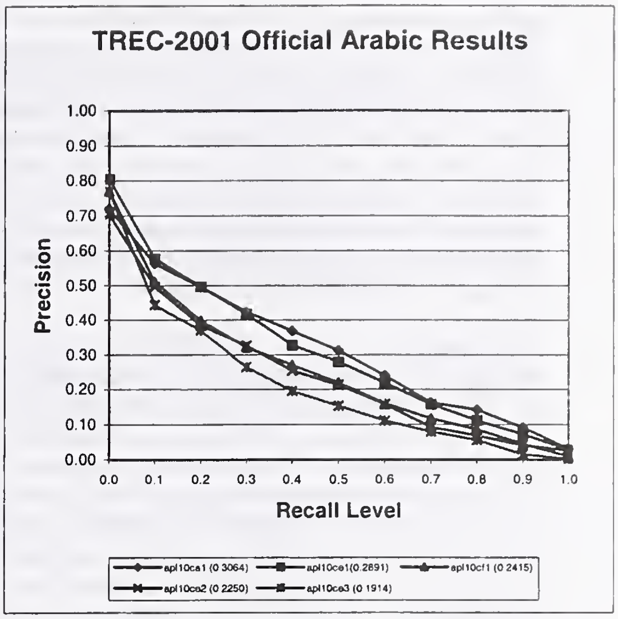


Figure 1. Recall-precision graph for APL's official Arabic track automatic submissions

We do observe that the use of n-grams accounted for a 17% relative improvement over words in mean average precision (0.2250 vs. 0.1914) as seen in the results for runs *apl10ce2* and *apl10ce3*..

We are still examining the data from all of our many base runs, and do not report on those runs. However, our preliminary analysis finds that character elimination was helpful, but the effect was not extremely large.

Web Retrieval

All of our work for this task was done in essentially one day using an index file previously created for the *wt10g* collection and used by APL during TREC-9. We submitted four content-only based runs for the ad hoc task, and produced two submissions for the homepage finding task. Our site finding runs were based entirely on query content; we did not use site popularity (backlink frequency) or any graph-theoretic analysis of the hyperlink structure. Our purpose was to see how well a pitifully under-informed approach would compare to the more sophisticated methods we anticipated others would apply to the problem.

We indexed documents using unstemmed words; the resulting dictionary contained over three million entries and the index files consumed roughly 3GB of disk space. Each document was processed in the following fashion. First, we ignored HTML tags and used them only to delimit portions of text. Thus no special treatment was given for sectional tags such as <TITLE> or <H1> and both tags and their attribute values were eliminated from the token stream. The text was lowercased, punctuation was removed, and diacritical marks were retained. Tokens containing digits were preserved; however only the first two of a sequence of digits were retained (e.g., 1920 became 19##). The result is a stream of blank-separated words. Queries were parsed in the same fashion as document, except that tried to remove stop structure from the description and narrative sections of the queries using a list of about 1000 phrases constructed from previous TREC topic statements.

After the query is parsed each term is weighted by the query term frequency and an initial retrieval is performed followed by a single round of relevance feedback. In performing blind relevance feedback we first retrieve the top 1000 documents. We use the top 20 documents for positive feedback and the bottom 75 documents for negative feedback; however duplicate or near-duplicate documents are removed from these sets. We then select 60 terms for the expanded query.

For the most part we ignored the web-nature of the documents and relied on textual content alone to rank documents.

Informational Task

We submitted four runs for this subtask, three runs that simply used the short (Title) portion of the topic statement, and one run that used all parts of the topic (TDN). The four runs were:

- *apl10wa*: title only, no blind relevance feedback
- *apl10wb*: title only, no blind relevance feedback, all query terms must be present in a document
- *apl10wc*: title only, with pseudo relevance feedback, all query terms must be present in the document
- *apl10wd*: TDN, with psuedo relevance feedback, no constraints on query term presence

	P@5	P@10	MAP	Recall (3363)	Bests / Median
apl10wa	0.1600	0.1460	.0805	1702	1 / 11
apl10wb	0.2400	0.1900	0.0671	599	1 / 9
apl10wc	0.2520	0.2380	0.1567	2105	2 / 28
apl10wd	0.3720	0.3380	0.2035	2525	2 / 30

Table 9. Performance of APL Official TREC-2001 Web submissions (Ad hoc)

Results for our official submissions are shown in Table 9. The submissions that used pseudo relevance feedback (RF) had much higher precision at 10 docs, mean average precision, and recall at 1000 docs. The run using all parts of the topic statement (*apl10wd*) had the highest performance across the board, including precision at 5 documents. Runs *apl10wb* and *apl10wc*, both of which required all query terms (only terms from the topic titles) to be present in returned documents, had about a 50 percent improvement in precision at 5 documents over *apl10wa*. This is important, because it suggests that when high precision is desirable, not all documents containing any query term need be examined, a practice common to many web search engines today (instead, the smaller set of documents that contain all of the query terms could be scored). Also, while *apl10wc* had high performance at higher recall levels than did *apl10wb*, this was not really true at high precision. This lends support for the practice of not using relevance feedback when only few relevant documents are needed to satisfy a user's need.

Navigational Task

We submitted just two runs for this subtask, and decided to see how well a purely content-based ranking would perform. As in the informational task, we compared performance between runs where all of the query terms were required to be present in relevant documents. We simply ordered our ranked list of hyperlinks using the similarity scores from the retrieval process. As was mentioned earlier, no use of document popularity or hyperlink structure was attempted. The two runs we submitted were:

- *apl10ha*: all terms required, no relevance feedback
- *apl10hb*: all terms not compulsory, no blind relevance feedback used

	MRR	% top 10	% failure
apl10ha	0.238	44.8%	22.1%
apl10hb	0.220	42.8%	21.4%

Table 10. Performance of APL Official TREC-2001 Web submissions (site finding task)

On the officially reported measures, mean reciprocal rank, percent of topics with a correct entry page found in the top 10 documents, and the failure percentage (when none was found in the top 100 docs), these two runs were virtually identical. The mean reciprocal rank is just slightly higher for apl10ha, in which all query terms were required to be on the given page.

Conclusions

This year we participated in three tracks that each presented new challenges: filtering, video, and Arabic.

We investigated the use of Support Vector Machines (SVMs) for batch text classification and noticed a large sensitivity to parameter settings for these classifiers. We also found that we were able to choose reasonable score thresholds for the routing task when using a language model for estimating document relevance.

Due to a lack of experience with multimedia retrieval (e.g., we had never previously participated in the TREC Spoken Document Retrieval task), the video track was a significant challenge for us. We placed an emphasis on simple techniques to quickly create a retrieval system while planning to add more advanced components such as speech recognition in the future. From our initial analysis, there was a correlation between how we parsed queries and our performance on different types of queries

Arabic retrieval was especially interesting for our team, which had no personnel who could read Arabic. The lack of available translation resources left us with little alternative but to use weak machine translation systems; yet, we found bilingual performance rivaled a good monolingual baseline in terms of mean average precision (94%), had equal performance at high precision levels (such as measured in precision at 5 or 10 documents), and even achieved higher recall at 1000. Our results emphasizing language-neutral techniques indicate that excellent performance is attainable without sophisticated linguistic processing.

While we did not put significant effort into the Web track this year, we did attempt to improve our retrieval performance at high precision levels (in contrast to our previous work attempting to maximize mean average precision). We found support for several techniques currently used in the commercial sector that improve query processing efficiency without impacting high precision performance.

References

[1] A. Diekema and W-Y Hsiao, 'Cross-Language Information Retrieval Using Dutch Query Translation'. In Carol Peters (ed.) *Cross-Language Information Retrieval and Evaluation: Proceedings of the CLEF-2000 Workshop*, Lisbon, Portugal, Lecture Notes in Computer Science 2069, Springer, pp. 230-236, 2001.

[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] Dave Lewis, personal communication, TREC 2001 Batch Filtering Task Experiments.

[9] B. Manjunath, P. Wu, S. Newsam, H. Shin. 'A Texture Descriptor for Browsing and Similarity Retrieval.' Journal of Signal Processing: Image Communication, 16(1), pp. 33-43, September 2000.

[10]J. Mayfield, P. McNamee, and C. Piatko, 'The JHU/APL HAIRCUT System at TREC-8.' In E. M. Voorhees and D. K. Harman, eds., *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pp. 445-451, 2000.

[11]P. McNamee, J. Mayfield, and C. Piatko, 'A Language-Independent Approach to European Text Retrieval.' In Carol Peters (ed.) *Cross-Language Information Retrieval and Evaluation: Proceedings of the CLEF-2000 Workshop*, Lisbon, Portugal, Lecture Notes in Computer Science 2069, Springer, pp. 129-139, 2001.

[12]P. McNamee, "Experiments in the retrieval of unsegmented Japanese text at the NTCIR-2 workshop," Proceedings of the 2nd NTCIR Workshop, 2001.

[13]P. McNamee and J. Mayfield, "JHU/APL experiments at CLEF 2001: Translation resources and score normalization". In Results of the CLEF-2001 Cross-Language System Evaluation Campaign (Working Notes), Darmstadt, Germany, pp. 121-132, 2001.

[14]P. McNamee, J. Mayfield, and C. Piatko, "HAIRCUT at TREC-9". In E. Voorhees and D. Harman (eds.), Proceedings of the Ninth Text REtrieval Conference (TREC-9), 2001.

[15]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.

[16] M. Swain and D. Ballard. 'Colour Indexing.' International Journal of Computer Vision, 7(1), pp. 11-32, 1991.

[17] R. Zabih, J. Miller, K. Mai. 'Feature-based Algorithms for Detecting and Classifying Scene Breaks.' Proceedings of the 4th ACM Int. Conf. on Multimedia, 1995.

[18] Image Processing and Vision Research Lab Texture Code, http://www-iplab.ece.ucsb.edu/texture/software/index.html

[19] http://ais.gmd.de/~thorsten/svm_light/

[20] http:/english.ajeeb.com/

[21] http://www.almisbar.com/salam_trans.html

[22] 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.

More Reflections on “Aboutness” TREC-2001 Evaluation Experiments at Justsystem

Sumio FUJITA
JUSTSYSTEM Corporation
Brains Park, Tokushima, 771-0189 JAPAN
+81-88-666-1000
Sumio_Fujita@justsystem.co.jp

ABSTRACT

The TREC-2001 Web track evaluation experiments at the Justsystem site are described with a focus on the “aboutness” based approach in text retrieval.

In the web ad hoc task, our TREC-9 approach is adopted again, combining both pseudo-relevance feedback and reference database feedback but the setting is calibrated for an early precision preferred search.

For the entry page finding task, we combined techniques such as search against partitioned collection with result fusion, and attribute-value basis re-ranking.

As post-submission experiments, distributed retrieval against WT10G is examined and two different database partitioning and three database selection algorithms are combined and evaluated.

Keywords

Aboutness, pseudo-relevance feedback, reference database, fusion, attribute-value basis re-ranking, distributed retrieval, collection partitioning, database selection.

1. INTRODUCTION

When a web page gives information to readers, readers have already understood what the information is about. Information can be about anything, but should be about something in order to be “information”.

A subject concept comprehended by explicit/implicit pacts between authors and readers is the main instance of the objective position of such “about” phrases.

In the case of artistic writings, they do not necessarily give any information but give some emotional feelings.

Even an informative document does not necessarily regard a subject concept. A curriculum vitae, for example, gives some information about someone but does not regard any subject concept. Such functional documents work as information carriers according to the complex social/institutional protocols. A curriculum vitae gives information about the professional history of someone. The problem of information access here is split into 1) whose curriculum vitae it is and 2) if it is a curriculum vitae or not. A curriculum vitae of someone can be compared with that of someone else’s but also with the medical examination report of this person. Thus information is located in the lattice of syntagmatic and paradigmatic relations of semantics.

Information access problems against entity topics are in general the identification of the type and the entity in question, given the source of information as well as information needs.

In the topic relevance search, aboutness is comprehended as representation of subject concepts, while in the entry page finding task, aboutness is split into “entriness”(entry page or not) and entity correctness (which entity it is about?). Neither “entriness” nor entity correctness can be processed as the bag of word representation.

2. SYSTEM DESCRIPTION

For the TREC-2001 Web track experiments, we utilized the engine of Justsystem ConceptBase Search™ version 2.0 as the base system.

A dual Pentium III™ server (670MHz) running Windows NT™ server 4.0 with 1024MB memory and 136GB hard disk was used for experiments.

The document collections are indexed wholly automatically, and converted to inverted index files of terms.

2.1 Term Extraction

In order to compose possible noun phrases, queries and documents in target databases are analyzed by the same module that decomposes an input text stream into a word stream and parses it using simple linguistic rules.

Extracted units are single word nouns as well as simple linguistic noun phrases that consist of a sequence of nouns or nouns preceded by adjectives.

2.2 Vector Space Retrieval

Each document is represented as a vector of weighted terms by tf*idf in inverted index files and the query is converted in similar ways.

Similarity between vectors representing a query and documents are computed using the dot-product measure, and documents are ranked according to decreasing order of RSV.

OKAPI BM25 function is utilized as the TF part of weighting function [7] so that the retrieval process can be considered as probabilistic ranking.

2.3 Passage Retrieval

Since some pages are extremely long in the wt2g data set, we became aware that using passages rather than whole pages as the indexing unit is appropriate for the sake of retrieval effectiveness.

Passage delimiting is done such that each passage becomes a similar length rather than looking for thematic/discourse boundaries.

2.4 Phrasal Indexing and Weighting

Our approach consists of utilizing noun phrases extracted by linguistic processing as supplementary indexing terms in addition to single word terms contained in phrases. Phrases and constituent single terms are treated in the same way, both as independent terms, where the frequency of each term is counted independently based on its occurrence.

2.5 Pseudo-Relevance Feedback and Reference Database Feedback

Automatic feedback strategy using pseudo-relevant documents was adopted for automatic query expansion.

The system submits the first query generated automatically from topic descriptions against the target or reference databases, and considers the top n documents from the ranked list as relevant.

The term selection module extracts salient terms from these pseudo-relevant documents and adds them to the query vector.

Then the expanded query vector is submitted against the target databases again and the final relevance ranking is obtained.

The whole retrieval procedure is as follows:

1) Automatic initial query construction from the topic description
2) 1st pilot search submitted against the reference database
3) Term extraction from pseudo-relevant documents and feedback
4) 2nd pilot search submitted against the target database
5) Term extraction from pseudo-relevant documents and feedback
6) Final search to obtain the final results

2.6 Term Selection

Each term in the example documents is scored by some term frequency and document frequency based heuristics measures described in [4].

The terms thus scored are sorted in decreasing order of each score and cut off at a threshold determined empirically.

In effect, the following parameters in feedback procedures should be decided:

1) How many documents to be used for feedback?
2) Where to cut off ranked terms?
3) How to weight these additional terms?

These parameters are carefully adjusted using TREC-9 queries (topic 451-500), wt10g data set and the relevance judgement file provided by NIST. Parameter sets for official runs are calibrated so that the early precision rather than average precision is maximized.

2.7 Spell Variation

When the system finds non stop-word terms from the "title" field text of topic description, it is clear that no document is returned. In such a case, the initial queries are expanded automatically by generated spell variations.

The procedure consists of looking for similar words in the word lists extracted from the database. Spelling similarity is measured by a combination of uni-gram, bi-gram and tri-gram matching scores.

This query expansion was adopted originally for the TREC-9 Web track runs where the "title" field contained some spell errors.

2.8 Another source of "aboutness": Anchor Text of Hyperlinks

When we are asking what a page is talking about, sometimes anchor texts (or link texts, the texts on which a hyperlink is set) indicate an exact and very short answer.

The anchor text is typically an explanation or denotation of the page that it is linked to. Some commercial-based search engines utilize such information for advanced searches [1][2]. We treat anchor texts literally as the part of the linked document.

In total, 6,077,878 anchor texts are added to 1,173,189 linked pages out of 1,692,096 pages in the wt10g data set. So 69% of document pages in the data set are attributed to anchor text information on top of the original page information.

2.9 Link Structure Analysis

There seems to be a misunderstanding about the usage of pagerank[2] like popularity-based ranking that utilizes indirect-link information propagating rank values through hyper-link networks.

Such a ranking would not help the information seeking activities of individuals unless the individuals' information needs are strongly correlated with the popularity or the collection is heavily polluted by spam pages. The situation in navigation-oriented search seems to be the same as in the subject-oriented search. In order to show the effectiveness of popularity based ranking, information needs should be arranged according to the popularity.

Instead of the popularity-based ranking, we apply adequate link analysis according to the nature of the information seeking tasks behind the evaluation model.

2.10 Attribute-Value Basis Re-ranking

Our approach to the entry page finding task consists of combining the scoring results from different analysis procedures of pages. This is intended to rank the pages according to the following aspects:

"Entriness": the likelihood that the page is the entry point of a site.

Entity correctness: the likelihood that the page is about the entity indicated by the information need.

The following four types of analyses are processed:

-Bag of words analysis

This is mainly intended to gather candidate pages to be examined precisely hereafter.

The following three analyses are intended for rating both "entriness" and entity correctness.

-Link analysis

This examines the number of inter-server linked, inner-server linked and inner—server linker to rate "entriness" of the page.

-URL analysis

This examines URL form, length and names to rate both "entriness" and entity correctness.

-Text analysis

This examines title, inter/inner-server anchor texts and other page extracts to rate mainly entity correctness but also "entriness" by scored pattern matching.

Run tag	Index	RefTerms	Avg. Prec	R-Prec
jscbtawtl1	N	Strong	0.1890	0.2020
jscbtawtl2	N	Weak	0.1954	0.2150
jscbtawtl3	NVA	Strong	0.2003	0.2226
jscbtawtl4	NVA	Weak	0.2060	0.2308

Table 1: Performance of official runs

Through the experiments, we confirmed our expectation that only a small portion of each page is enough to be indexed for the entry page finding task. In fact, only 500 bytes of plain text including the title, the URL, anchor texts and beginning part of the page are indexed in view of the bag of word analysis.

3. WEB AD HOC EXPERIMENTS

We submitted four title-only automatic runs as follows:

jscbtawtl1: title only, link run with noun phrase indexing, more weight on reference terms

jscbtawtl2: title only, link run with noun phrase indexing, less weight on reference terms

jscbtawtl3: title only, link run with noun phrase, adjective and verb indexing, more weight on reference terms

jscbtawtl4: title only, link run with noun phrase, adjective

Run tag	words avg.	words min	words max	phrase avg.	phrase min	phrase max
jscbt9wcs1 Initial	2.1	0	5	0.7	0	3
jscbt9wcs1 Final	44.1	0	138	31.0	0	176
jscbtawtl1-2 Initial	2.38	0	5	0.56	0	2
jscbtawtl1-2 Final	80.4	0	184	34.86	0	114
jscbtawtl3-4 Initial	2.72	0	5	0.60	0	3
jscbtawtl3-4 Final	84.86	0	160	37.76	0	114

Table 2: Length of queries measured by number of single word terms and phrasal terms (without spell variation expansion)

and verb indexing, less weight on reference terms

As for the link usage, we adopted the “anchor text” of the hyperlink information as we did in TREC-9 [5].

Table 1 shows the performance of official runs and Table 2 shows the length of the queries utilized in each run.

Initial queries are very short (in average, 2.38-2.72 single word terms and 0.56-0.60 phrasal terms, maximum 5 single word terms and 3 phrasal terms , minimum 0 single word terms and 0 phrasal terms) and they do not contain enough terms.

Table 3 shows the performance comparison combining pseudo-relevance feedback and reference database feedback as well as with/without phrasal terms on the basis of jscbtawtl2 and jscbtawtl4 settings.

The automatic feedback procedure contributes to 16.1% to 18.3 % of consistent improvements in average precision in all cases.

The final queries contain 80.4-84.86 single word terms and 34.86-37.76 phrasal terms in average (maximum 184 single word terms and 114 phrasal terms, minimum 0 single word terms and 0 phrasal terms). Note that we added many more terms in the final queries than we did in TREC-9.

The improvement gained by the combination of pseudo-relevance feedback and reference database feedback is 21.4% for N index run and 20.9% for NAV index run. It is natural that N index runs where initial queries are shorter gained more from the feedback process. The improvement gain from combined feedback is larger than our TREC-9 experiments(17% in link runs). This is mainly caused by our approach to have taken more terms from feedback and promote some terms to the foreground.

In TREC-9, we explained our approach utilizing “foreground vs background” metaphor. In other words, foreground terms denote directly the subject concept of the information need while background terms connote the subject topic. If the weighting balance is changed in the query, the information need is also shifted.

In order to promote some terms to the foreground, we adopted a simple voting from two sources of feedback; one is the target collection and the other is the reference collection.

Doing such calibration, we intended to make the runs be early precision preferred rather than MAP preferred as our TREC-9 runs. Despite this, the official result showed that our system was still MAP and recall preferred in comparison with other systems.

Run description	Ref	PFB	AvgPrec	R-Prec
N index / SW + phrases (jscbtawtl2)	Yes	Yes	0.1954	0.2150
N index / SW + phrases	Yes	No	0.1730	0.2013
N index / SW + phrases	No	Yes	0.1903	0.2074
N index / SW + phrases	No	No	0.1609	0.1898
N index / Single words only	Yes	Yes	0.1854	0.2051
N index / Single words only	Yes	No	0.1685	0.1915
N index / Single words only	No	Yes	0.1837	0.2078
N index / Single words only	No	No	0.1537	0.1841
NVA index / SW + phrases (jscbtawtl4)	Yes	Yes	0.2060	0.2308
NVA index / SW + phrases	Yes	No	0.1824	0.2106
NVA index / SW + phrases	No	Yes	0.1979	0.2417
NVA index / SW + phrases	No	No	0.1704	0.2149
NVA index / Single words only	Yes	Yes	0.1997	0.2357
NVA index / Single words only	Yes	No	0.1745	0.2083
NVA index / Single words only	No	Yes	0.1894	0.2217
NVA index / Single words only	No	No	0.1641	0.2062

Table 3: Performance comparison (Title only, jscbtawtl2-4 parameter set)

Supplemental phrasal indexing runs perform better in average precision as well as in R-precision both with/without pseudo-relevance feedback and with/without reference database feedback. The situation observed here is consistent with our experience in TREC-9 web track experiments, but in this case, the effectiveness of phrasal indexing seems to be more stable.

4. ENTRY PAGE FINDING EXPERIMENTS

As table 4 shows, we submitted four entry page search runs: jscbtawep1, jscbtawep2, jscbtawep3 and jscbtawep4.

These four runs adopt essentially the same configuration but differ in two parameters of final scoring.

The full phrase match bonus weights and the bag of word analysis weights are changed as shown in table 4.

Run tag	full-match	bow wght	MRR	Top10 %	NF%
jscbtawep1	moder	low	0.754	83.4	9.0
jscbtawep2	moder	med	0.769	83.4	9.0
jscbtawep3	high	med	0.752	83.4	9.0
jscbtawep4	moder	high	0.746	83.4	8.3

Table 4: Performance of official runs of the Entry Page Finding Task

4.1 The Server Database and the Whole Database

The server database contains 11680 server pages in the wt10g collection.

In fact, this covers 78% of 100 pre-test queries, i.e., this database contains at least one answer page against each of 78 queries out of 100 queries. It also covers 66.2% of 145 test queries.

Ten pages from the server database and 1000 pages from the whole database are merged in the manner that the 10 pages from the server database come to the top of the rank.

Thus far, we applied normal retrieval processing, utilizing bag of word queries.

MRR of the 10 page ranked lists against the server database accounts for 0.6409 and that of the 1000 page ranked lists against the whole database accounts for 0.4176.

Merging them together makes MRR of 0.6462.

4.2 Attribute-value Basis Re-ranking

Thus obtained ranked page lists of 1010 pages are cut off at the top 200 pages and re-ranked by the attribute-value basis analysis modules.

4.3 Basic Text Matching and Scoring in view of "Entity Correctness"

Text fields are scored by the matching procedure that accumulates each word matching point and adjacency point.

Such analysis is much more powerful than bag of word analysis and is equivalent to the full sub-phrase indexing against all long phrases.

4.4 Augmented Text Matching and Scoring in view of "Entity Correctness"

It is likely that the URL text contains the entity name as the part of the server name or the directory names.

But it is sometimes the case that the constituent words are agglutinated. The matching is augmented in order to treat such agglutinated names.

4.5 Supplemented Text Matching and Scoring in view of "Entriness"

Some field are matched against pre-coded patterns as follows:

The "InterServerAnchorText" field is intended to be matched with anchor texts like "go to the homepage of XXX".

The "InnerServerAnchorText" field is expected to be matched with "back to the home(of XXX)".

The "Title" field is something like "Welcome to the homepage of XXX".

4.6 Link Analysis

The number of interserver linked, normalized by maximum number of interserver linked, among the candidate pages simply indicates the "entriness" of the page.

The entry page is also very likely to have at least one linker to the inner server pages unless his/her/their/its web site consists of only one page.

4.7 Score Composition

The final score is computed as the sum of the weighted scores from each analysis. Each analysis weight is calibrated by the 100 pre-test topics.

$$
\begin{aligned}
PageScore = {} & w_1 S(BOW) + w_2 S(URLType) + w_3 S(URLText) \\
& + w_4 S(InterServerAnchor) + w_5 S(InnerServerAnchor) + w_6 S(Title) \\
& + w_7 S(Larg\ eFonts) + w_8 S(FullMatch) + w_9 S(InterServerLinked) \\
& + w_{10} S(InnerServerLinker\ ) \qquad (1)
\end{aligned}
$$

The full phrase match bonus is added only when all the constituent words of the entity name matches and prevents inclining to partial matching in many fields rather than full matching in one field.

After such re-ranking processes, the final results of MRR 0.746 to 0.769 are obtained.

5. DISTRIBUTED RETRIEVAL AGAINST WT10g

In view of the trade-offs between efficiency and effectiveness, there might be two possibilities for large collection retrieval.

1)Centralized Multi-stage Search

All the units are indexed in a system and first some important parts of each document like title and large font text parts are searched. If the user is not satisfied with the first results or he/she requests an exhaustive search through the collection, the second search looks through all the text part of the documents.

2)Distributed Selective Search

The collection is partitioned by some criteria like publication date order, author's name order, original document location or content basis classification, etc., and stored into separate databases. The search process consists of 1) selecting databases to be searched, 2) distributed search in all the databases selected, 3)fusion of the result lists from the selected databases, and 4) if the user requests it, the search result lists from all the databases are presented.

Many studies on distributed retrieval have been done by researchers of the IR society, but so far, web commercial search engines tend to be implemented as centralized search systems. The problem in distributed IR is the database selection; failing to properly select the target databases causes severe degradation in effectiveness. However, some studies claim that the effectiveness of a distributed search is even better than a centralized search when an adequate selection algorithm is applied[6].

5.1 Collection Partitioning

WT10G collection is partitioned in two ways.

5.1.1 104 Pre-defined directory Partitioning

Each of 104 directories(WTX001 – WTX104) of distribution CD-R is utilized as a single database.

Each database is almost the same size. Each database contains about 10,000 to 20,000 pages and these sizes account for 60 to 80MB in text file.

5.1.2 326 Category Partitioning

Content basis classification has been done using 326 categories derived from the Yahoo US categories.

The highest two level categories of Yahoo US[8] directories were adopted and Web pages linked from them were downloaded in March 2001. These pages (142MB, 19048pages) are stored in the classifier database and each page in WT10G is submitted as a query against this classifier database. Scores of the best 15 ranked (Yahoo linked) pages are voted for the category from which the (yahoo linked) page is linked. Thus, for each page in WT10G, the category is decided and the WT10G pages are stored in partitioned databases.

Average	5190.479
Standard Error	836.2663
Median	492
Standard deviation	15099.18
Distribution	2.28E+08
Kurt	49.18961
Skew	6.136924
Range	162594
Min	1
Max	162595
Sum	1692096
Number of Samples	326

Table 5: Basic Statistics of number of pages in each database of 326 category partitioning

In this case, the database size is diverse, ranging from as small as only one page to the maximum 162595 pages (9.6% of the whole collection). Basic statistics measures of the number of pages in each database are shown in table 5.

5.2 Database Selection

The following three algorithms for selecting databases are examined.

5.2.1 CORI

The formula proposed in [3] is adopted.

$$T = d\_t + (1 - d\_t)\frac{df}{df + K} \quad (2.1)$$

$$K = k((1-b) + b\frac{cw}{mean(cw)} \quad (2.2)$$

$$I = \frac{\log(\frac{|C|+0.5}{CF})}{\log(|C|+1.0)} \quad (2.3)$$

$$p(t|c) = d\_b + (1 - d\_b) * T * I \quad (2.4)$$

d_t, d_b : 0.4

cw : number of words in a database

df : document frequency of the term tin the collection c

CF : number of collections where the term t appears

|C| : number of collections

p(t|c) is the weight of the term t against the collection c and the each database is ranked by the sum of this weight over all query terms. We utilized the setting of k=200 and b=0.8.

*5.2.2 Simple DF*ICF*

This is simplest version of DF*ICF, an essential part of the CORI method.

$$DF = d\_t + (1 - d\_t)\frac{df}{MAX\_df} \quad (3.1)$$

$$ICF = \log(\frac{|C|}{CF}) \quad (3.2)$$

$$p(t|c) = DF * ICF \quad (3.3)$$

d_t : 0.5

df : document frequency of the term tin the collection c

MAX df : maximum df of the term through collections

CF : number of collections where the term t appears

|C| : number of collections

*5.2.3 DF*AVG-IDF*

DF*AVG-IDF is similar to DF*ICF but instead of ICF, average IDF of the term over all the databases is utilized.

5.3 Experiments

We compared two collection partitioning and three database selection methods. Figure 1 in Appendix A. shows a comparison of the combination of two partitioning and three database selection methods by using MAP, R-precision, precision at 20 docs (PREC@20) and the number of relevant documents retrieved (REL_RET). For each of the six combinations, we examined 10 runs, decreasing the number of databases to be searched from 100% down to 10% by 10% of the whole collection. For each of topic 501 to 550, the databases were selected from the top n % of the ranked database list utilizing one out of three methods. No feedback is applied in these experiments. Once the databases to be searched are decided, statistics from each database selected are gathered so that a centralized search against the whole selected database is simulated. Consequently, the problem of result fusion is excluded in these experiments.

Because of the essential similarity of the method, CORI and simple DF*ICF perform very similarly even though CORI seems to perform better in MAP and REL_RET.

After examining each database selected, we noticed that CORI tends to select larger databases than other methods. In fact, when selecting 10% databases, CORI searched 39% of the pages while TF*ICF searched 20% of pages (See Figure 2 in Appendix A.).

Content basis partitioned databases perform clearly better, especially when the portion of the collection to be searched is reduced. The most notable thing is that using a combination of content basis partitioned databases and CORI or DF*ICF, the early precision(PREC@20) is even getting better when reducing the number of databases.

DF*ICF especially marked the best PREC@20 when searching only 41% of pages out of the whole collection(20% by database numbers).

6. CONCLUSIONS

TREC-2001 Web track evaluation experiments at Justsystem group are described.

The following conclusions are drawn from these experiments:

1)We modified our TREC-9 approach, i.e., longer vectors with background down-weighting and promoting some terms to the foreground, seem to perform well.

2)A three stage approach, i.e., bag of word analyses, result fusion and attribute-value basis re-ranking, is successfully applied to the entry page finding task.

3)A distributed selective search performs better than a centralized search in early precision when an adequate database selection method and collection partitioning are applied.

4)A simple DF*ICF database selection method performs as well as the CORI method.

5)A distributed selective search performs better with content basis category partitioning of the collection than (near) random partitioning.

6)Distributed selective search is possibly a good option in early precision preferred retrieval tasks against very large collections.

In future work, we will examine better partitioning methods by equalizing the number of pages in each database of content basis category partitioning.

REFERENCES

[1] Altavista: http://doc.altavista.com/adv_search/ast_ma_clickhere.html

[2] Brin, S. and Page, L. , The Anatomy of a Large-Scale Hypertextual Web Search Engine, in Proceedings of the Seventh International World Wide Web Conference, 1998, 107-118.

[3] Callan, J.P., Lu, Z. and Croft, W.B., Searching Distributed Collections With Inference Networks, in Proceedings of the 18th Annual International ACM SIGIR Conference, Seattle Washington, 1995, 21-28.

[4] Evans, D.A. and Lefferts, R.G., Grefenstette, G., Handerson, S.K., Hersh, W.R., and Archbold, A.A., CLARIT TREC Design, Experiments and Results, in Proceedings of the First Text REtrieval Conference(TREC-1), NIST Special Publication 500-207, Washington D.C., 1993, 494-501.

[5] Fujita, S. , Reflections on "Aboutness"—TREC-9 Evaluaton Experiments at Justsystem ,in the Notebook version of the Ninth Text REtrieval Conference(TREC-9), Gaithersburg MD, 2000.

[6] Powell, A.L., French, J.C., Callan, J., Connell, M., and Viles, C.L., The impact of Database Selection on Distributed Searching, in Proceedings of the 23rd Annual International ACM SIGIR Conference, Athens Greece, 2000, 232-239.

[7] Robertson, S.E., Walker S., Jones S., Hancock-Beaulieu, M.M., Gatford, M. Okapi at TREC-3, in Proceedings of the Third Text REtrieval Conference(TREC-3), NIST Special Publication 500-225, Washington D.C., 1995, 109-126.

[8] Yahoo: http://www.yahoo.com/

Appendix A. Database Selection Experiments

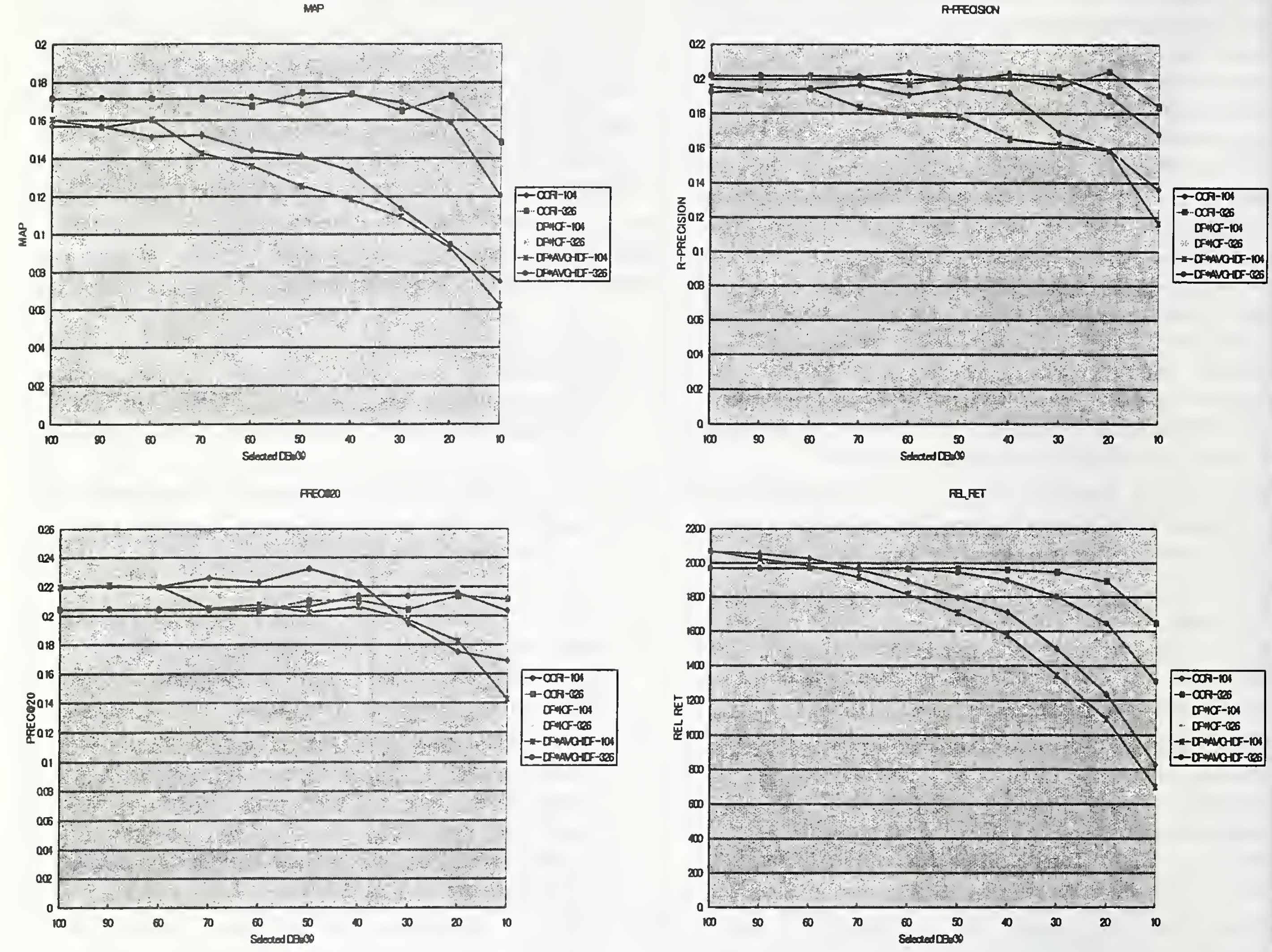


Figure1: Performance of DB Selection Runs with topic 501-550

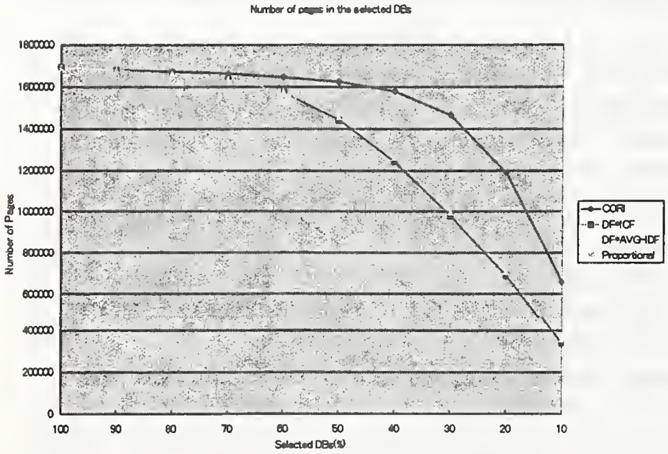


Figure2: Page numbers in the selected DBs of 326 partitioning runs

Kasetsart University TREC-10 Experiments

P. Norasetsathaporn and A. Rungsawang
{g4365020,fenganr}@ku.ac.th
Massive Information & Knowledge Engineering
Department of Computer Engineering
Faculty of Engineering
Kasetsart University, Bangkok, Thailand.

Abstract

We use WORMS, our experimental text retrieval engine, in our TREC-10 experiments this year. The Okapi weighting scheme is used to index and test in both web ad hoc and homepage finding tasks. In web ad hoc task, we propose a novel approach to improve the retrieval performance using text categorization. Our approach is based on an assumption that "Most relevant documents are in the same categories as their query". This retrieval approach has a great effectiveness in reducing the number of searching documents and the searching time. In addition, it can be used to improve precision of retrieval. In homepage finding task, we use a query expansion based method and the Google search engine to find more useful words to be added to the topics. Before using with homepage finding task, we test the query expansion method with the homepage finding training set to get the best type of expanded query.

1 Introduction

In our TREC-9 experiments last year [11], we modified the Cornell's SMART version 11.0, in addition with the notable pivoted unique normalization weighting scheme [3], to run smoothly on our Linux machine. However, we confronted with intrinsic operating system problem. We could not index the whole web track collection in one-shot. We therefore decided to split the web track collection into sub-collections. We indexed and tested all sub-collections separately and merged all subsequence results to get the final top-1000 scores to send to NIST.

In the TREC-10 experiments this year, we use WORMS [15], our own text retrieval engine, to index and experiment instead of using the Cornell's SMART version 11.0. WORMS can eliminate the intrinsic Linux operating system problem because it does not create inverted file image that is larger than 2 GB, as it can split the inverted file image into several smaller ones. We also implement the notable Okapi weighting scheme [13, 14] in WORMS and use it to index and test both web ad hoc and homepage finding tasks. In the web ad hoc task, we propose a novel approach to improve the retrieval performance using text categorization. We formulate an assumption and set the experiments to test that assumption. We study the relation between the relevant documents for a query and its categories using TREC10 as the test collection and Google Web Directory as the knowledge base for the SVM classifier [5, 8, 12]. This retrieval approach provides a great effectiveness in reducing the number of searching documents and the searching time. In addition, it can be used to improve precision of retrieval by re-ranking method. In homepage finding task, we use a query expansion technique to add more useful words to topics. Before we use it with homepage finding task, we tested the method with homepage finding training set to get the best type of expanded query.

This paper is organized in following way. Section 2 introduces shortly our information retrieval engine. Section 3 gives more detail what we do in web ad hoc task and about the experimental results. Section 4 describes the query expansion method we use in homepage finding task and the experimental results we obtain. Section 5 concludes this paper.

2 Information Retrieval Engine

An experimental information retrieval engine, WORMS [15] is a high-performance text retrieval system implemented using the PVM message-passing library, and running on the cluster of low-cost PC based machines. WORMS is based on the vector space indexing model and the inverted file structure for effective and fast retrieval. There are actually 3 prototypes of WORMS. The first one is the sequential WORM prototype that we use in this TREC10 experiments. The two others are the parallel WORMS and the high performance WORMS that will not be mentioned here. The 4 main components of the sequential WORM are Stop&Stem, Indexer, Invfile, and Retriever, respectively. WORM's indexer can read user's input parameter and create several chunks of document vector images and its corresponding inverted files, of which each file's size is less than 2 GB barriers of the Linux x86 operating system.

Okapi Weighting Scheme

We found during our TREC10 experiments that the pivoted normalized weighting scheme [3] we used in TREC-9 experiments last year are not as good as we expect. In case of the TREC-9 data, weighting it with Okapi's byte length normalization gives better retrieval effectiveness. Therefore, we use the Okapi's byte length normalization which is base on the 2-Possion distribution model [13, 14] in TREC-10 this year. The following weighting scheme is implemented in WORMS.

The o weight implemented in WORMS:

$$L(n,i) = \frac{f_{ni}}{2 \times (0.25 + 0.75 \times \frac{dl}{avg\_dl}) + f_{ni}} \quad (1)$$

The p weight implemented in WORMS:

$$\log \frac{N - df_i}{df_i} \quad (2)$$

where dl is the document length, avg_dl is the average document length, f_{ni} is the raw tf-factor, N is the total number of documents in the collection and df_i is the number of documents which a term i appears.

3 Web Ad Hoc Task

We perform 2 experiments for this task. First, we index and test the small web track collection (WT10g) using WORM and Okapi weighting scheme. Second, we use text categorization technique to increase the retrieval effectiveness. The rest of this section gives more detail of our propose text categorization technique.

3.1 Using Text Categorization

Owing to the incessant growth of the Internet and the abundant availability of text data in the World Wide Web, text classification is acquiring more popularity with the growing interest. Many search engines, such as Google[1], Yahoo[6] and AltaVista[7], provide data that has been grouped into categories or directories. In addition to managing data, text categorization is a great benefit in our research. We use text categorization to improve retrieval performance of web ad hoc task. Our method bases on the assumption that "Most relevant documents are in the same categories as their query". To evaluating our assumption, we have to study the relation between the relevant documents for a query and its categories.

The problem of this experiment is how we classify WT10g documents and queries into hierarchical structure. To solve this problem, the machine learning technique called "Support Vector Machine" (SVM) [5, 12] is used to automatically construct classifier by using the existing web directory of the Google, designed for human Web browsing, as the source of knowledge base for training and testing process. After training step, we obtain the top and the second level classifiers, one classifier for one category, trained by the Google Web documents. Following that classifiers, we classify WT10g collection and the topics number 501-550 into categories by following the Google Web Directory structure. Then we study the relation between the relevant documents for a topic and its corresponding category to test our assumption and use it to improve retrieval effectiveness.

Support Vector Machine

Support Vector Machines (SVM) [5, 12] is a relatively new learning approach, introduced by Vapnik in 1995, for solving the two-class pattern recognition problems. It is based on the Structural Risk Minimization principle for which error-bound analysis has been theoretically motivated. The method is defined over the vector space. The decision surface separates the data points into two classes. In this research, we employ the Joachim's implementation, SVM^{light} [8, 9], owing to its accuracy and effectiveness.

Google Web Directory

Google Web directory integrates search technology with Open Directory pages to create the most useful tool for finding information on the Internet. The Open Directory Project[2] is claimed to be the largest, most comprehensive human-edited directory of the Web database. It is constructed and maintained by a vast, global community of volunteer editors. The Google directory contains over 1.5 millions URLs. These URLs are organized in Google Web documents which connected together. The more general the category, the closer to the root of the tree it is. The connection of Google documents are quite complex. There are some category names linked to other Google categories, classified under a different path of the Google hierarchy.

Pre-processing Phase

For TREC10, we parse all html tags, images, all messy data and the others, out of the WT10g collection. We also remove stop-word and stem the rest [4] from the WT10g collection. In case of web ad hoc topics, we also remove stop-word and stem the rest to build queries. After that, WT10g documents are weighted with *nno* weighting scheme and queries are weighted with *npn* weighting scheme.

For Google collection, we create the local copy of Google Web directory by using Wget [1], a GNU network utility for retrieving files from the WWW. The document is stored in a hierarchy structure by Wget itself. From documents which contains hyperlink information between them, we build the same hierarchical structure as the Google Web directory. We have 2 methods for managing documents into categories. First, we choose only documents that are not cross-linked to other categories. Second, we choose all documents under the category, including those in their cross-links. In this paper, we choose the first method because most of the documents in Google hierarchy belong to many cross-linked categories. For this reason, each category content has not been obviously separated. When we classify queries into categories, most of the queries belong to many categories as well. There are 15 top-level and 431 second-level categories we choose to experiment. We do not include World top-level categories because most web pages in this category is not written in English. Then we remove all HTML tags from documents and extracted "entry title", the title of a categorized entry indexed in a category, and "entry summary", the brief textual description of an entry [10]. We also reduce the number of features by removing words contained in the stop word list and stemming the rest [4].

The feature selection methods we used to reduce a high dimensional feature space in Google Web Directory is Document frequency thresholding (DF) [17]. Because of its simplicity, effectiveness, and low cost in computation, DF was choosen. Document frequency is the number of documents in which a term occurs. We compute the document frequency for each unique term in the Google documents and remove the term with DF below a threshold (DF = 5) from the feature space.

Building Classifier

We build a classifier for each category in both levels. In each category, a training set and a testing set are chosen by applying the systematic random selection to the documents. For the top-level category, we select the same number of documents in each of the top sub-categories by choosing every n documents, where n calculated from dividing the number of documents in each sub-category by the number of documents we want. With this method, we obtain sample documents throughout sub-categories in the hierarchy. In the second-level, we use the same method but the training and testing examples only come from documents in the same top-level category. These sets are positive examples of the category. The negative examples are selected from the other categories. After the selection process, we obtain training and testing document sets for every category. The number of training and testing Google Web documents in the top-level and the second-level are shown in Table 1 and Table 2, respectively.

Category Name	Total	Training		Testing
		P	N	P+N
Arts	244232	39932	77436	98912
Business	194269	57052	66765	89351
Computers	104371	33454	76155	91117
Games	42984	13989	79255	86600
Health	51164	28161	76716	87311
Home	33578	24163	77822	87221
Kids and Teens	12790	12790	78790	79091
News	48311	48337	76116	76182
Recreation	99150	45136	75715	96773
Reference	80513	48967	73782	73532
Regional	654467	9121	80904	90284
Science	75738	15132	79656	92201
Shopping	102067	73385	72101	94066
Society	183309	77882	45146	104355
Sports	74341	17092	79314	89360

Table 1: The number of Training and Testing Web documents in the top-level. (P=Positive, N=Negative)

Category Name	Total	Training		Testing
		P	N	P+N
Arts.Animation	16068	717	5962	138
Business.Accounting	1243	767	13854	105
Computers.Graphics	1568	1523	5621	138
Games.Gambling	3048	1797	1905	109
Health.Fitness	1118	963	4630	108
Home.Gardens	3178	2124	2927	120
News.Media	685	633	5945	83

Table 2: Training and Testing Samples in the second-level. (P=Positive, N=Negative)

Then, Google training data were weighted by *nno* weighting schemes and Google testing data were weighted by *npn* weighting schemes. Finally, we create top-level classifiers and second-level classifiers from training sets we provided.

Level	miR	miP	miF_1	maF_1	error
top	0.731	0.739	0.735	0.710	0.078
second	0.752	0.730	0.741	0.672	0.117

Table 3: Performance of classifiers in the top-level and the second-level using SVM classifiers.

[1] http://www.lns.cornell.edu/public/COMP/info/wget

Category Name	Topic number
Arts	505, 510, 527, 534, 545
Business	502, 514, 538, 542
Computers	501
Games	-
Health	504, 508, 509, 511, 524, 532, 537, 539, 540, 542, 543, 544, 549
Home	507, 514, 548
Kids and Teens	512, 515, 519, 525, 527, 539, 549
News	541
Recreation	503, 507, 519, 521, 529, 535, 547, 549
Reference	502, 547
Regional	-
Science	501, 502, 504, 513, 519, 522, 525, 528, 530, 542, 549, 550
Shopping	507, 508, 509, 511, 517, 519, 520, 522, 530, 532, 537, 539, 548, 549
Society	509, 510, 516, 517, 519, 520, 529, 534, 544, 545
Sports	506, 509, 548

Table 4: The categories of Topics.

Evaluating classifiers

After building the classifier in the training process, we obtain 15 top-level classifiers and 431 second-level classifiers. Then we test both levels classifiers with provided testing sets. To evaluate the performance of classifiers, we use the standard precision (P), recall (R) and F_1 measures. Precision is the ratio of correct assignments by the system to the total number of the system's assignments. Recall is defined to be the ratio of correct assignments by the system to the total number of correct assignments. The F_1 measures equally weights between precision and recall in the following form :

$$F_1(R, P) = \frac{2RP}{R + P} \quad (3)$$

This measure can be computed in 2 ways, micro-averaging and macro-averaging. Micro-averaging can be calculated from global average in all categories. Macro-average can be calculated from individual calculation in each category first, and then average over categories. Table 3 shows the accuracy of our both top and second level classifiers.

Applying the classifiers with TREC10

We use the top-level classifiers to classify the WT10g collection and its 50 queries into 15 top-level categories: Arts, Business, Computers, Games, Health, Home, Kids_and_Teens, News, Recreation, Reference, Regional, Science, Shopping, Society and Sports. Some documents and topics cannot be classified in any category, while some can be classified in one or more categories. Categories of topics are summarized in Table 4. There are 7 topics that cannot be contained in any category i.e. 518, 523, 526, 531, 533, 536 and 546. Then we pass the result to the second-level classifier to classify them into 431 second-level categories. The documents that pass the top-level classifier in each category are classified by the second-level classifiers of that category. Table 5 shows an example of the result. Topic number 507 is in category Home, Recreation, and Shopping within the top-level. After being classified by the second-level classifier, topic 507 resides in Home&Customer_Information, Home&Homeowners, Recreation&Autos, Recreation&Climbing, Shopping&Office_Products, Shopping&Tobacco, and Shopping&Vehicles, which are still in the sub-category of Home, Recreation, and Shopping. We show the amount of documents in the top and the second level category in Figure 1 and 2 respectively. The X axis in figure 1 represents the 15 top-level categories ranging from Arts to Sports, while the X axis in figure 2 represents all 431 sub-categories in the second-level.

Topic number	Category of topic
507	Home.Consumer_Information Home.Homeowners Recreation.Autos Recreation.Climbing Shopping.Office_Products Shopping.Tobacco Shopping.Vehicles

Table 5: The second-level category of topic 507

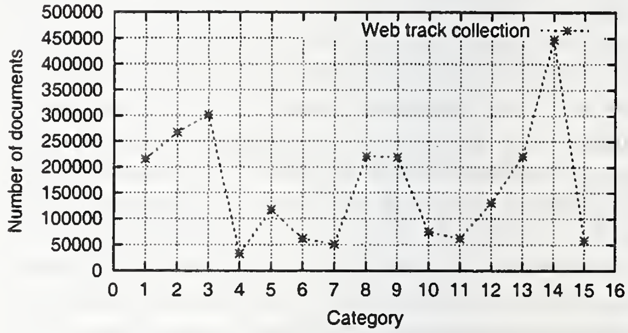


Figure 1: The number of documents in the top-level category.

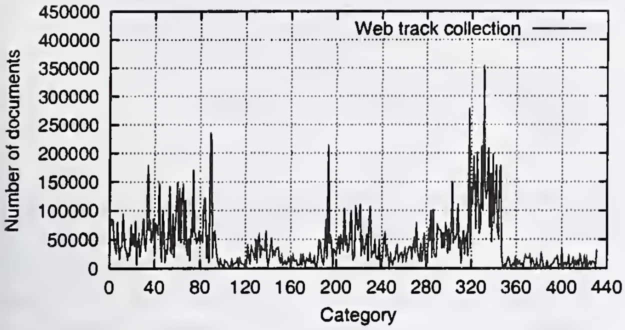


Figure 2: The number of documents in the second-level category.

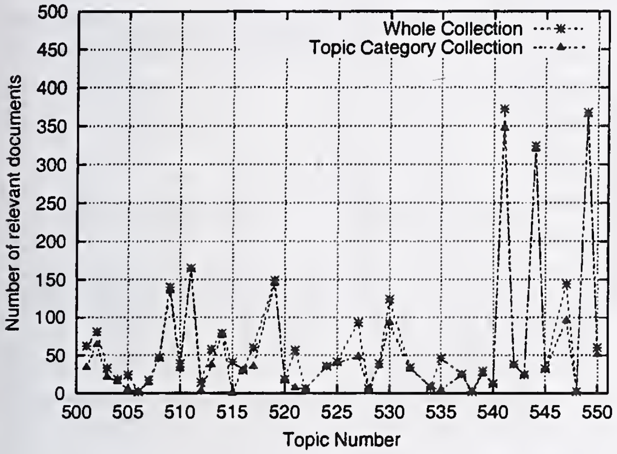


Figure 3: Comparison between relevant documents in the whole collection and the top-level topic category collection.

3.2 Experimental Setup and Results

We set the experiments into 3 phases: evaluating the assumption, searching in topic category collection and re-ranking score.

Phase I: Evaluating the assumption

In this sub-section we evaluate our assumption that "Most relevant documents are in the same categories as their query", by examining the number of relevant documents in topic category collection. Topic category collection of each query is defined as a set of documents that is in the same categories as its query.

Figure 3 provides the comparison between the total amount of relevant documents that can be retrieved from the whole collection and those from the top-level topic category collection. The X axis is the topic number. The Y axis is the number of relevant documents. For example, for the topic number 541, there are totally 372 relevant documents in the whole collection, and 347 relevant documents in the top-level topic category collection. Figure 4 shows the percentage of relevant documents and the percentage of the number of documents for each topic in the top-level topic category collection. For example, for the topic number 540, there are 100% of the total relevant documents and 6.93% of the whole documents being categorized in the top-level topic category collection. Figure 5 shows the comparison between the number of relevant documents for each topic in the top-level topic category collection and the second-level topic category collection, while figure 6 shows the comparison of the number of documents being categorized in both levels.

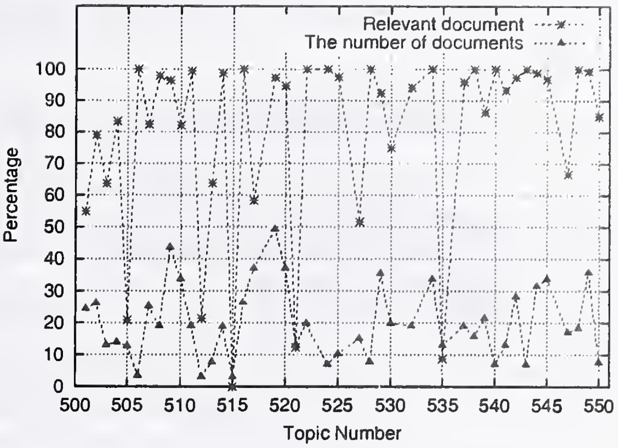


Figure 4: Percentage of relevant documents and number of documents in the top-level topic category collection.

As we can see from figure 3 and 4, most relevant documents are categorized in their corresponding topic category collection. There are 12 topics that have the number of relevant documents in the topic category collection less than 80 percents. Moreover, the size of the topic category collection for each topic is less than 50 percents of the whole collection. This shows that our assumption is correct for most of the relevant documents and their corresponding topics. From figure 5 and 6, relevant documents in the top-level topic category collection are also found in the second-level topic category collection, while the number of documents in the second-level topic category collection decrease about 30 percents on average. This shows that our assumption in the second-level is still correct.

However, there are some topics in both levels that do not follow this assumption. The cause may be come from the inefficiency of the classifier we use. There are some topics and documents that are classified to the wrong category. For this reason the result does not as good as it should be.

Experiment	Relevant		Average per Query		Average Precision
	Doc	Ret	Searched Documents	Searching time (sec)	
Retrieve from whole documents	3363	2247	1692096 (100%)	20.5067	0.2088
Retrieve from top-level topic category collection	3363	2132	530332 (31.34%)	7.7179 (37.63%)	0.2032 (−2.7%)
Retrieve from second-level topic category collection	3363	2030	450912 (26.65%)	6.8697 (33.50%)	0.1997 (−4.4%)
Score Re-ranking	3363	2300	1692096 (100%)	-	0.2163 (+3.6%)

Table 6: Average precision concluded from the experiments (50 Queries).

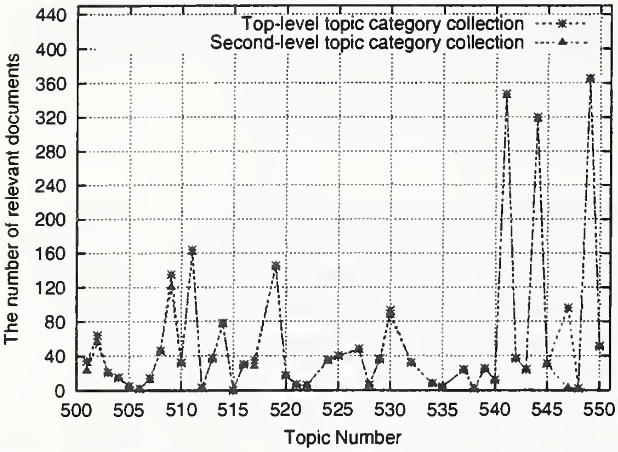


Figure 5: Comparison between relevant documents in the top-level and the second-level topic category collection.

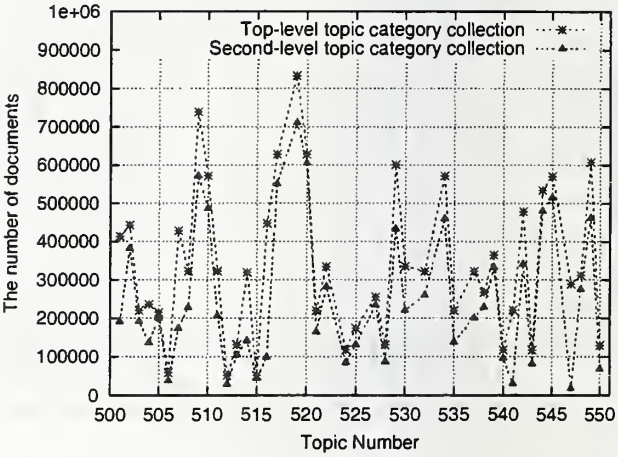


Figure 6: Comparison between the number of documents in the top-level and the second-level topic category collection.

Phase II: Searching in topic category collection

In this phase, we study the average precision of searching in both levels of topic category collection. We retrieve top 1000 documents for each query. Table 6 show the result of this phase comparing with the searching result from the whole collection. From Table 6, the average precision (0.2032) from searching only in the top-level topic category collection is almost the same as the average precision obtaining from searching in the whole collection (0.2088), while the number of searching documents and the searching time are only 31.34% and 37.63% of the whole. In case of the second-level, the average precision (0.1997) decreases a little bit, while the number of searching documents and the searching time are only 26.65% and 33.50% of the whole.

Phase III: Re-ranking Score

The experiments from re-ranking phase use the prediction value from the SVM classifiers. We retrieve 5000 documents for each query and re-rank the score of each retrieved document depending on its prediction value using the re-ranking equation(5) below and then we keep top 1000 documents for evaluation. The parameters of the equation are the prediction value, the constants, and the original score. In current experiments we can increase the precision a little bit. The result shows that our re-ranking equation does not take much effect so we must still look for better parameters of equation. For this experiment we consider only the top-level topic category documents. The re-ranking equation is:

$$Newscore = orig\_score + 0.5 \times orig\_score \times i \quad (4)$$

$$i = \frac{predict\_value}{2.75}; (i > 1) \rightarrow (i = 1) \quad (5)$$

The prediction value is calculated as the following. For example, query number 1 is in category A and B with prediction value 0.1 and 0.3. Document 1 is in category A, B, and C. Document 2 is in category A and C. Document 3 is in category C. The prediction

Experiment	Relevant document	Relevant retrieved	Average precision
HomePage Finding Task with Query Expansion	252	120	0.1902

Table 7: Final result of HomePage Finding Task with Query Expansion.

value in query 1 of document 1 is 0.4, document 2 is 0.1 and document 3 is 0. The number of 0.5 and 2.75 in equation (4) and (5) are concluded from the experiments, 0.5 controls the max increasing score, 2.75 is the reference value. If our prediction value exceeds the reference value, the total point (half of original score) will be increased.

Table 6 shows that re-ranking method can increase the fraction of precision from 0.2088 to 0.2163 (+3.6%). However, this re-ranking method is in preliminary step, we are looking for better re-ranking equation to get more retrieval accuracy.

4 Homepage Finding Task

Since topics in homepage finding task have very few words, we then apply an query expansion technique [16] to add some more useful words to them. We first remove stop words and stem the rest of the homepage finding topics and use them as our unexpanded queries. Our query expansion method has been divided into 3 steps described in the next paragraph. After these steps, we obtain 5 types of queries: I, II, III, IV and V, respectively. The query expansion type which gives the best result for homepage finding training set is chosen to expand the homepage finding topics.

Step I, we send the unexpanded queries to the Google search engine and keep the top 20 search results. If the query length is more than 10 words, we break that query into multiple word-segments (each segment has at least 9 words), send every word-segment to the Google search engine and keep the top 20 search results of all word-segments. We then intersect each search result of the word-segments to obtain the final 20 search results.

Step II, we remove stop words and stem the rest of the search results from step I and find document frequency of all words. If document frequency of any word is more than 5, we will use that words to expand the query to be searched in the homepage finding collection. This query is called Type I query. For those words whose document frequency are less than or equal to 5, they will be sorted by their document frequency, the first highest document frequency is used to build the Type II query, the second highest document frequency is used to build the Type III query, and so on.

Step III, for the Type II query, we add the first highest document frequency word that we obtain from step II above to the unexpanded query and resend that query to the Google search engine to get the top 20 search results. For these 20 search results, we repeat the step II, i.e. remove stop words and stem the rest and keep only words whose document frequency are more than 5 to add to the unexpanded query to build the Type II query. For the Type III to Type V query we repeat the same process as we do for the Type II query described in the beginning of this paragraph.

After we tested 5 Types of the expanded queries with homepage finding training set, we have found that the Type IV query provided the best result. We then apply the Type IV query to expand all the homepage finding queries, and use those expanded queries to search in the homepage finding collection. Table 7 concludes the final results.

5 Conclusion

In our Kasetsart TREC-10 experiments this year, we change indexing and testing tool from SMART version 11.0 to WORMS, our own retrieval engine, and use the Okapi weighting scheme in both web ad hoc and homepage finding tasks instead of the pivoted length normalized weighting scheme that we used in TREC-9 experiments last year. We also propose a novel retrieval approach using text categorization to improve retrieval effectiveness of the TREC-10 web ad hoc task. After we re-rank the resulting score, we get a little bit of precision increased. By searching relevant documents only in the top-level topic category collection, we obtain as good average precision as searching in the whole document collection, while the number of searching documents and the searching time reduce to 31.34% and 37.63% of the whole.

In the homepage finding task, we apply a query expansion method and using the Google search engine to find more words to be added into the homepage finding topics. We found that the result is quite promising.

Acknowledgement

We would like to thank all MIKE staffs, especially Hong, Wit, A, Banana, for their programming support and working spirit. We also thank Mr. Somsak Sriprayoonsakul from the Parallel Research Group for good support in our work.

References

[1] Google search engine. http://www.google.com.

[2] Open directory project. http://dmoz.org/.

[3] C. B. A. Singhal and M. Mitra. Pivoted document length normalization. In *Proceedings of the 19th ACM-SIGIR Conference. ACM Press*, pages 412–420, 1996.

[4] R. Baeza-Yates and B. Riberiro-neto. *Modern Information Retrieval*, chapter 7, pages 167–168. Addison-Wesley, 1999.

[5] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[6] D. Filo and J. Yang. Yahoo home page. http://www.yahoo.com.

[7] A. Inc. Altavista search index. http://www.altavista.digital.com.

[8] T. Joachims. Making large-scale svm learning practical. *In Advances in Kernel Methods. MIT Press*, 1998.

[9] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. *European Conference on Machine Learning (ECML)*, 1998.

[10] Y. Labrou and T. Finin. Yahoo! as an ontology - using yahoo! categories to describe documents to describe documents. *In Proceedings of CIKM'99*, pages 180–187, Oct. 1999.

[11] P. Norasetsathaporn and A. Rungsawang. Kasetsart university trec-9 experiments. *In The Ninth Text Retrieval Conference (TREC-9), NIST Special Publication 500-249*, 2000.

[12] E. Osuna, R. Freund, and F. Girosi. Support vector machines: Training and applications. Technical report, AIM-1602, 1997.

[13] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-2. *In The Second Text REtrieval Conference (TREC-2), NIST Special Special Publication 500-215*, pages 21–34, August-September 1993.

[14] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. *In The Third Text REtrieval Conference (TREC-3), NIST Special Publication 500-226*, November 1994.

[15] A. Rungsawang, P. Uthayopas, M. Lertprasertkul, P. Ingongngam, and A. Laohakanniyom. Worms: A high-performance text retrieval prototype. *HPC-ASIA The Fourth International Conference/Exhibition on High Performance Computing in Asia-Pacific Region*, 2001.

[16] A. Sugiura and O. Etzioni. Query routing for web search engines. In *The Proceedings 9 th International World Wide Web Conference*, pages 412–420, May 2000.

[17] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *International Conference on Machine Learning*, pages 412–420, 1997.

TREC-10 Experiments at KAIST: Batch Filtering and Question Answering

Jong-Hoon Oh, Kyung-Soon Lee, Du-Seong Chang, Chung Won Seo, and Key-Sun Choi

Computer Science Division, Department of EECS, Korea Terminology Research Center for Language and Knowledge Engineering (KORTERM)
Korea Advanced Institute of Science & Technology (KAIST)
Kusong-Dong, Yusong-Gu Taejon, 305-701 Republic of Korea
{rovellia,kslee,dschang,cwseo,kschoi}@world.kaist.ac.kr

1.Introduction

In TREC-10, we participated in two tasks: batch filtering task in the filtering task, and question answering task. In question answering task, we participated in three sub-tasks (main task, list task, and context task).

In batching filtering task, we experimented a filtering technique, which unifies the results of support vector machines for subtopics subdivided by incremental clustering. For a topic, we generated subtopics by detecting similar documents in training relevant documents, and unified the results of SVM classifier for subtopics by OR set operation.

In question answering task, we submitted two runs for main task (KAISTQAMAIN1, KAISTQAMAIN2), two runs for list task (KAISTQALIST1, KAISTQALIST2), and one run for context task (KAISTQACTX).

2. Batch Filtering track

2.1 Experimental Procedure

We experimented a filtering technique, which unifies the results of support vector machines (Vapnick, 1995) for subtopics subdivided by incremental clustering. For a topic, we generated subtopics by grouping similar documents in training relevant documents, and unified the results of SVM classifiers for subtopics by OR set operation.

2.1.1 Subdividing a topic into several subtopics by incremental clustering

For each topic, we generated subtopics by incremental clustering. In incremental clustering, the input documents consist of relevant documents among training documents for a topic. Incremental clustering sequentially processes the input documents and grows clusters incrementally. The first input document itself becomes one cluster. A new document is assigned to a member of all the clusters if the similarity between the document and the pre-generated cluster is above threshold. (A document could be a member of several clusters.) Otherwise, the document becomes a new cluster.

In our experiment, we set a cluster threshold to 0.1. It is a random selection. The comparative experiment was not conducted according to various cluster thresholds. Each cluster has a centroid vector which represents all the documents included in the cluster. We used cosine coefficient as a measure to calculate similarity between a document vector and a centroid vector. For 84 topics of Reuters test collection, we generated 275 clusters. For a topic, a cluster represents a subtopic. Therefore, we generated 275 subtopics. Training relevant documents of a subtopic are the subset of those of a topic.

2.1.2 Filtering based on SVM for each subtopic

Support vector machines are based on the Structural Risk Minimization principle (Vapnik, 1995) from computational learning theory. We tested RBF (radial basis function) models offered by SVM^{light} system (Joachims, 1998), which is an implementation of Vapnik's Support Vector Machine for the problem of pattern recognition. SV learning is based on sub-relevant documents subdivided by incremental clustering and all non-relevant documents for each topic. For each subtopic, we filtered the test documents by using SVM classifier.

2.1.3 Unifying the results of subtopics

We unified the results generated from SVM classifiers by OR set operation. If the binary decision of SVM classifier has true (+1) for at least one subtopic, we decided the document to be relevant for the user's interest.

2.2 Results

We submitted two runs, KAIST10bfo1 and KAIST10bfo2, for the batch filtering track. The TREC-10 filtering track used the Reuters test collection. In batch filtering, the number of training documents is 23,208 and the number of test documents is 822,805. We didn't use any other data. The number of features is 51,265. The weights of terms are calculated by ltc weighting scheme in SMART (Salton, 1983). The KAIST10bf01 run was generated by SVM, and the KAIST10bf02 run was generated by our method.

The batch filtering task was evaluated according to a utility measure (T10SU), a version of the F measure (beta=0.5), precision, and recall. We also evaluated the results according to macro averaged F1 and micro averaged F1. Table 4.1 shows the results. The result of KAIST10bfo1 differ little from KAIST10bfo2.

We expected that the unified results of SVM for subtopics might perform much better than SVM for a topic. However, the result is negative.

Topic set Measure	KAIST10bfo1	KAIST10bfo2
MeanT10SU	0.295	0.298
F-beta	0.496	0.498
Set Precision	0.788	0.785
Set Recall	0.288	0.292
Macro averaged F1	0.379	0.384
Micro averaged F1	0.578	0.585

Table 1. TREC-10 Batch Filtering Results

3.Question and Answering Track

We participate in main task, list task, and context task in TREC-10. Our QA system operates on a set of documents retrieved by information retrieval system. For convenience, we worked with the top-ranked document set generated by NIST. First, 'Question Analyzer' analyzes the given question. It generates question types and extracts keywords of the given question. Then top documents retrieved by the information retrieval system are analyzed for extracting relevant answer. POS tagger and Named entity tagger are used for the purpose. Finally, 'Answer Extractor' generates relevant answers from named entity tagged documents using question type and keywords of the question.

3.1 Main Task

3.1.1 Question Analyzer

A question analyzer parses the given question to identify question types and extract keywords. We define seven kinds of question type for the expected answer as following:

<Person>, <Location>, <Organization>, <Time>, <Currency>, <Measure>, <OTHERS>

They are detected by various patterns (Lee, *et.al*, 2000) and WordNet (Miller, *et.al,* 1991) synsets. <OTHERS> question type is assigned to a question, when there is no pattern for the question.

For extracting keywords, POS tagger (Brill, 1992) and WordNet are used. Noun, adjective, countable numeric and verb except "be (is, are, was, were)" and "do (do, does, did)" are extracted as keywords. Noun phrases in the question are also considered and are extracted with a CFG-styled grammar rule.

NP = (DT (JJ | JJR | JJS)) { NN | NNS | NNP | NNPS }*

If there is an acronym, its expanded form is added to keyword lists. For example, 'Cable News Network', which is expanded form of CNN, is added into keyword list for the question containing word 'CNN'. A query expansion technique is used for a noun phrase and a keyword, which is the only keyword for the given question. They are expanded with noun phrases in the WordNet definition. In the question "What is autism[1]?", for example, 'abnormal absorption', 'communication disorders' and 'short attention span' is added to keyword lists

3.1.2 Answer Extractor

Our answer extractor generates top-5 ranked 50byte phrases. Its inputs are keyword, question type of the given question and top-50 retrieved texts. The top-50 texts are tagged with a POS tagger and a named entity tagger. There are two steps in answer extractor. First, a candidate sentence group is selected. Since, we believe that the context is very important for selecting relevant sentences, we group the previous two sentences, the current sentence and the next two sentences. Among these groups, top-50 sentence groups are selected with a keyword and a question type. Second, selected sentence groups are partitioned into phrases with fixed length. In this step, we use a WordNet definition for detecting relevant answers for the question type <PERSON>, and <LOCATION>. If there is terms with <PERSON> or <LOCATION> tag in the partitioned phrase and a question type of the question is <PERSON> or <LOCATION>, the number of keywords, which appear in WordNet definition of the term[2] is used as a feature for extracting answer. Scoring formula for is as follow:

$$Score(s_i) = 0.5 \times S(s_i, k_j) + 0.25 \times S(s_i, ek_j) + 0.25 \times (S(d_i, ek_j) + S(d_i, k_j))$$

where, S(A,B) represents scoring function for sentence group A and keyword B, s_i represents the i^{th} sentence group, k_j and ek_j represent keywords and expanded

[1] Its definition in WordNet is "autism -- ((psychiatry) an abnormal absorption with the self; marked by communication disorders and short attention span and inability to treat others as people)"

[2] We call the term as a boosting term

keywords for the j^{th} question, and d_i represents WordNet definition for a boosting term in the i^{th} sentence group.

If the $Score(s_i)$ of the top ranked phrase is below T (T is threshold), it is determined that there is no answer for the question in the text.

We expected that the results of this year might perform much better than that of last year (Lee, *et.al*, 2000). However, the result is negative. Since, list task and context task use the module of main task, the result of those is not really good.

3.2 List Task

List task requires the given number of answers. In TREC-10 list task, each question has information of required answer number. Question analyzer should give a question type and the number of the answer. Answer extractor should give the proper number of answers.

For example, "Question Number 1: Name 20 countries that produce coffee.", the question requires twenty answers. Question analyzer just finds number sequence and passes it to the answer extractor.

Answer extraction processes of list task are similar to those of main task except for the limitation of number of the answer. We extended the passage search algorithm for giving the right number of answer.

Answer extractor consists of two phases.

1) Candidate answer listing
2) Finding the right number of answers

First, 'Candidate answer listing' finds the passages and candidate answer set of the passages. We sort the candidate answer by the frequency. Candidate answers are marked with named entity tagger and if a question type and a named-entity type are equal, the entity is added to an answer list. The answer list is sorted by the frequency of the each candidate.

Second, 'Finding the right number of answers' explores the passages for making the answer set. Answer set select by the rules as follow.

1) Answers in the same passages are selected all or nothing.
2) Answers that are on the previous answer passages are deleted from the list.
3) Answers that are included at the highly ranked passages but not in the list are excluded from the output.
4) If the numbers of the candidate answer list are smaller than the needed answer number, we uses the same methods for main task and stop when reached the right

number of the output.

Answers are weighted by normalized frequency of answers and passage weight. The passage-weighting scheme is the same as that of main task. Frequencies are normalized by total frequency and scaled up for balancing passage weights. Frequencies of answer sets are just added to each answer's frequency. It makes the passages to contain more candidates in top-rank.

We combine the frequency and the passage weight using the geometric mean.

$$Weight_{passage} = \sqrt{\left(S \bullet \sum_{i \in answer\ list} freq_i \right)^2 \bullet passage^2}$$

If passages have no candidate answer, the weight is zero; it is similar to context that contains no support information.

3.3 Context Task

In this TREC-10, the context task is introduced for the QA system to exploit context when answering questions. Each individual question in this task has short, fact-based answers as in the main task, and each question has an answer in the collection. However, the interpretation of the question depends on the meaning of and answers to one or more earlier questions in a series. Interpreting a question correctly often involve resolution of referential links within and across sentences (TREC, 2001).

The QA system for the main task did not prepare any solution for this referential link problem. An anaphora resolution module and a keyword expansion module were made up for this weak point of the main task QA system. The system architecture is shown at figure 1. The anaphora resolution module recognizes the anaphora and finds its referent. This resolved referent could be added to the question keyword list.

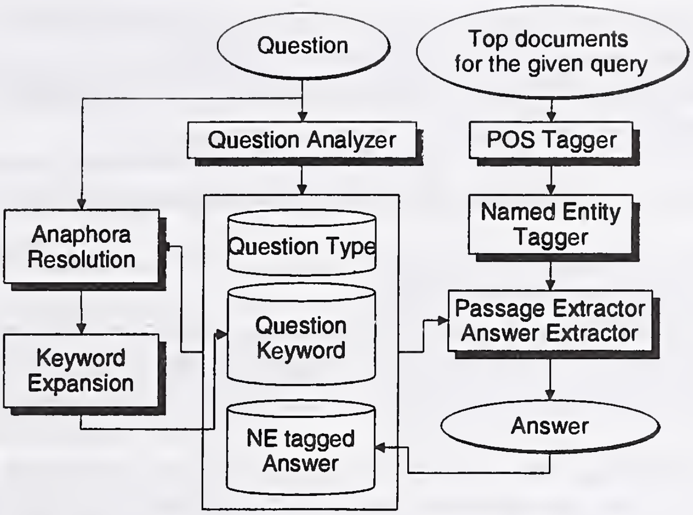


Fig 1 System Description for context task

The anaphora resolution module is composed of NP extractor, anaphora finder, and referent finder. For extracting noun phrase from previous questions and answers, NP extractor has some CFG-styled grammar rules. All found NPs are added to a referent candidate list. Some of the rules are shown as following:

NP = (DT (JJ | JJR | JJS)) { NN | NNS | NNP | NNPS }*

NP = { WP | WP$ } ({ JJ }* NP)

These rules were implemented as FSN (Finite State Network). And, after some rules are exchanged, this FSN is also used to find the anaphora from the question. For all extracted NP, keyword type was given as ‘PERSON’, ‘LOCATION’, ..., ‘Other’.

When it finds the anaphora for the given question, the anaphora resolution module does its duty, looking up the referent candidate list as following sequences.

1) It tries to match the keyword type and agreement between anaphora and referent candidates of the previous question.
2) If the referent was not found, it considers others of the given question series.
3) If the decided referent is WH-pronoun, It selects the real referent from the answer list.

The resolved referent could be added to the question keyword list, and real QA process is working on the top 50 documents of the first one of question list.

References

Brill, E. (1995). Transformation-Based error-driven learning and natural language processing: a case study in part of speech tagging. Computational Linguistics

Miller, G.A., R.Beckwith, C.Fellbaum, D.Gross, and K.Miller. (1991). Five Papers on WordNet. International Journal of Lexicography.

Joachims, T. (1998). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In Proceedings of the European Conference on Machine Learning (ECML).

Lee, K.S., J.H. Oh, J. Huang, J.H. Kim, and K.S. Choi. (2000). TREC-9 Experiments at KAIST : QA, CLIR, Batch Filtering, TREC-9

Rocchio, J.J. (1971). Relevance feedback in information retrieval. In THe SMART Retrieval System - Experiments in Automatic Document Processing, (pp. 313-323). Prentice Hall, Inc.

Salton, G. & McGill, M.J. (1983). Introduction to Modern Information Retrieval, McGraw-Hill, Inc.

Salton, Gerard & Buckley, Chris. (1990). Improving retrieval performance by relevance feedback. Journal of the American Society for Information Science, 41(4):288-297.

Singhal, Amit & Mitra, & Mandar & Buckley, Chris. 1996. Learning routing queries in a query zone. In Proceedings of the Twentieth ACM SIGIR Conference on Research and Development in Information Retrieval, (pp. 21-29).

TREC (2001). http://trec.nist.gov/act_part/guidclincs/qa_track_spec.html, TREC 2001 Question Answering Track Guideline,

Vapnick, Vladimir N. (1995). The Nature of Statistical Learning Theory, Springer-Verlag, New York.

Answering complex, list and context questions with LCC's Question-Answering Server

Sanda Harabagiu*, Dan Moldovan*,
Marius Paşca*, Mihai Surdeanu*, Rada Mihalcea, Roxana Gîrju, Vasile Rus,
Finley Lăcătuşu, Paul Morărescu and Răzvan Bunescu
Language Computer Corporation*
Dallas, TX 75206
{sanda,moldovan}@languagecomputer.com

Abstract

This paper presents the architecture of the Question-Answering Server (QAS) developed at the Language Computer Corporation (LCC) and used in the TREC-10 evaluations. LCC's QAS^{TM} extracts answers for (a) factual questions of vairable degree of difficulty; (b) questions that expect lists of answers; and (c) questions posed in the context of previous questions and answers. One of the major novelties is the implementation of bridging inference mechanisms that guide the search for answers to complex questions. Additionally, LCC's QAS^{TM} encodes an efficient way of modeling context via reference resolution. In TREC-10, this system generated an RAR of 0.58 on the main task and 0.78 on the context task.

Introduction

Systems providing question-answering services need to process questions of variable degrees of complexity, ranging from inquiries about definitions of concepts, e.g. *"What is semolina?"* to details about attributes of events or entities, e.g. *"For how long is an elephant pregnant?"*. Finding the answer to questions often involves various degrees of *bridging inference*, depending on the formulation of the question and the actual expression of the answer extracted from the underlying collection of documents. For example, the question *"How do you measure earthquakes?"* is answered by the following text snippet extracted from the TREC collection: *" Richter scale that measures earthquakes"* because the required inference is very simple: a measuring scalar, i.e. *Richer scale*, has a relative adjunct introduced by the same verb as in the question, having the same object of measurement. Yet a different, more complex form of inference is imposed by questions like *"What is done with worn and outdated flags?"*.

The Question-Answering Server (QAS^{TM}) developed at the Language Computer Corporation (LCC) encodes methods of performing several different bridging inferences that recognize the answer to questions of variable degree of complexity. The pragmatic knowledge required by different forms of inference is distributed along the three main modules of LCC's QAS^{TM}: the *Question Processing* module, the *Document Processing* module and the *Answer Processing* module. Some of the inference forms enabled by LCC's QAS^{TM} determine the answer fusion mechanisms that assemble list-answers expected by questions like *"Name 20 countries that produce coffee."*

Rarely questions are asked in isolation. When satisfied by the answer, a user may have follow-up questions, requiring additional information. If the answer is not satisfactory, a new question may clarify the user's intentions, thus enabling a better disambiguation of the question. LCC's QAS^{TM} is capable of answering questions in context, thus exploiting the common ground generated between the answer of questions like *"Which museum in Florence was damaged by a major bomb explosion in 1993?"* and its follow-up questions *"On what day did this happen?"* or *"Which galleries were involved?"*. These new capabilities of (a) answering more complex questions than those evaluated in TREC-8 and TREC-9; (b) detecting when a question does not have an answer in the collection; (c) fusing several answers that provide partial information for questions expecting list-answers; and (d) answering questions in context - stem from a new architecture, that enhances the three-module streamlined operation used in the previous TREC Q/A evaluations[1].

The architecture of LCC's QAS^{TM}

The architecture of LCC's QAS^{TM} used in the TREC-10 evaluations is illustrated in Figure 1. Three dif-

[1] In TREC-8, the Q/A evaluations showed that the best performing systems exploited the combination of Named Entity semantics with the semantic of question stems. In TREC-9, two trends could be observed: (1) systems that used advanced pragmatic and semantic knowledge in the processing of questions and answers, and (2) systems that improved on new ways of indexing and retrieving the paragraphs were the answers may lie.

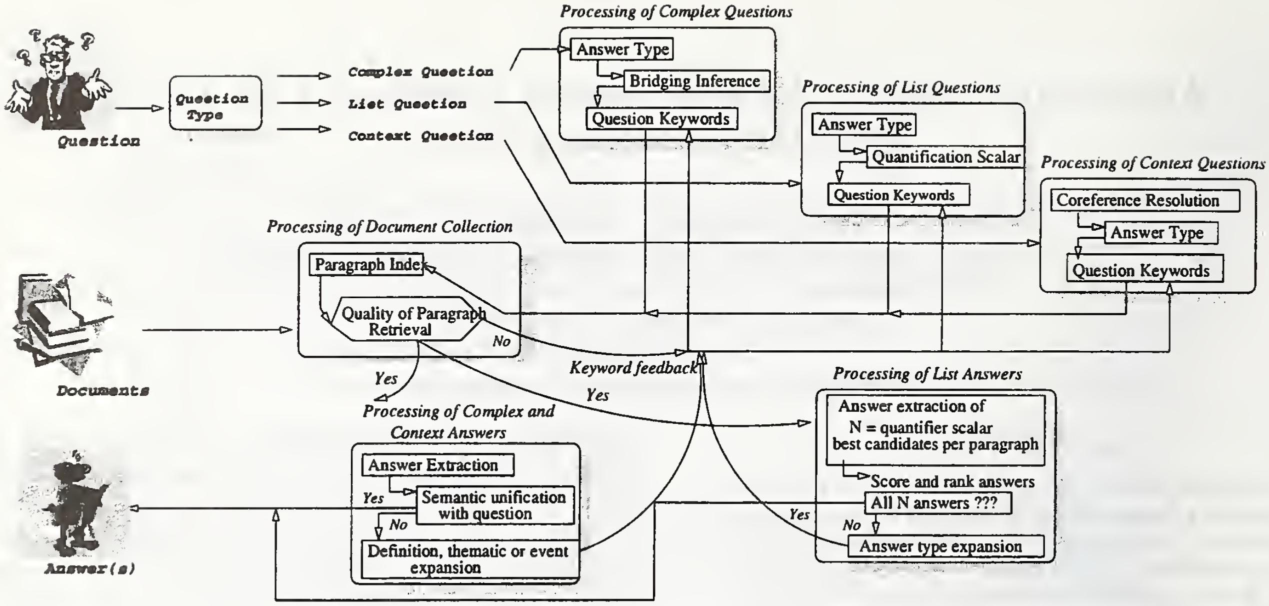

Processing of Complex Questions
Answer Type
Bridging Inference
Question Keywords
Question
Question Type
Complex Question
List Question
Context Question
Processing of List Questions
Answer Type
Quantification Scalar
Question Keywords
Processing of Context Questions
Coreference Resolution
Answer Type
Question Keywords
Processing of Document Collection
Paragraph Index
Quality of Paragraph Retrieval
No
Documents
Yes
Keyword feedback
Yes
Processing of Complex and Context Answers
Answer Extraction
Yes
Semantic unification with question
No
Definition, thematic or event expansion
Answer(s)
Processing of List Answers
Answer extraction of N = quantifier scalar best candidates per paragraph
Score and rank answers
All N answers ???
Yes
No
Answer type expansion

Figure 1: Architecture of LCC's QAS^{TM}

ferent kinds of questions were evaluated: (1) *complex questions*, that expect an answer from the text collections without knowing if such an answer exists; (2) *list questions*, requiring a list of answers; and (3) *context questions*, in which the question was considered in the context of the previous questions and answers processed by the system. Three distinct evaluations were conducted, but a single question-answering architecture handled all three cases.

The *Question Processing* is different for each of the three kinds of questions that were evaluated. For complex, factual questions like *"Q1147: What is the Statue of Liberty made of?"*, the processing involves at first the recognition of the *expected answer type* from an off-line taxonomy of semantic types. In TREC-10, the factual questions were far more complex than those evaluated in TREC-9 and TREC-8 because frequently the expected answer type could not be easily identified. For example, in the case of the question Q1147 virtually anything could be a criterion. To help narrow down the search for the expected answer type and to generate robust processing at the same time, a set of bridging inference procedures were encoded. For example, in the case of the question Q1147, the bridging inference between the question and the expected answer type encode several meronymy relations between different materials and the Statue of Liberty. Instead of searching for the expected answer type in each retrieved paragraph LCC's QAS^{TM} looks for meronymy relations involving any of the keywords used in the query.

For questions expecting a list of answers, the quantification scalar, defining the size of the list, is identified at the time of question processing and used when the answers are extracted and fused together. For example, in the case of question *"Name 15 religious cults."* the expected answer type is ORGANIZATION of the type *religious cult* and the quantifier is 15. Sometimes, the expected answer type have multiple attributes, e.g. *"Name 4 people from Massachusetts who were candidates for vice-president."* Such attributes are translated into keywords that retrieve the relevant document passages or paragraphs.

If the question needs to be processed in the context of the previous questions and answers, a coreference resolution process takes place prior to the recognition of the expected answer type. For example, the pronoun *this* from the question *"On what day did this happen?"* is resolved as the event mentioned in its preceding question, i.e. *"Which museum in Florence was damaged by a major bomb explosion in 1993?"*. The reference resolution entails the usage of the keywords defining the antecedent along with the keywords extracted from the current question.

The *Document Processing* module uses a paragraph index to retrieve document passages that (a) contain the keywords from the query, and (b) contain either a concept of the expected answer type or a relation indicated by the bridging inference mechanisms. However, if insufficient evidence of the paragraph relevance exists, pragmatic information is passed back to the feedback loop that reformulates the query searching for the complex answer. When the most relevant paragraphs are retrieved, the answers are processed.

When answers to complex, factual questions are extracted, their validity is granted by semantic unifications with the question. If the question was asked in context, the unifications of previous questions and answers are also used to grant the validity of the answer of the current question. When unifications are not possible, several expansions that use the gloss definitions of the WordNet concepts are considered. The question is ruled not to have an answer when none of the expansions generate unifications. The processing of list answers is performed differently because LCC's QAS^{TM} extracts for each question all N best candidate answers, where N is the quantifier scalar. Additional answers are sought only if we could not find all N answers and if variations of the keywords defining the same answer type are possible.

Processing questions from the main task

Two main trends have characterized the main task in TREC-10. First, the percentage of questions that ask for definitions of concepts, e.g. *"What are capers?"* or *"What is an antigen?"* represented 25% of the questions from the main task, an increase from a mere 9% in TREC-9 and 1% in TREC-8 respectively. The definition questions normaly require an increase in the sophistication of the question-answering system. Second, in general, the questions had an increased level of difficulty. Questions like *What is the esophagus used for?"* or *"Why is the sun yellow?"* are difficult to process because the answer relies on expert knowledge, from medicine in the former example, and from physics in the latter one. Nevertheless, if a lexical dictionary that explains the definitions of concepts is available, some supporting knowledge can be mined. For example, by inspecting WordNet (Miller 1995), in the case of *esophagus* we can find that it is *"the passage between the pharynx and the stomach"*. Moreover, WordNet encodes several relations, like *meronymy*, showing that the esophagus is part of the *digestive tube* or *gastrointestinal tract*. The glossed definition of the *digestive tube* shows that one of its function is the *digestion*.

The information mined from WordNet guides several processes of bridging inference between the question and the expected answer. First the definition of the concept defined by the WordNet synonym set {*esophagus, gorge, gullet*} indicates its usage as a *passage* between two other body parts: the pharynx and the stomach. Thus the query *" esophagus AND pharynx AND stomach"* retrieves all paragraphs containing relevant connections between the three concepts, including other possible functions of the esophagus. When the query does not retrieve relevant paragraphs, new queries combining esophagus and its holonyms (i.e. gastrointestinal tract) or functions of the holonyms (i.e. digestion) retrieve the paragraphs that may contain the answer. To extract the correct answer, the question and the answer need to be semantically unified.

Q912:	*What is epilepsy?*
Q1273:	*What is an annuity?*
Q1022:	*What is Wimbledon?*
Q1152:	*What is Mardi Gras?*
Q1160:	*What is dianetics?*
Q1280:	*What is Muscular Distrophy?*

Table 1: Examples of definition questions.

The difficulty stands in resolving the level of precision required by the unification. Currently, LCC's QAS^{TM} considers an acceptable unification when (a) a textual relation can be established between the elements of the query matched in the answer (e.g. esophagus and gastrointestinal tract); and (b) the textual relation is either a syntactic dependency generated by a parser, a reference relation or it is induced by matching against a predefined pattern. For example, the pattern "X, particularly Y" accounts for such a relation, granting the validity of the answer *"the upper gastrointestinal tract, particularly the esophagus"*. However, we are aware that such patterns generate multiple false positive results, degrading the performance of the question-answering system.

Definition pattern	*Phrase to be defined (QP)*	*Candidate answer phrase (AP)*
<*AP*> such as <*QP*>	What is autism?	developmental disorders such as autism
<*AP*> (also called <*QP*>)	What is bipolar disorder?	manic-dipressive illness (also called bipolar disorder)
<*QP*> is an <*AP*>	What is caffeine?	caffeine is an alkaloid

Table 2: Identifying candidate answers with pattern matching.

Predefined patterns are also important for processing definition questions, similar to those listed in Table 1. Table 2 lists several patterns and their components: the question phrase (QP) that requires a definition, and the candidate answer phrase (AP) providing the definition. To process definition questions in a more robust manner, the search space is enlarged, allowing the substitution of the phrase to be defined QP with the immediate hypernym of its head. Table 3 illustrates several examples of definition questions that are resolved by the substitution of the QP with its hypernym. The usage of WordNet hypernyms builds on the conjecture that any concept encoded in a dictionary like WordNet is defined by a *genus* and a *differentia.* Thus when asking about the definition of a concept, retrieving the genus is sufficient evidence of the explanation of the definition, especially when the genus is identical with the hypernym.

Phrase to be defined (QP)	*Hypernym from* WordNet	*Candidate answer phrase (AP)*
What is a shaman?	{priest, non-Christian priest}	Mathews is the priest or shaman
What is a nematode?	{worm}	nematodes, tiny worms in soil.
What is anise?	{herb, herbaceous plant}	aloe, anise, rhubarb and other herbs

Table 3: WordNet information employed for detecting answers of defintion questions.

In WordNet only one third of the glosses use a hypernym of the concept being defined as the genus of the gloss. Therefore, the genus, processes as the head of the first NP from the gloss, can also be used to substitute the QP of the definition question. For example, the processing of question Q1273 from Table 1 relies on the substitution of *annuity* with *income* the genus of its WordNet gloss, rather than its hypernym, the concept *regular payment.* The availability of the genus or hypernym helps also the processing of definition questions in which the QP is a named entity, as it is in the case of questions Q1022 and Q1152 from Table 1. In this way *Wimbledon* is replaced with *a suburb of London* and *Mardi Gras* with *holiday.* The processing of definition question is however hindered by the absence of the QP head from the WordNet database. For example, both *diametics* from Q1160 and *Muscular Distrophy* from Q1280 are not encoded in WordNet.

Processing list questions

Unlike complex, factual questions, list questions expect a list of answers. The length of the list is specified by a *quantification scalar* that has to be identified in the natural language question. All the elements of the list must be valid answers to the question and, in addition, the list cannot have duplicate items.

The extraction of the answers depends in large measure on the recognition of the expected answer type. 21 out of 25 test questions (84%) asked about categories that are easily matched by Named Entity recognizers, e.g. countries, cities, people, organizations and currencies. LCC's QAS^{TM} uses a robust answer extraction technology that enables it to extract candidate answers even when the expected answer is unknown. A combination for the keyword features (e.g. the distance between the keywords in the paragraph) enables the server to pinpoint possible answers. However, for processing list answers two additional enhancements had to be encoded.

First we had to allow the extraction of multiple candidate answers from each paragraph. Table 4 illustrates two different paragraphs containing multiple elements of the answer list expected by two distinct questions. The first paragraph contains two elements of the answer list whereas the second paragraph contains three elements of the answer list. Each paragraph has a relevance score associated with it, enabling an ordering of the answers based on the relevance score of its original paragraph. The answer list is assembled by collecting the first N ranked answers, where N is the quantification scalar identified in the question.

Question: Name 20 countries that produce coffee.
FT944-7920: It would also co-ordinate assistance for countries such as *Angola* and *Rwanda*, whose coffee sectors have been badly damaged by war or climatic disasters.
Question: Name 10 countries that banned beef imports from Britain in the 1990s.
AP900601-0140: *West Germany* and *Luxembourg* joined *France* on Friday in banning British beef imports because of "mad cow" disease.

Table 4: Paragraphs containing multiple candidate answers.

Second, we had to discard duplicate candidate answers from the answer list. This operation improves the recall of the answer lists. To this end we implemented an *answer normalization* procedure, encoding two functions: (1) name alias recognition, identifying *United States of America*, *USA*, *U.S.* and *US* as the same entity; and (2) distinguish separate entities bearing the same name, e.g. *Paris, France* and *Paris, TX.* Table 5 illustrates several text snippets that contain duplicate candidate answers for the same question.

Context questions

Processing a sequence of questions posed in the same context requires the resolution of several forms of reference. A question may use:

1. *demonstrative pronouns*, like this, these or there; (e.g. "On what day did this happen?", or "Where were these people located?" or "Name a company that flies there?")
2. *third person pronouns*, like he or it; (e.g. "What California winery does he own?" or "In what facility was it constructed?")

Question: Name 20 countries that produce coffee.
First instance non-duplicate (FT944-2823): in Brazil. Mr. Jorge Cardenas, head of the
Duplicate (FT933-1482): Brazil, intends to cover exporters' costs
Duplicate (WSJ911203-0140): in Brazil, said producers told her there has
Duplicate (AP900718-0272): notably Brazil, the world's largest producer
Duplicate (WSJ870602-0079): said Brazil, the world's largest coffee

Table 5: 50-byte text snippets containing duplicate candidate answers.

3. *possessive pronouns*, like his or its; (e.g. "What was his first radio song?")
4. *definite nominals*, in which the definite article or the demonstrative pronoun indicate that the concept was already introduced by a previous question or answer; (e.g. "What executive from the company was a member of the Supreme Council in 1994?" or "This city's name was later changed to what?")
5. *nominalizations* of verbs used in previous questions; (e.g. "When was construction begun?" following "In what facility was it constructed?")
6. *elliptical* reference, in which the expected answer type is inherited from the previous question; (e.g. "How many are poisonous to humans?" following "How many species of spiders are there?")
7. *causal-effect* reference; e.g. explosive from "How much explosive was used?" is the cause of explosion from its preceding question "Which museum in Florence was damaged by a major bomb explosion in 1993?"
8. *meronymic* reference, e.g. galleries from "Which galleries were involved?" are referenced as a part of the *museum* from the preceding question "Which museum in Florence was damaged by a major bomb explosion in 1993?".

The resolution of all the forms of reference is performed by identifying the antecedent of the anaphora in (1) a previous question; or (2) the answer to a previous question; or (3) an anaphor used in a previous question. Before applying the reference resolution algorithm, the pleonastic usage of pronouns is identified, ruling out the resolution of pronouns like *there* in "How many species of spiders are there?"

The reference resolution algorithm employed by LCC's QAS^{TM} is different from reference resolution algorithms used in discourse or dialog processing because the goal is not to resolve the reference, but to identify the question that either contains the antecedent of the reference or expects an answer that contains the antecedent. Consequently, when processing the question Q_1 that contains a reference, by knowing which preceding question Q_0 generates the antecedent, we can combine the keywords of Q_1 with the keywords of Q_0 to retrieve relevant paragraphs. Moreover, since question keywords are extracted in a predefined order in QAS^{TM}, when keywords from two different questions are combined, the keywords from the previous question always preceded the keywords from the current question. This keyword ordering is important for the feedback loops implemented in LCC's QAS^{TM}, illustrated in Figure 1. For example Table 6 illustrates the combination of keywords resulting from the reference resolution within context questions.

Example 1 *Question CTX1d:* How many people were killed?
Keywords from CTX1d: (k_1=killed)
Reference of question CTX1d = question CTX1a: Which museum in Florence was damaged by a major bomb explosion in 1993?
Keywords from CTX1a: (k_2=Florence, k_3=bomb, k_4=explosion)
Keywords used to process CTX1d: (k_2=Florence, k_3=bomb, k_4=explosion, k_1=killed)
Example 2 *Question CTX7g:* How wide?
Keywords from CTX7g: (k_1=wide)
Reference of question CTX7g = question CTX7a: What type of vessel was the modern Varyag?
Keywords from CTX7a: (k_2=Varyag)
Keywords used to process CTX7g: (k_2=Varyag, k_1=wide)

Table 6: Keyword extraction for context questions.

The algorithm that performs reference resolution for context questions is:

```
Algorithm Reference Context Resolution(Q)
Input: LQ = precedence-ordered list of previous
    questions asked in the same context + $w_Q$,
    where $w_Q$ = the reference word from Q
    when we have an ellipsis $w_Q = \phi$

if ($w_Q$ repeats in a question Q' from LQ)
      return Q' if it does not contain a reference
      else return Reference Context Resolution(Q')
if ($w_Q$ is a pronoun)
   CASE ($w_Q \in$ {he,his,she,her,they,their})
     return Q', the closest question from LQ that has
       the expected answer type=PERSON or has a
       PERSON mentioned
   CASE ($w_Q \in$ {it,its})
     if $w_Q$ is the subject of one of
       the verbs {happen, occur} return Q' the first
       question that mentions an event
     return Q', the closest question from LQ that has
```

```
      the expected answer type different than PERSON
      or mentions some non-PERSON entity
  CASE ($w_Q$ =there)
    return Q', the closest question from LQ that has
      the expected answer type=LOCATION or has a
      LOCATION mentioned
  CASE ($w_Q$ =this or $w_Q = \phi$)
    return Q', the first question from LQ
      if it does not contain a reference
      else return Reference Context Resolution(Q')
if ($w_Q$ morphological-root ($w_Q$) =
                        morphological-root ($w_{Q'}$))
  -where $w_{Q'}$ is a word from a question Q'∈ LQ
  return Q'
if (there is a WordNet semantic relation
        (e.g. meronymy) between $w_Q$ and $w_{Q'}$)
  -where $w_{Q'}$ is a word from a question Q'∈ LQ
  return Q'
```

An interesting by-product of this reference resolution algorithm was the way it allowed the modeling of context through the passing of keywords from the antecedent question to the follow-up question. It is interesting to be noted that each time when a follow-up question would be processed, LCC's QAS^{TM} would operate on the same relevant paragraphs as for the antecedent question in 85% of the cases. However, it would extract different answers, since the expected answer type would be different.

Performance evaluation

Table 7 summarizes the scores provided by NIST for our system. At the time of this writing we did not have the results for the list questions.

	NIST score *lenient*	NIST score *strict*
Main Task	58.7%	57.0%
Context Questions	77.8%	77.0%

Table 7: Accuracy performance

The reading of the results from Table 7 may be misleading - one could conclude that it is easier to process questions in context rather than processing them in isolation. There is a quantitative and a qualitative aspect to this conclusion. First, in TREC-10 there were only 31 questions that were processed in the context of another question whereas in the main task, where questions were processed in isolation, we evaluated close to 500 questions. Second, the first questions in each context were much easier to process then most of the questions from the main task (e.g. definition questions). However, the way context was modeled in LCC's QAS^{TM} was quite felicitous.

Lessons learned

In TREC-10 we learned again that open-domain resources such as WordNet can be fully exploited to process more and more complex definition questions or for processing questions in context. We also learned that such resources are not exhaustive, thus Q/A systems need to robustly process questions even when lexico-semantic information is not available. We also learned that when questions are classified by very broad, practical criteria, e.g. questions asked in isolation vs. questions asked in context, we need to operate changes in the architecture of the Q/A system, using novel ways of solving reference - by customizing its resolution for the Q/A task rather than using methods of resolving the linguistic phonemenon of reference.

For TREC-10 we introduced new modules at the level of question processing to guide the search and extraction of answers based on several forms of bridging inference, mostly determined by lexico-semantic cues. As context questions will probably become more complex, we hope to enhance our bridging inference procedures by relying more on the semantics of follow-up questions.

References

Sanda Harabagiu and Dan Moldovan. Knowledge Processing on Extended WordNet. In *WordNet: An Electronic Lexical Database and Some of its Applications*, editor Fellbaum, C., MIT Press, Cambridge, MA, 1998.

Sanda Harabagiu and Steven Maiorano. Finding answers in large collections of texts: paragraph indexing + abductive inference. *Working Notes of the Fall AAAI Symposium on Question Answering*, November 1999.

Sanda Harabagiu, Marius Paşca and Steven Maiorano. Experiments with Open-Domain Textual Question Answering. In the *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*, pages 292–298, 2000.

Sanda Harabagiu, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Girju, Vasile Rus and Paul Morarescu. FALCON: Boosting Knowledge for Answer Engines. *Proceedings of the Text Retrieval Conference (TREC-9)*, 2000.

Sanda Harabagiu, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Girju, Vasile Rus and Paul Morarescu. The

Role of Lexico-Semantic Feedback in Open-Domain Textual Question-Answering. *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-2001)*, Toulouse France, pages 274-281, 2001.

G.A. Miller. WordNet: A Lexical Database. *Communication of the ACM*, vol 38: No11, pages 39–41, November 1995.

Dan Moldovan and Rada Mihalcea. A WordNet-based Interface to Internet Search Engines. In *Proceedings of the FLAIRS-98*, pages 275–279, 1998.

Dan Moldovan, Sanda Harabagiu, Marius Paşca, Rada Mihalcea, Richard Goodrum, Roxana Gîrju and Vasile Rus. LASSO - A tool for Surfing the Answer Net. *Proceedings of the Text Retrieval Conference (TREC-8)*, 1999.

Dan Moldovan, Sanda Harabagiu, Marius Paşca, Rada Mihalcea, Richard Goodrum, Roxana Gîrju and Vasile Rus. The Structure and Performance of an Open-Domain Question Answering System. *Proceedings of the 38th Annual Meeting of the Association for Comoutational Linguistics (ACL-2000)*, pages 563–570, 2000.

Marius Paşca and Sanda Harabagiu. High Performance Question/Answering. *Proceedings of the 24th 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-2001)*, New Orleans LA, pages 366-374, 2001.

Marius Paşca and Sanda Harabagiu. The Informative Role of WordNet in Open-Domain Question Answering. *Proceedings of the NAACL 2001 Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations*, Carnegie Mellon University, Pittsburgh PA, pages 138-143, 2001.

Marius Paşca and Sanda Harabagiu. Answer Mining from On-Line Documents. *Proceedings of the ACL-2001 Workshop on Open-Domain Question Answering*, Toulouse France, pages 38-45, 2001.

Finding an answer based on the recognition of the question focus

O. Ferret, B. Grau, M. Hurault-Plantet, G. Illouz, L. Monceaux, I. Robba, A. Vilnat
LIR Group
LIMSI – CNRS (France)

1. Introduction

In this report we describe how the QALC system (the Question-Answering program of the LIR group at LIMSI-CNRS, already involved in the QA-track evaluation at TREC9), was improved in order to better extract the very answer in selected sentences. The purpose of the main Question-Answering track in TREC10 was to find text sequences no longer than 50 characters or to produce a "no answer" response in case of a lack of answer in the TREC corpus.

As QALC first retrieves relevant sentences within the document corpus, our main question was: how to find the answer in a sentence? This question involves two kinds of answer: a) it is better to know what you look for and b) you have to know the location of what you look for. The first case is solved by applying a question analysis process. This process determines the type of the expected answer in term of named entity. This named entity is searched for in the sentences. However, all answers cannot be expressed in term of a named entity. Definition questions or explanation questions for example demand phrases (noun phrases or verb phrases) as answers. So, after having studied the structure of subpart of sentences that contained answers, we defined criteria to be able to locate the precise answer within a sentence. These criteria consist in defining triplets composed of a question category, the question focus and an associated list of templates allowing the location of the answer according to the focus place in the candidate sentence.
In the following sections, we will detail this novel aspect in our system by presenting the question analysis module, the different processes involved in the answer module and the results we obtained. Before, we give a brief overall presentation of QALC.

2. The overall architecture of QALC

The basic architecture of QALC is composed of different modules, one dedicated to the questions, one to the corpora, and a last module in charge of producing the answer. Each of these main modules is decomposed in several processes (see Figure 1).

The system is based on the following modules:

- Question module. This module regroups a question analysis process and a term extractor. The analysis of the questions relies on a shallow parser (Aït-Mokhtar 1997) in order to extract several pieces of information from the questions:
    - an answer type that corresponds to the types of entities which are likely to constitute the answer to this question.
    - a question focus: a noun phrase that is likely to be present in the answer
    - a question category that gives clues to locate the answer

The term extractor is based on syntactic patterns that describe compound nouns. The maximal extension of these compounds is produced along with the plausible sub-phrases. All the noun phrases belonging to this maximal extension are also produced.

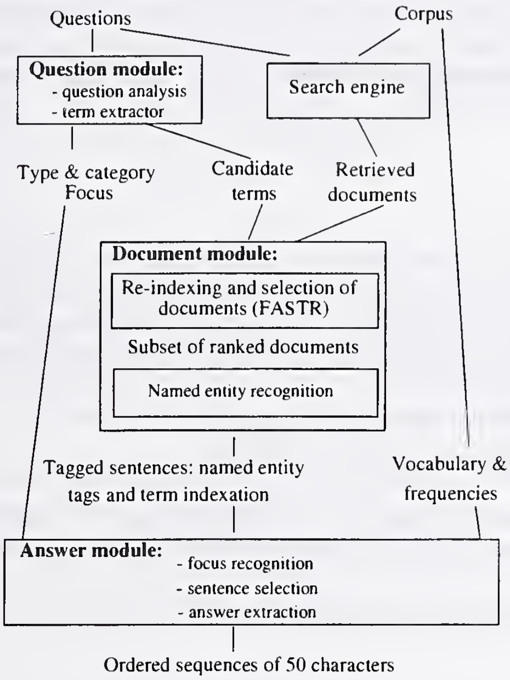


Figure 1: QALC architecture

- Document module. We use the outputs provided by NIST, resulting from the application of the ATT search engine. The 200 best documents are re-indexed by Fastr (Jacquemin 1999), a shallow transformational natural language analyzer that recognizes the occurrences and the variants of the terms produced by the term extraction process. Each occurrence or variant constitutes an index to the document that is ultimately used in the process of document ranking and in the process of question/document pairing. These indexes allow QALC to reorder the documents and entail the selection of a subpart of them (Ferret & al. 2001). A named entity recognition process is then applied on the resulting sets of documents.
- Answer module. This module relies on two main operations: the sentence selection and the answer extraction. All the data extracted from the questions and the documents by the preceding modules are used by a pairing module to evaluate the degree of similarity between a document sentence and a question. The answers are then extracted from the more relevant sentences according to several criteria:
  a) the presence of the expected answer type or not,
  b) the focus recognition in the sentence
  c) the category of the question and its associated patterns.

3. Natural Language Question Analysis

Question analysis is performed in order to assign the questions some features that will be used in the answer module. In view of a better search for the response, question analysis has to give as much information as possible. In our Trec9 system, this analysis allowed the prediction of an answer type, when it was a named entity (for instance, ORGANIZATION). In our Trec10 system, question analysis still allows the prediction of a named entity answer type but also the prediction of a more general answer type. Moreover, question analysis provides new information: the question focus and the question category.

3.1 Answer Type

The question analysis module tries to assign to each question an answer type, which may be a named entity or a more general type. In the first case, the module tries to find if the answer type corresponds to one or several named entity tags sorted by importance order. The named entity tags are hierarchically organized within 17 semantic classes (Ferret and al. 2000). For example:

Question: Who developed the Macintosh Computer?
Named Entity List = PERSON ORGANIZATION

In addition, question analysis tries to deduce a more general type. It means to find a noun or a noun phrase that corresponds to an entry in the WordNet lexical base. For example,

Question: What metal has the highest melting point?
General Type = metal

Question: What is the name of the chocolate company in San Francisco?
Named Entity List = ORGANIZATION
General Type = company

3.2 Focus

Next, question analysis tries to deduce the question focus, which corresponds to a noun or a noun phrase that is likely to be present in the answer. For each question, we will determine a focus, a focus head (the main noun) and the "modifiers" of the focus head (adjective, complement...). For example:

Question: Who was the first governor of Alaska?
FOCUS = the first governor of Alaska
FOCUS-HEAD = governor
MODIFIERS-FOCUS-HEAD = ADJ first, COMP Alaska

3.3 Question Category

The detection of question category gives us a clue to find the location of the answer in a candidate sentence. Each question category corresponds to a syntactic pattern. The question category is the "syntactic form" of question. For example:

Question: What does a defibrillator do?
Category = WhatDoNP

Question: When was Rosa Park born?
Category = WhenBePNborn

After studying the questions of TREC8 and TREC9 along with the sentences containing an answer, we found more than 80 question categories. This repartition of questions in categories enables the definition of rules to find the focus and answer type information.

3.4 Criteria for question analysis

To find all these different items of information, we used syntactic and semantic criteria. Syntactic information is provided by a shallow parser (Aït-Mokhtar 1997) applied to all questions. Thus, QALC obtains a segmentation of each question into chunks and a set of syntactic relations between them. But often, the shallow parser is not appropriate for analyzing question, so we had to recapture parse mistakes.

Rules to find the focus, the category and the answer type were written from the syntactic representation of the question. Semantic criteria are extracted from the WordNet lexical base to improve the named entities glossary, and to find a more general answer type.

For the TREC10 questions, our question module finds 85 % of the correct focus, 87 % of correct general answer type and 90.5 % of correct named entity type.

4. Focus recognition

The focus of a question is structured as follows: (a) the head of the focus, (b) a list of modifiers. QALC tries to locate this focus in the sentences of the selected documents. It first detects the head of the focus, and then identifies the noun phrase in which the head is enclosed. To determine the frontiers of this noun phrase, we define a local grammar for the NP in English. This grammar relies on the tagging made by the Tree-Tagger (Smidt&Stein 99). For example, for the question 827:

"Who is the creator of the Muppets?",

the focus is "the creator of the Muppets", with the head : "creator".

In a document, we found the following NP:

late Muppets creator Jim Henson,

which fits the expression:

Adjective + Plural Noun + Noun + Proper Noun + Proper Noun

We also look for NPs containing synonyms of the question focus head. These synonyms are determined by FASTR. When the recognition of a focus in the question failed, QALC looks for the proper nouns in the question, and it tries to recognize NPs containing these proper nouns.

When these NPs are delimited, we associate them a score. This score takes into account the origin of the NP and the modifiers found in the question: when the NP contains the modifiers present in the question, its score is increased. The best score is obtained when all of them are present.

In the example on question 827, the score is maximal: the NP has been obtained directly from the focus of the question, all the significant words of the focus are present: "*creator*" and "*Muppets*".

When the NP is obtained with a synonym of the focus head, the score is only slightly decreased, and a little more when it is obtained via a proper noun. However the scoring algorithm always takes into account the ratio between the number of words present in the question phrase and in the document noun phrase.

For example the score assigned to the NP:

"*their copy of the 13th century Magna Carta*" obtained for the question 801 :

"*Which king signed the Magna Carta*", has a lower score because it has not been obtained from the focus ("*king*"), but from the proper noun "*Carta*", even if it contains all the words of this proper noun phrase: "*Magna*" and "*Carta*".

For each sentence of the selected document, QALC tags all the relevant NPs following the preceding algorithm, with the associated scores. It only keeps the NPs obtaining the best scores, which in turn provides an evaluation of the relevance of the sentence, which will be used in the pairing module in charge of the sentence selection.

5. Sentence selection

In our system for TREC 10, the pairing module achieving the selection of a set of sentences that possibly contain the answer to a question is based on the same principle as the pairing module used in our TREC 8 and TREC 9 systems: it compares each sentence from the selected documents for a question to this question and constantly keeps in a buffer the *N*[1] sentences that are the most similar to the question. This comparison relies on a set of features that have been extracted both from the questions and the sentences of the selected documents:

- terms;
- focus;
- named entities;
- scattering of terms in the sentence.

A specific similarity score is computed for each of these features. The last feature enables the module to decide between two sentences having the same score for the first three features.
We tried different weighting schemes for terms (Ferret & al 2000). The one we choose here was to sum the weights of the terms of the question that are in the document sentence. A term weight integrates its normalized information with regards to a part of the *QA* corpus (vocabulary frequencies in figure 1) and the fact that it is or not a proper noun.
The term score is combined with the focus score and the resulting score constitutes the first criterion for comparing two document sentences *S1* and *S2*: if *S1* has a combined score much higher than *S2*[2], *S1* is ranked on top of *S2*. Otherwise, the named entity score is used in the same way. It evaluates to what extent a named entity in a document sentence can fit the target of a question when the expected answer is a named entity. This measure takes into account the distance of their two types in.our named entity hierarchy.
When the two preceding criteria are not decisive, the first criterion is used once again but with a smaller threshold for the difference of scores between two sentences. Finally, if there is still an uncertainty, the module ranks first the sentence that has the shortest matching interval with the question. This interval corresponds to the shortest part of the sentence that gathers all the terms of the question that were recognized in it.

[1] *N* is at least equal to 5. The selected sentences are ranked according to their similarity to the question.

[2] « Much higher » means that the difference of scores for *S1* and *S2* is higher than a fixed threshold.

6. Answer extraction

The extraction process depends on whether the expected answer type is, or is not, a named entity. Indeed, when the answer type is a named entity, the extraction consists of the location of the named entity within the sentence. Thus, it mainly relies on the results of the named entity recognition module. On the other hand, when the answer type is not a named entity, the extraction process mainly relies on the recognition of the question focus, as it consists of the recognition of focus-based syntactic answer patterns within the sentence.

6.1. Named entity extraction

When the question allows the system to predict the kind of expected answer in term of a named entity type, the extraction of the answer is based on this information. This process looks for all the expressions tagged with the searched type. If several such expressions exist, we choose the closest to the focus, if it was recognized in the sentence, otherwise the first one. When there is no named entity of the type desired, QALC generalized the searched type using our own hierarchy. By this way, when looking for a person, QALC will look for a proper name, or look for a number instead of a length, etc.

6.2. Answers of type "common noun or verb phrase"

When the expected answer type is not a named entity, the QALC system locates the very answer within the candidate sentence through syntactic patterns. Syntactic patterns of answer include the focus noun phrase and the answer noun phrase, which can be connected by other elements such as comma, quotation marks, a preposition or even a verb. Thus, a syntactic pattern of an answer always includes the focus of the question. As a result, the focus has to be determined by the question analysis module in order to enable the QALC system to find a common noun or verb phrase as answer.

If we consider the following question (n°671):

" *What do Knight Ridder publish?* "

The focus of the question, determined by the rules of the question analysis module, is "*Knight Ridder*". This question pertains to the question type What-do-NP-VB, with "*Knight Ridder*" as NP and the verb "*publish*" as VB.

One answer pattern applying to this category is called FocusBeforeAnswerVB and consists of the following syntactic sequence:

NPfocus Connecting-elements NPanswer

The NPfocus is the noun phrase corresponding to the question focus within the sentence-answer. It is followed by the connecting elements, then by a noun phrase that is supposed to contain the very answer. The connecting elements mainly consist of the question verb (VB in the question type).

The following answer, which was found in the documents corpus, fits with the FocusBeforeAnswerVB pattern:

" *Knight Ridder publishes 30 daily newspapers ...* ",

This answer was extracted from the following sentence:

> " *Knight Ridder publishes 30 daily newspapers, including the Miami Herald and the Philadelphia Inquirer, owns and operates eight television stations and is a join venture partner in cable television and newsprint manufacturing operations.* ".

We saw, in section 3.3, that about 80 question categories were determined from the corpus. Among them, about 45 do not expect a named entity as answer, and thus need syntactic patterns. For each of those question types, we built syntactic patterns. The different patterns, as well as the different question types, were empirically determined from corpus analysis. The corpus consisted of the questions and answers provided after the TREC8 and TREC9 conferences. We considered 24 patterns. The number of patterns for each question type varies from 2 to 20, with an average of 10 patterns for each question category. Thus, several question types share the same pattern.
The difficulty in finding syntactic patterns varies according to the question type. This difficulty is partly due to the small number of some question types within the corpus, and, for the most part, to the grammatical diversity of the answers. For example, there is few " Why " questions (4) and few " How verb " questions (4), such as " Why can't ostriches fly? " (n° 315) and " How did Socrates die? " (n° 198). Moreover, answers to those questions can hardly be reduced to a pattern. We also hardly found grammatical regularities in the answers to the " What-GN-be-GN " questions, such as " What format was VHS's main competition? " (n° 426) or " What nationality was Jackson Pollock ? " (n° 402) for instance. Indeed, depending on the situation, it is the first NP (" format " or " nationality ") or the second NP (" VHS " or " Jackson Pollock "), which plays the main role in the pattern.

7. Results and Analysis

The three runs that we sent to TREC10 come from the same selection of the top ten more relevant sentences. Those runs are the result of three different weighting schemes for the top ten answers, weighting that thus ranked them differently.

	run QALIR1	run QALIR2	run QALIR3
strict evaluation	0.181	0.176	0.167
lenient evaluation	0.192	0.188	0.179

7.1 Top five answer selection

For each question, the pairing module presented section 5 selects ten sentences. Hence, the final problem is to choose five ranked answers among them. Three strategies were implemented:

- selecting the first five answers according to the order given by the pairing module. This is more precisely the order of the selected sentences from which the answers were extracted;
- selecting the first five answers according to the order given by the answer extraction module. This module ranks its answers according to the patterns that were applied for extracting them. The answer score is the highest when the pattern applied is the most typical for the question category;
- a mixed strategy that merges the two previous lists : following the order of the two preceding lists, one answer is alternately taken from one list and the following from the other list until having five answers.

No specific processing was done for detecting that an answer cannot be found in the *QA* corpus: a « no answer » answer is provided when the pairing module cannot select at least one sentence or when the answer extraction module cannot apply a syntactic pattern in the selected sentences.
For providing a « final answer », we only worked on detecting when the answers to a question are globally not sure. Otherwise, we considered that the first answer of our list (rank 1) was the final answer. Comparing the lists given by the two first strategies of answer selection did the detection of the unsure cases: if the two lists were too different according to a similarity measure, the question was marked as unsure. This measure takes into account the differences concerning both the presence of an answer and the rank in the lists.

7.2 QALC performances according to the expected answer type of the question

We previously distinguished questions that expect a named entity as answer, from questions that expect a noun or verb phrase. Indeed, when the answer is a named entity, the location of the answer within the sentence is facilitated by the presence of the named entities tags within the documents. In fact, QALC obtains better results regarding the named entity questions than other questions. Actually, while 46.5% of the TREC10 questions expect a named entity as answer, 56% of the correct answers respond to a named entity question. The named entity questions are questions that have been recognized as such by the question analysis module. On the other hand, QALC achieves not so good performances regarding the named entity questions in TREC10 (31.4% of correct answers) than in TREC9 (39.3% of correct answers).
Anyway, the QALC system performs better than its previous version concerning the questions that expect a noun or verb phrase. Indeed, 21.3% of those questions have correct answers in TREC10 evaluation, for only 10% in TREC9.

8. Conclusion

In this article, we focused on the extraction of the precise answer. As we described above, the QALC system first selects the sentences that respond to the question, and then, extracts the precise answer from them. This principle is efficient when the selected sentences have weights very different from each other. In this case, the answer has a high probability to be in one of the top ten sentences. But, when many sentences have close weights, the answer may be just as well in the fiftieth sentence as in the first one. To face up this situation, another strategy has to be carried out.
Another problem we met with is the setting-up of the syntactic patterns of answer. Those patterns are drawn from question-answer corpora, and thus require large corpora to be efficient. This is not the only difficulty: answers to some categories of question can hardly be reduced to patterns. We have to find another solutions concerning those categories. One solution we tested uses WordNet. Indeed, we noticed that knowing the expected answer type (when it does exist) facilitates the recognition of the answer within the sentence. At present, QALC recognizes named entities such as persons, cities, states, organizations, and numbers such as financial amounts and physical magnitudes. However, the QALC question analyzer is able to recognize more answer types than those that are now tagged by the named entity recognition module. For instance, the question n° 380 "*What language is mostly spoken in Brazil* " expects a language name as answer. As this type is not tagged within the documents, QALC is not able to locate it directly within the sentences. Thus, we tested the use of WordNet so as to validate the answer. " *Portuguese* ", which is the answer in our example, is a member of the hyponym hierarchy for « language » in WordNet. This gives us a way to validate the answer: if one of the answers found

by QALC has as hypernym the answer type recognized by QALC within the question, thus QALC would select this answer.

References

(Aït-Mokhtar 1997) Aït-Mokhtar S. and Chanod J. (1997), Incremental Finite-State Parsing, In Proceedings of *ANLP-97*, Washington.

(Ferret & al 2000) O. Ferret , B. Grau , M. Hurault-Plantet, G. Illouz, C. Jacquemin, QALC — the Question-Answering system of LIMSI-CNRS, pre-proceedings of TREC9, NIST, Gaithersburg, CA.

(Ferret & al 2001) O. Ferret , B. Grau , M. Hurault-Plantet, G. Illouz, C. Jacquemin, Document selection refinement based on linguistic features for QALC, a question answering system, RANLP 2001, bulgarie

(Jacquemin 1999) Jacquemin, C., Syntagmatic and paradigmatic representations of term variation. In Proceedings, ACL'99, University of Maryland, 341-348.

(Schmid 1999) H. Schmid, Improvements in Part-of-Speech Tagging with an Application To German. In Armstrong S., Church K.W., Isabelle P., Manzi S., Tzoukermann E., and Yarowsky D. (eds) Natural Language Processing Using Very Large Corpora, Kluwer Academic Publishers, Dordrecht.

MSR-Asia at TREC-10 Video Track: Shot Boundary Detection Task

Yu-Fei Ma[*], *Jia Sheng*[+], *Yuan Chen*[#], *Hong-Jiang Zhang*[*]

[*] Microsoft Research, Asia {i-yfma, hjzhang}@microsoft.com
[+]Department of Computer Science and Technology, Tsinghua University
[#] Computer and Information Science and Engineering Department, University of Florida

Abstract

The video track is added into TREC-10, composed of two tasks, automatic shot boundary detection and video retrieval. In this year, we (MSR-Asia) participated in the video track, focusing on shot boundary detection task. Our work is to find out all of boundaries the shot changes by a fast algorithm based on uncompressed domain. In our algorithm, all of non-cut transitions are considered as gradual transition, including dissolve, fade-in, fade-out, and all kinds of wipes. Experimental results indicate that the accuracy and processing speed of our algorithm are all very satisfactory.

1. Introduction

Shot is the basic unit of video sequence, hence, is important for digital video processing. Shot boundary detection is the first step for video content analysis. Since there are usually many shots in a video sequence, the automatic algorithm for shot boundary detection is indispensable.

The shot transition can be classified into two types: abrupt transition (cut) and gradual transition. Gradual transition usually includes dissolve, fade-in, fade-out and all kinds of wipes. Cuts are generated by camera operations, such as starting or stopping recording, or editing operations, while gradual transitions are generated only by editing operations

There are so many literatures addressing the algorithms of shot boundary detection [1-7]. Two fundamental approaches are used: 1) Compressed domain based methods [3,4,5], and 2) Uncompressed domain based methods [2,6,7]. The former usually is much faster than the latter, but its accuracy is difficult to be improved. Additionally, the former must be adaptive to different compression formats or decoders. Comparing with the former, much more methods could be used for the uncompressed domain based methods. Moreover, with the enhancement of hardware and compression standards, the decoding speed is never a drawback.

In this paper, we propose an uncompressed domain based approach for fast shot boundary detection. We employ a block-wise comparison based algorithm for cut detection and a run-length based algorithm for gradual transition detection. They are integrated seamlessly under the framework of Finite State Automata. In addition, the self-adaptive thresholds are used for robustness purpose. The experiments were carried out on large amount video sequences. The test set is provided by NIST (National Institute of Standard and Technology). The results are also evaluated by NIST.

The rest of paper is organized as follows. In section 2, we first introduce the system framework. The details of our algorithms will be described in section 3. Then the experimental results are presented in section 4. Section 5 draws some conclusions.

2. Framework

Our approach consists of four functional modules which are decoding, feature extraction, inter-frame comparison, and decision, as shown in Fig.1. Since our algorithm is based on uncompressed data, the video sequences would be decoded in the decoding module first, if it is in a compressed format, such as mpeg. Then, the visual features are extracted from each decoded frames and compared in the feature extraction module and the Inter-frame comparison module respectively. In our

[+] [#] This work was done while these authors were visiting Microsoft Research, Asia.

algorithm, block-based average color and color histograms are used as visual features. With a Finite State Automata, the shot boundaries are detected by self-adaptive threshold in the decision module.

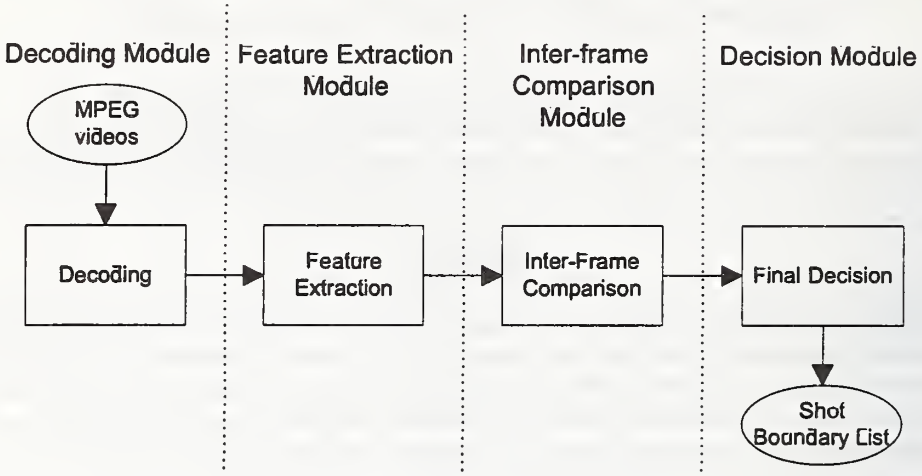


Fig.1: Framework

3 Algorithms Description

In our algorithms, we assume that if there is an apparent deviation in the visual feature between two frames, a shot transition will occur there. So the difference between two successive frames is used as a measure of variation of video sequence. The key issues here are 1) what visual features are extracted from frames, ands 2) what similarity measure is adopted. We used different methods to deal with cut and gradual transition.

3.1 Cut Detection

The pixel-wise difference between two successive frames could be used as dissimilarity measure. However, it is very sensitive to the motion, including camera movement and object motion. To reduce the disturbance of motion, we employ a block based comparison. In RGB color space, let $c_{ij}^{(t)} = (r_{ij}^{(t)}, g_{ij}^{(t)}, b_{ij}^{(t)})$ denote the color of the pixel at the pixel (i, j) in the t-th frame. Then we divide each frame into $m \times n$ blocks and comparisons are carried out on blocks instead of pixels. The average color of block (p, q) in the t-th frame could be defined as follows:

$$\overline{c_{pq}^{(t)}} = \frac{1}{m \times n} \sum_k c_k^{(t)} = \frac{1}{m \times n} \sum_k (r_k^{(t)}, g_k^{(t)}, b_k^{(t)}) \qquad (1)$$

where $c_k^{(t)}$ is the pixel color within a block field. Then the block-wise difference between t and $(t-1)$-th frame is defined:

$$d'^{(t)}_{pq} =| \overline{c_{pq}^{(t)}} - \overline{c_{pq}^{(t-1)}} | =| \overline{r_{pq}^{(t)}} - \overline{r_{pq}^{(t-1)}} | + | \overline{g_{pq}^{(t)}} - \overline{g_{pq}^{(t-1)}} | + | \overline{b_{pq}^{(t)}} - \overline{b_{pq}^{(t-1)}} | \qquad (2)$$

Finally, we count the number of blocks in frame t whose difference $d'^{(t)}_{pq}$ exceeds a threshold T_b. The inter-frame difference value $D_c^{(t)}$ is thus decided by the proportion of the blocks which are sufficiently different between each other.

$$D_c^{(t)} = n_{d'^{(t)}_{pq} \geq t} / (m \times n) \qquad (3)$$

If the $D_c^{(t)}$ is larger than another threshold T_c, we will declare the current frame as a cut boundary. Experiment results show that this block-wise comparison based method lessens the influence of global and local motion effectively.

After we got the differences between every consecutive frame pair, we can put them along the time axis to observe the temporal distribution of the frame difference values. Fig.2. shows the inter-frame pixel-wise difference values of a video sequence, which includes two cut transitions. The cut position can be clearly observed.

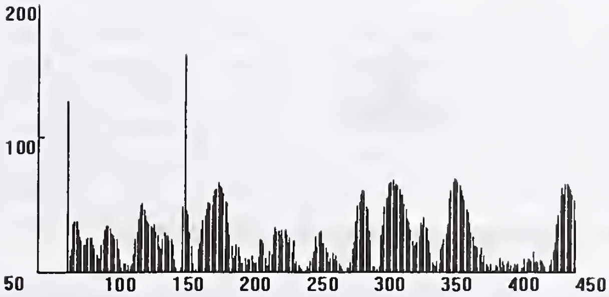


Fig.2. Frame-to-frame Different based on Block-wise Comparison

Two cut transitions occur at about frame #70 and frame #155 and the "jump up" is easy to figure out.

Since it is difficult for a global threshold to be suitable for any type of videos, even for different segments in one sequence, we adopt a sliding widow scheme for getting a self-adaptive threshold locally. It could be decided by (4) considering the local first several maximum values during previous *m* frames.

$$T^{(t+1)} = w_1 \times \frac{1}{n_0} \times \sum LMax \qquad (4)$$

where $LMax$ denotes the first n_0 maximum values during that sliding window, $\frac{1}{n_0} \times \sum LMax$ is the average difference and w_1 is the weight factor. By self-adaptive threshold, our method could adapt to the variation of motion intensity in video. When the motion is intense, the threshold will become higher; and when the motion slows down, the threshold will also drop.

3.2 Gradual Transition Detection

Due to the complexity of gradual transition, we extract color histogram from each frame as visual feature. Let $His^{(t)}$ denotes the color histogram of the *t*-th frame in RGB color space. Then we define a dissimilarity measure of successive frames based on histogram intersection as (5):

$$D_g^{(t)} = 1 - \frac{\sum_{i=1}^{N} MIN(His^{(t)}[i], His^{(t-1)}[i])}{m \times n} \qquad (5)$$

where $His^{(t)}[i]$ denotes the number of pixels falling into *i*-th bin, and $m \times n$ is the total number of pixels in one frame.

By observing the dissimilarity sequence in Fig. 3, we can see that the difference between the frames during the dissolve are higher, although only slightly, than those in the preceding and following frames. Moreover, it is a continuous region. Our task is to find the boundary of this region viz. the start and the end frame numbers. We propose a run-length based method to detect this kind of regions. First, we still need a self-adaptive threshold much lower than cut's threshold to detect the variation during gradual transition. If the dissimilarity of two successive frames is higher than this threshold, we count the frame number until this criterion is not met. We call this number as run-length. If the run-length reaches sufficient length, a gradual transition is declared. Otherwise, no gradual transition occurs. Unlike cut detection, we take into account the all of values in the previous sliding window except for those very high or very low values to decide threshold this time. It could be defined as (6)

$$T_g^{(t+1)} = w_2 \times \frac{1}{n_0} \times \sum LMedian \qquad (6)$$

where $LMedian$ are the median values in the sliding window, $\frac{1}{n_0} \times \sum LMedian$ is the average value and w_2 is the weight factor.

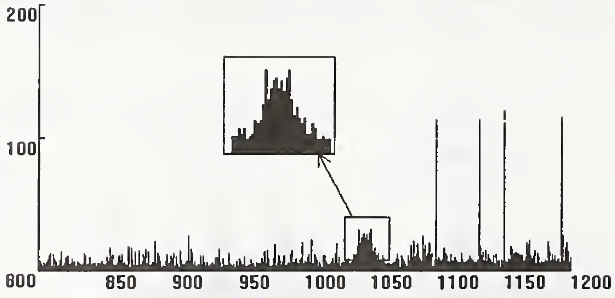
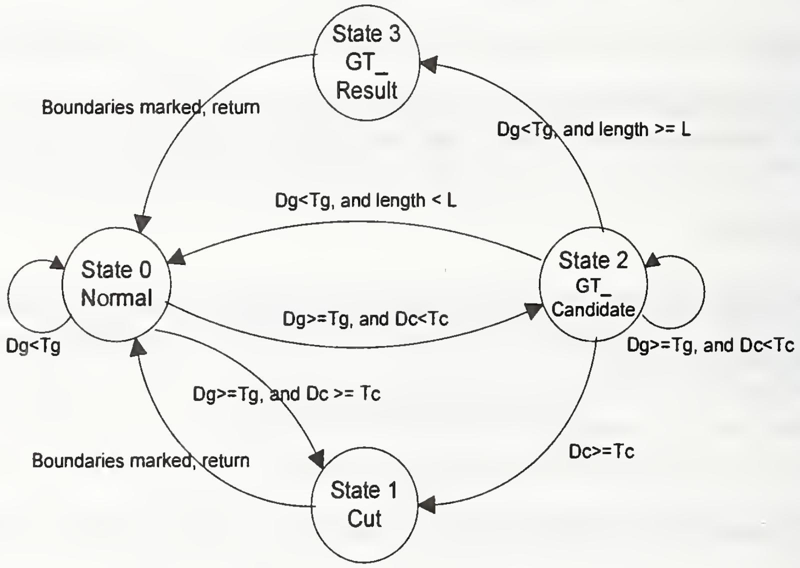


Fig. 3. Frame-to-frame Difference based on Histogram Intersection.
A graduate transition occurs from #1040 to #1060.

3.3 Integration

In the final decision module, the two detection algorithms above are integrated by finite state automata (FSA), shown as Fig. 4. There are 4 states in this FSA for each frame: State 0 is normal state, namely, no cut or gradual transition occurs in current frame; State 1 denotes a cut occurred in current frame; State 2 is a state during a suspect gradual transition; and State 3 denotes a gradual transition occurred from the frame enters State 2 to that enters State 3.



Dc: Inter-frame difference based on Block Pair-wise Comparison
Dg: Inter-frame difference based on Histogram Intersection
Tc: Cut threshold
Tg: Graduate transition threshold
L: Graduate transition minimum length

Fig. 4: Integration in Finite State Automata

When a detection process is started, the frame is assumed in State 0 by default. If $D_g<T_g$, FSA will keep in State 0. Otherwise we consider D_c. If $D_c>=T_c$, FSA will transfer to State 1, a cut is declared and FSA will go back to State 0. When $D_g>=T_g$ and $D_c<T_c$, FSA will transfer to State 2, entering a potential gradual transition, and the run-length detection process will be started. In this state, if the condition $D_g>=T_g$ and $D_c<T_c$ is satisfied, the state will be kept. When $D_g<T_g$, the run-length detection will finish. If the duration of run-length process is long enough, such as length>=L, FSA will jump to State 3, a gradual transition is declared, then FAS go back to State 0. When FAS is in State 2, there is another path to jump. If $D_c>=T_c$, FSA will jump to State 1, a cut will be declared, then FAS go back to State 0. The FSA will resume a new detection process, if only it goes back to Sate 0.

4. Experiments

Experiments are carried out on the test video sequences provided by NIST, and the results are also evaluated by NIST. NIST provided participants of video track about 11 hours video data. Among them, more than 5 hours' data are used as final test set including 42 video sequences. In this section, we pick out 16 video sequences with the bigger size from the NIST evaluation results to analysis the performance of our algorithms. As we can see, Table. 1 lists the general evaluation results considering cut and gradual transition as a whole. Table. 2 and Table. 3 list the evaluation results of cut and gradual transition detection respectively. Besides, we also test processing speed of our algorithm on PC with the configuration of PIII 450MHz, 256MB. The results are listed in Table. 4.

Table. 1 Evaluation Results: as a whole

Name	Frame Number	Reference Transition Count	Deletion rate	Insertion rate	Recall	Precision	Correction probability
ahf1.mpg	15679	107	0.102	0.168	0.897	0.842	0.948
anni005.mpg	11364	65	0.153	0.107	0.846	0.887	0.922
anni009.mpg	12307	103	0.368	0.155	0.631	0.802	0.814
bor03.mpg	48451	237	0.067	0.139	0.932	0.870	0.965
bor08.mpg	50569	528	0.094	0.140	0.905	0.865	0.951
bor12.mpg	24550	135	0.340	0.140	0.659	0.824	0.829
bor17.mpg	49801	246	0.337	0.150	0.662	0.815	0.830
eal1.mpg	16048	81	0.271	0.370	0.728	0.662	0.863
nad28.mpg	52927	298	0.161	0.177	0.838	0.825	0.918
nad31.mpg	52405	239	0.175	0.213	0.824	0.794	0.911
nad33.mpg	49768	214	0.046	0.168	0.953	0.85	0.976
nad53.mpg	25783	158	0.107	0.183	0.892	0.829	0.945
nad57.mpg	12781	67	0.208	0.208	0.791	0.791	0.894
pfm1.mpg	14686	82	0.134	0.231	0.865	0.788	0.932
senses111.mpg	86789	308	0.090	0.142	0.909	0.864	0.954
ydh1.mpg	22276	119	0.168	0.252	0.831	0.767	0.915
Average	34136	187	0.176	0.184	**0.823**	**0.817**	**0.910**

Table. 2 Evaluation Results: Cut

Name	Frame Number	Reference Transition Count	Deletion rate	Insertion rate	Recall	Precision	Correction probability
ahf1.mpg	15679	62	0.080	0.161	0.919	0.850	0.959
anni005.mpg	11364	38	0.026	0.052	0.973	0.948	0.986
anni009.mpg	12307	38	0.157	0.105	0.842	0.888	0.920
bor03.mpg	48451	226	0.061	0.044	0.938	0.954	0.968
bor08.mpg	50569	375	0.034	0.128	0.965	0.882	0.982
bor17.mpg	49801	126	0.087	0.015	0.912	0.982	0.956
eal1.mpg	16048	61	0.131	0.0	0.868	1.0	0.934
nad28.mpg	52927	181	0.160	0.077	0.839	0.915	0.919
nad31.mpg	52405	183	0.087	0.060	0.912	0.938	0.956
nad33.mpg	49768	188	0.010	0.037	0.989	0.963	0.994
nad53.mpg	25783	81	0.024	0.024	0.975	0.975	0.987
nad57.mpg	12781	44	0.227	0.022	0.772	0.971	0.886
pfm1.mpg	14686	61	0.098	0.081	0.901	0.916	0.950
senses111.mpg	86789	292	0.061	0.006	0.938	0.992	0.969
ydh1.mpg	22276	67	0.014	0.194	0.985	0.835	0.992
Average	34776	135	0.084	0.067	**0.915**	**0.934**	**0.957**

Table. 3 Evaluation Results: Gradual Transition

Name	Frame Number	Reference Transition Count	Deletion rate	Insertion rate	Recall	Precision	Correction probability
ahf1.mpg	15679	45	0.133	0.177	0.866	0.829	0.933
anni005.mpg	11364	27	0.333	0.185	0.666	0.782	0.833
anni009.mpg	12307	65	0.492	0.184	0.507	0.733	0.753
bor03.mpg	48451	11	0.181	2.090	0.818	0.281	0.908
bor08.mpg	50569	153	0.241	0.169	0.758	0.816	0.878
bor12.mpg	24550	135	0.340	0.140	0.659	0.824	0.829
bor17.mpg	49801	120	0.6	0.291	0.4	0.578	0.699
eal1.mpg	16048	20	0.7	1.5	0.3	0.166	0.649
nad28.mpg	52927	117	0.162	0.333	0.837	0.715	0.918
nad31.mpg	52405	56	0.464	0.714	0.535	0.428	0.767
nad33.mpg	49768	26	0.307	1.115	0.692	0.382	0.845
nad53.mpg	25783	77	0.194	0.350	0.805	0.696	0.902
nad57.mpg	12781	23	0.173	0.565	0.826	0.593	0.912
pfm1.mpg	14686	21	0.238	0.666	0.761	0.533	0.880
senses111.mpg	86789	16	0.625	2.625	0.375	0.125	0.687
ydh1.mpg	22276	52	0.365	0.326	0.634	0.66	0.816
Average	34136	60	0.346	0.714	**0.652**	**0.571**	**0.826**

Table. 4 Processing Speed Comparison

Name	Frame Number	Test time (s)	Normal time (s)	Speed up
ahf1.mpg	15679	367	540	1.47
anni005.mpg	11364	254	379	1.49
anni009.mpg	12307	287	410	1.43
bor03.mpg	48451	1115	1616	1.45
bor08.mpg	50569	1171	1687	1.44
bor12.mpg	24550	565	819	1.45
bor17.mpg	49801	1189	1661	1.40
eal1.mpg	16048	378	540	1.43
nad28.mpg	52927	1177	1766	1.50
nad31.mpg	52405	1250	1748	1.40
nad33.mpg	49768	1152	1660	1.44
nad53.mpg	25783	605	860	1.42
nad57.mpg	12781	297	426	1.43
pfm1.mpg	14686	342	495	1.45
senses111.mpg	86789	1998	2986	1.45
ydh1.mpg	22276	518	743	1.43
Average	34136	792	1146	**1.44**

By observing the evaluation results, it is concluded that:

1) The average probability of correct cut detection is more than 95% with more than 90% average recall and precision. This result indicates that our cut detection algorithm is very effective.
2) The average probability of correct gradual transition detection is more than 80% with about 60% average recall and precision. This result is satisfying considering the complexity of gradual transition. Because gradual transition consists of all kinds of non-cut transitions.
3) The average correction probability of all of transitions is more than 90% with more than 80% average recall and precision. It proves that integration method with finite state automata is much effective and efficient.

4) Comparing with the video playback time, the processing speed of our algorithm is much faster. Without any optimization, the processing speed could approach about 1.5 times of real-time on the PIII 450MHz 256MB personal computer.

On the other hand, we also find some mismatching between our ground truth and those provided by the organizer in the case of graduate transition. These mismatching affect our evaluation results to a certain degree. Some examples are listed in Table.5.

Table. 5. Some Mismatched Samples

Video Sequences	Mismatching in ground truth
bor08.mpg	#49326 to #49349
bor12.mpg	#16497 to #16520
bor17.mpg	#9679 to #9686; #9745 to #9752
nad28.mpg	#327 to #339, #2035 to #2057, #26591 to #26609, #37696 to #37709, #52197 to #52211
... ...	......

5. Conclusions

In this paper, we described our work in TREC-10 video track shot boundary detection task. We also reported and analyzed the evaluation results by NIST. The experimental results indicate that our shot boundary detection algorithm based on uncompressed domain is effective and much faster than real-time. By some optimizations, the speed of processing can be further improved.

However, there still is much room to improve our algorithm, especially for gradual transition. For example, some tolerances should be added into run-length based method. Because if the potential gradual transition state ends when only one inter-frame difference drops below the threshold *Tg*, some gradual transitions would be truncated. Another shortcoming is that the spans of gradual transition we detected are much longer than the real transitions sometime. Besides, how to integrate two detection algorithms also can still lead to additional improvements.

Acknowledgement

The authors would like to thank Dong Zhang for his helpful work on implementing some useful experimental tools.

References

[1] G. Ahanger and Thomas D.C. Little, "A Survey of Technologies for Parsing and Indexing Digital Video", *Journal of Visual Communication and Image Representation,* 7, 1, pp. 28043, 1996.

[2] H.J. Zhang, A. Kankanhalli, and S.W. Smoliar, "Automatic partitioning of full-motion video", *Multimedia Systems,* 1, pp.10-28, 1993.

[3] J. Meng, Y. Juan, S.-F. Chang, "Scene Change Detection in a MPEG Compressed Video Sequence", *Journal*, Publisher, Location, pp. 1-10, Date.

[4] B.L. Yeo and B. Liu, "Rapid scene analysis on compressed video", *IEEE Transactions on Circuits and Systems for Video Technology,* 5, 6, pp. 533-544, 1995.

[5] S.-B. Jun, K. Yoon, H.-Y. Lee, "Dissolve transition detection algorithm using spatio-temporal distribution of MPEG macro-block types", ACM Multimedia 2000, pp. 391-394, 2000.

[6] A. Nagasaka and Y. Tanaka, "Automatic Video Indexing and Full-Video Search for Object Appearances", *Visual Database Systems,* II, pp. 113-127, Elsevier Science Publishers, 1992.

[7] C.F. Lam, M.C. Lee, "Toward some innovative video scene change detection techniques", A CM Multimedia 1998, 1998.

Microsoft Cambridge at TREC–10: Filtering and web tracks

S E Robertson[*] S Walker[†] H Zaragoza[‡]

1 Summary

This report is concerned with the Adaptive Filtering and Web tracks. There are separate reports in this volume [1, 2] on the Microsoft Research Redmond participation in QA track and the Microsoft Research Beijing participation in the Web track..

Two runs were submitted for the Adaptive Filtering track, on the adaptive filtering task only (two optimisation measures), and several runs for the Web track, both tasks (adhoc and home page finding). The filtering system is somewhat similar to the one used for TREC–9; the web system is a simple Okapi system without blind feedback, but the document indexing includes anchor text from incoming links.

2 Okapi at TRECs 1–9

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 [5]. In TRECs 7–9 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, although with one limitation discussed below, and produced good results on both the TREC–9 measures, linear utility and the 'precision-oriented' measure. 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 The system

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 Okapi BSS is seen as a component of Keenbow. Many aspects of the system, including the weighting scheme and the query expansion methods used, reflect the various components of the probabilistic model of retrieval discussed at length in [7].

The Okapi Basic Search System (BSS), which has been used in all Okapi and Okapi/Keenbow TREC experiments, 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 [6, 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. There have been no major changes to the BSS during TREC–10.

Query expansion or modification methods using known or assumed relevant documents are generally built as scripts on top of the BSS. The range of methods used is described in our TREC–9 report [3]. A characteristic of the methods is that terms are selected for the modified query by a 'term selection value'.

The filtering system is again built as scripts on top of the BSS and lower-level scripts such as for query expansion. The incoming 'stream' of documents is divided fairly arbitrarily into batches. For each topic a current state is maintained, including query formulation, threshold etc., what happened at the last batch, and some history, including docids for any documents judged relevant up to now. As a new batch of documents is processed, the current query formulation of each topic is searched against it; cumulative databases are created, and each topic goes through the adaptation process in preparation for the next batch. Adaptation includes threshold adaptation as well as query modification.

3.1 Hardware

All the TREC–10 processing was done at Microsoft Research, Cambridge. Most of the work was 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.

[*] Microsoft Research Ltd, 7 J.J.Thomson Avenue, Cambridge CB3 0FB, UK, and City University, London, UK. email ser@microsoft.com

[†] Microsoft Research Ltd, 7 J.J.Thomson Avenue, Cambridge CB3 0FB, UK. email sw@microsoft.com

[‡] Microsoft Research Ltd, 7 J.J.Thomson Avenue, Cambridge CB3 0FB, UK. email hugoz@microsoft.com

4 Web track

4.1 Methods

We wanted to investigate the base performance of the Okapi BSS system when used with documents and queries originating from the WWW as opposed to other more homogenous collections. For the ad-hoc task, we experimented with word phrases (adjacent word pairs) and query expansion from blind feedback. For the Home-Page finding task we experimented with anchor text as well word phrases.

The methods used to select and weight phrases is the same one introduced in [8]. All adjacent word pairs are given a ***plausibility*** score and sorted, retaining word pairs above a certain threshold. This threshold was set to its usual value of 10. Word pairs were weighted by the weighs of its individual terms plus a small value constant across all terms (using a term-dependant weight yields similar results [8]).

The query expansion method used is also the same as in [8]. It was only used in one run, ad-hoc retrieval using all query fields (wtnd1). The top 15 documents were considered relevant for query expansion, and 20 terms were retained (forcing original query terms to be selected). the constant k3 in the BM25 formula, set to 0 for all runs, is increased after query expansion.

No HTML information was used, except for the extraction of anchor text. This was done using several heuristics to speed up the pre-processing of the data and reduce the size of the resulting index. First of all, the outlink records were used to index the relevant anchor text contained in every document of the collection. Then, using the inlink records, each document was augmented with the anchor text found in the links of other documents pointing to it. We limited the number of inlinks for each document, allowing at most 5 (chosen randomly when there were more available). Furthermore, a matching heuristic was used to parse relative addresses; full URL resolution was carried out only on potential links, greatly decreasing the pre-processing time (this also introduced some errors, estimated at less than 0.1%).

4.2 Web track results

Table 1 summarises the results obtained for the ad-hoc retrieval task. The use of 2-word phrases improved average precision slightly, while query expansion was clearly beneficial.

Table 2 summarises the results obtained for the home page finding task. The use of 2 word phrases lead to a slight increase in performance. The use of anchor text yielded more significant gains of performance, while decreasing the number of home pages not found by 2%. However, overall results are not impressive; using the Okapi system *off-the-shelf* for the task of home page finding is not ideal (in [2] different techniques are used to adapt the Okapi system to this task).

5 Adaptive Filtering

5.1 System and parameters

Last year's report [3] contains a fairly detailed account of the filtering system and the adaptation methods used, in particular the relation between the optimisation measure and the threshold. This year's system is very similar to last year's; Table 3 is an attempt to summarise the large number of parameters used.

5.2 Optimising Fbeta

The new Fbeta measure measure presents an interesting problem for optimisation. For last year's target-based measure, we used a fairly general optimising method, which involved estimating the value of the measure at different cut-off points in the collection-so-far, and then extrapolating to the estimated size of the final collection. In one respect this procedure is simpler for Fbeta, but in another it is more complex.

The simplicity arises from the fact that Fbeta is independent of absolute numbers of documents, in the sense that if every quantity (document count, such as total retrieved or total relevant) is doubled, Fbeta does not change. Thus the simplest estimate of Fbeta at a given score cutoff in the whole final collection is just the actual or estimated Fbeta at that score in the collection-so-far. This estimate assumes that the proportion of relevant documents does not change – in general this is the kind of assumption that is made in adaptive systems, although it might be possible for example to take account of a trend to get a better estimate.

The complexity arises from the fact that Fbeta depends on recall, which in turn depends on the total number of relevant documents in the collection. Unfortunately we do not usually know this even for the collection-so-far, let alone for the full final collection.

The methods that we have used in setting thresholds do give us some kind of handle on this question. As discussed in previous TREC reports and in [4], we calibrate the score to give an explicit probability-of-relevance estimate for each document. If this calibration is accurate, then (given a query and a collection) the total number of relevant documents in the collection is:

$$\sum_{\text{all docs}} p_{\text{doc}}$$

where p_{doc} is the calibrated probability of relevance of the document. While this would be quite a heavy calculation, we can select those documents with highest probabilities

Table 1: Ad-hoc task (WEB track).

Run	Avg. Precision	Run Description	Used for evaluation pool
ok10wt1	0.1908	title only	Yes
ok10wt3	0.1952	title only, word-pairs	Yes
	0.2028	title only, query expansion	
ok10wtnd0	0.2512	title and description	No
	0.2612	title and description, word-pairs	
ok10wtnd1	0.2831	title and description, query expansion	No

Table 2: Home page finding task (WEB track).

Run	Avg. Reciprocal Rank	Top 10	Not Found	Run Description
ok10whd0	0.312	58.6%	15.2%	plain
ok10whd1	0.340	60.7 %	15.9%	word-pairs
ok10wahd0	0.362	62.1%	13.1%	anchor text
ok10wahd1	0.387	64.1%	13.1%	word-pairs + anchor text

of relevance by simply retrieving the top-scoring documents in the usual fashion, and then assume that the remainder will contribute little to the total. Given that we know that many of the Reuters topics have substantial numbers of relevant documents, it is appropriate nevertheless to retrieve a substantial number for this purpose. The number chosen for the present experiments was 10% of the collection. The results below contain some comments on this limit.

It should be noted that the calibration of the score is intended primarily to provide accurate estimates of the probability of relevance of documents scoring around the chosen threshold. It is very likely that those much further down the ranking are very much less accurate. (More specifically, the calibration assumes a linear model on the log-odds scale, and adapts the intercept but not the slope, to fit the documents of known relevance, which are those that have already been retrieved and are therefore high up the ranking. It is likely that the slope is critical for accurate estimation of the probabilities lower down the ranking.) Thus we should not expect the estimated total relevant to be very accurate either. Some experiments to determine the accuracy of these estimates are described below.

Given an estimate of the total number of relevant documents, we can estimate Fbeta for any score threshold by discovering the number retrieved at that threshold and estimating the number relevant. We can then choose the score threshold which maximises Fbeta. Because of the assumed independence of Fbeta from the absolute number of documents, we perform this procedure for each topic on the collection-so-far, and the resulting threshold is a candidate for applying to the next batch.

As with the utility measure, we have to start the process off. We do this (again as with utility) by starting with a target based simply on the number of documents to be retrieved over the period. Once we have retrieved a few relevant documents, we can consider an Fbeta-based threshold. However, there remains a danger that an early and inaccurate Fbeta-based threshold will be too stringent and will not retrieve any more documents. We therefore push up the target-based threshold gradually, using the ladder method discussed in previous TREC reports, until it is higher than the Fbeta-based threshold.

5.3 The accumulated collection

As in previous years, we assume that we accumulate the documents as they come into the system, so that we always have a cumulated collection of everything received up to now. Such a collection is needed for some of the forms of adaptation discussed; in TREC–10 (as in TREC–9), we actually need two such collections, respectively including and excluding the training set.

5.4 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 Reuters collection is batched by day (as it comes on the CD), with batch 0 as the training set. However, for efficiency reasons several days' documents may be searched without the adaptive procedures being initiated. The adaptive procedures were initiated every day for the first 10 days, every three days up to 40 days, and every 10 days thereafter.

For similar reasons, query modification is done after any batch in which a new checkpoint is reached for the particular topic. In these experiments, the checkpoints are defined in terms of the number of relevant documents

Table 3: Parameters for adaptive filtering

See notes below and [3] for explanations of these parameters	
BM25 parameters:	
k_1	1.2
b	0.8
Score calibration:	
Initial beta	-0.68
Mythical reldocs for beta re-calibration	3
Gamma	2.7
Threshold adaptation:	
Initial target no. of documents	100
Ladder step	0.2
Query modification:	
Reldocs used for modification	100
Maximum terms	25
Minimum terms	3
Absolute term selection value threshold	5

Notes

For both the Utility and Fbeta measures, the initial threshold is set to retrieve a specified target number of documents over the whole period of the test set.

For Utility, the threshold calibrated as a log-odds probability is raised by one ladder-step for each relevant document retrieved, until it reaches the level defined by the utility function.

For Fbeta, again the threshold is raised by one ladder-step for each relevant document. However, after 4 relevant documents have been retrieved, the ladder threshold is compared with the estimated optimum Fbeta threshold, and the lower of the two is chosen.

Query modification is based on the original text query, the training sample and the reldocs retrieved so far. If there are more reldocs than the limit, only the most recent are taken.

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.

retrieved so far, and are set at 1,2,4,8,16... relevant documents.

So the basic procedure is as follows: for each batch (day) i of incoming documents

1. Run current profiles against batch i
2. Update both cumulative databases (batches 0–i and batches 1–i)
3. If adapt today, then for each topic:
   (a) if new checkpoint has been reached,
      - reformulate query
      - recalculate scores of previously retrieved documents and re-estimate β (needed for score calibration)

   (b) set threshold (using methods described above)

5.5 Filtering results

As with the official track results, the measures reported are T10SU (scaled utility), T10F (van Rijsbergen's Fbeta measure with beta=0.5), set precision and set recall.

Submitted runs

Just two runs were submitted, one optimised for each of the two main measures, T10SU and T10F. The results are shown in Table 4.

These results are somewhat disappointing. The most obvious problem is that the run optimised for T10F actually does better on T10SU than the run optimised for T10SU. Some examination of the results suggests that in quite a number of topics, the profile, having retrieved a small number of poor documents initially, gets locked into a state with a high threshold, and fails to retrieve any other documents. Although this is exactly one of the conditions that the procedures were designed to avoid, it seems that they have failed on several occasions. The problem is exacerbated by the fact that many of these topics actually have very large numbers of relevant documents – thus once out of this trap, the optimisation could be expected to do very well.

The T10F optimisation appears to fall into this trap less often. A pragmatic response might be to start the utility runs off with an Fbeta target rather than a document number target, and only switch to the utility target

Table 4: Official results for submitted runs

Run	Optimisation measure	T10SU	T10F	Set Precision	Set Recall
ok10f2br	T10F	0.137	0.330	0.427	0.273
ok10f2ur	T10SU	0.104	0.254	0.368	0.214

when a reasonable number have been retrieved. However, it is clear that some rethinking is required.

Estimating the number of relevant documents

Some experiments were done with the TREC-9 filtering collection (based on OHSUMED) and the Fbeta measure in order to assess the estimation of the total number of relevant documents in the collection. We first look at the estimate given towards the end of the process, that is for the next-to-last batch. Note that the system attempts to estimate the total relevant only when it has retrieved some relevant documents; initially, when it is using the target method or the ladder, no estimate is required or made. In the OHSUMED collection with the OHSU queries, a number of topics never reach that stage. Table 5 shows the estimates from the next-to-last batch and the first ten topics, compared with the actual total number of relevant in the collection, for a typical run (one of many trial runs).

Table 5: Estimating total relevant – OHSU

Topic	Estimated R	Actual R
1	46.6	44
2	23.2	44
3	168.9	165
4	(no estimate)	12
5	139.8	44
6	(no estimate)	27
7	7.9	19
8	(no estimate)	11
9	50.1	44
10	(no estimate)	19

These estimates were moderately promising but not wonderful – most are in the right ball-park, but topics 5 and 7 are fairly poor. The corresponding estimates for the TREC–10 Reuters data, the official ok10f2br run, are as in Table 6

Here the obvious problem topic is number 2, where the estimate is out by a factor of approximately 200. This is an instance of the problem mentioned above, where the profile, after retrieving a very small number of documents at the beginning, got stuck with a too-high threshold and failed to retrieve anything else. The others, while being within an order of magnitude of the correct figure, nevertheless may diverge from it by a factor of up to 4.

Table 6: Estimating total relevant – Reuters

Topic	Estimated R	Actual R
1	10,342	23,651
2	59	11,563
3	9,041	36,463
4	12,109	7,250
5	26,973	22,813
6	3,316	1,871
7	18,388	17,876
8	27,871	11,202
9	9,180	2,560
10	17,165	5,625

If we look at the progress of the estimates over the time-period of the simulation, it is clear that they may be more wildly out near the beginning. Visual inspection of the OHSU data suggested that the early estimates were often (not always) much too high. However, they tended to move into the right order of magnitude fairly quickly.

There are a number of variables whose effects on the estimates have not been investigated. One of these is the cut-off number of documents used for the estimate (see section 5.2 above. It appears (impression only) that we may get better estimates by limiting this number further. This does suggest that the linear calibration model, with fixed slope, is not working very well further down the ranking.

This observation is interesting, because it gives us another way of testing the calibration. It seems clear that, for this purpose at least, we need to adjust the slope as well as the intercept. However, it is not at all clear how this may be done.

5.6 Computational load

We commented last year that with about 5000 MeSH topics the OHSU filtering task was computationally very heavy. Even with less than 100 topics, the same comment can be made about the Reuters filtering task. Partly this is because many of the Reuters topics have very large numbers of relevant documents, and therefore retrieved sets tend also to get large. But the general point remains: adaptive filtering is computationally very much more demanding than adhoc searching.

6 Conclusions

The web track presents some interesting challenges. The use of anchor text of incoming links as a way of providing additional information about a page clearly has potential, but we have only scratched the surface in the present experiments. The tiny benefit from phrases (word pairs) and the somewhat greater benefit from blind feedback are both consistent with previous experiments.

The adaptive filtering task continues to be an interesting and fruitful one to investigate. The new Fbeta optimisation measure has been interesting, particularly because of the need to estimate the total number of relevant documents in the collection. Clearly further work in this area is required. Also, despite our attempts of the last three years to devise a robust system for setting and adapting thresholds, the Reuters collection has presented conditions which under some circumstances cause our methods to fail.

References

[1] Brill, E., Lin, J., Banko, M., Dumais, S. and Ng, A. Data-intensive question answering. In these proceedings.

[2] Gao, J., Gao, G., He, H., Walker, S., Robertson, S., Zhang, M. and Nie, J-Y. TREC-10 Web Track Experiments at MSRCN. In these proceedings.

[3] Robertson, S.E. and Walker, S. Microsoft Cambridge at TREC-9: Filtering track. In: [9], p361–xxx.

[4] Robertson, S.E. and Walker, S. Threshold setting in adaptive filtering. *Journal of Documentation*, 56, 2000 p312–331.

[5] Robertson, S.E. and Walker, S. Okapi/Keenbow at TREC-8. In: [10], p151–162.

[6] Robertson, S.E. *et al.* Okapi at TREC–3. In: [12], p109–126.

[7] Sparck Jones, K., Walker, S. and Robertson, S.E. A probabilistic model of information retrieval: development and comparative experiments. *Information Processing and Management*, 36, 2000, p779–808 (Part 1) and 809–840 (Part 2).

[8] Beaulieu, M.M. *et al.* Okapi at TREC–5. In: [11], p143–165.

[9] *The Ninth Text REtrieval Conference (TREC-9).* Edited by E.M. Voorhees and D.K. Harman. Gaithersburg, MD: NIST, 2001 (NIST Special Publication 500-249).

[10] *The Eighth Text REtrieval Conference (TREC-8).* Edited by E.M. Voorhees and D.K. Harman. Gaithersburg, MD: NIST, 2000 (NIST Special Publication 500-246).

[11] *The Fifth Text REtrieval Conference (TREC-5).* Edited by E.M. Voorhees and D.K. Harman. Gaithersburg, MD: NIST, 1997.

[12] *Overview of the Third Text REtrieval Conference (TREC-3).* Edited by D.K. Harman. Gaithersburg, MD: NIST, 1995 (NIST Special Publication 500-225).

TREC-10 Web Track Experiments at MSRA

**Jianfeng Gao*, Guihong Cao#, Hongzhao He#, Min Zhang##,
Jian-Yun Nie**, Stephen Walker*, Stephen Robertson***

* Microsoft Research, {jfgao,sw,ser}@microsoft.com
** Département d'informatique et de recherche opérationnelle, Université de Montréal,
nie@iro.umontreal.ca
#Department of Computer Science and Engineering of Tianjin University, China
State Key Lab. of Intelligent Tech. & Sys., Computer Science. &Tech. Dept, Tsinghua University, China

Abstract

In TREC-10, Microsoft Research Asia (MSRA) participated in the Web track (ad hoc retrieval task and homepage finding task). The latest version of the Okapi system (Windows 2000 version) was used. We focused on the developing of content-based retrieval and link-based retrieval, and investigated the suitable combination of the two.

For content-based retrieval, we examined the problems of weighting scheme, re-weighting and pseudo-relevance feedback (PRF). Then we developed a method called *collection refinement (CE)* for QE.

We investigated the use of two kinds of link information, link anchor and link structure. We used anchor descriptions instead of content text to build index. Furthermore, different search strategies, such as spreading activation and PageRank, have been tested.

Experimental results show: (1) Okapi system is robust and effective for web retrieval. (2) In ad hoc task, content-based retrieval achieved much better performance, and the impact of anchor text can be neglected; while for homepage finding task, both anchor text and content text provide useful information contributing more on precision and recall respectively. (3) Although query expansion does not show any improvement in our web retrieval experiments, we believe that there are still potential for CE.

1. Introduction

Microsoft Research Asia (MSRA) participated in the Web track (ad hoc retrieval task and page finding task) at TREC-10. We used, for the first time, the new version of the Okapi system (which is running on Windows-2000) developed at Microsoft Research Cambridge. We focused our researches on: (1) the use of traditional IR techniques (content-based retrieval) for web retrieval, (2) the use of query expansion (QE) for web retrieval, and (3) the use of link information.

In this paper, we will explore the following issues:

(1) Testing the Windows version of the Okapi system using 10GB web collection.

(2) The impact of query expansion on web retrieval. The expansion terms are chosen from the top-ranked documents retrieved using the initial queries. We used two types of collections for initial retrieval: the 10G web collection and an external collection, i.e. the MS-Encarta collection.

(3) The relative contribution of content information and link information to web retrieval. We exploit methods of combining both kinds of information to improve the effectiveness of web retrieval.

(4) The impact of link information on web retrieval. We investigate the use of two kinds of link information: link anchor text and link connection.

In the remainder of this paper, we will discuss in turn each problem together with our approaches and results of TREC experiments. The results include official runs we submitted and additional runs that we designed to help us explore the issues. Finally, we give our conclusions and present our future work.

2. The System

We used the Okapi system Windows-2000 version for our runs. The system was developed in October 2000. A detailed summary of the contributions to TREC1-9 by the Okapi system is presented in (Roberson and Walker, 2000; Roberson and Walker, 1999). In this section, we give a very brief introduction to the system.

The search engine in Okapi is called the Basic Search System (BSS). It 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 scheme functions known as BM25 and its variants. 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. BSS also provides functions for blind feedback.

All the TREC-10 processing was done at Microsoft Research China. Most of the experiments were run on two DELL severs. Both machines have four 500MHz Pentinum processors with 2GB RAM, and were running on Windows-2000. The network was 100Mbps Ethernet.

3. Data Processing

The collection we used in Web track is a set of web pages downloaded from the World Wide Web. The size of the original collection is more than 10GB. It is a good challenge for the new version of the Okapi system. Four query sets were used in our experiments:

(1) TREC-9 ad hoc query set of 50 queries (denoted by T9),

(2) TREC-10 ad hoc query set of 50 queries (denoted by T10),

(3) TREC-10 page finding query set of 145 queries (denoted by P10), and

(4) A page finding training set, which includes 100 queries (the query set is denoted by P9) and the relevance judgment.

3.1 Pre-processing

Our data pre-processing includes data cleaning and information extraction.

We first removed junk from the collection. The junk includes mismatched tags, and files which contain non-text material (i.e. compressed data, etc). All lines starting with "Sever:" and "Content-type" etc. were also removed. The resulting collection is of size 6GB.

We then used an HTML parser developed at Microsoft to extract logical fields, including Title <T>, Subtitle <H1>, <H2> and <H3>, and Passage delimited by tags <P> and </P>. We also, from the collection, established two tables. One table contains the link connection information; each entry of the table is a page-pair connected by a link. The other contains link anchor text information; each entry includes an anchor text, the page containing the anchor text, and the page pointed by the anchor text. Title, Subtitle and Passage were used for content-based retrieval while the link connection and link anchor text were used for link-based retrieval. We will describe both retrieval methods in detail later.

3.2 Indexing/Query processing

For query processing, we first performed stemming using the Okapi stemmer. Stop words were then removed. We used a stop word list of 222 words (Roberson and Walker, 1999). For four query sets, we used title-only queries in our experiments.

For each web page, before indexing, all words were stemmed, and stop words were removed. The term weight is BM2500. It is a variant of BM25 and has more parameters that we can tune. BM2500 is of the form:

$$\sum_{T \in Q} w^{(1)} \frac{(k_1 + 1)tf\ (k_3 + 1)qtf}{(K + tf\ )(k_3 + qtf\ )} \qquad (1)$$

where Q is a query containing key terms T, tf is the frequency of occurrence of the term within a specific document, qtf is the frequency of the term within the topic from which Q was derived, and $w^{(1)}$ is the Robertson/Spark Jones weight of T in Q. It is calculated by Equation (2):

$$\log \frac{(r+0.5)/(R-r+0.5)}{(n-r+0.5)/(N-n-R+r+0.5)} \qquad (2)$$

where N is the number of documents in the collection, n is the number of documents containing the term, R is the number of documents relevant to a specific topic, and r is the number of relevant documents containing the term. In Equation (1), K is calculated by Equation (3):

$$k_1((1-b)+b \times dl/avdl) \qquad (3)$$

where dl and $avdl$ denote the document length and the average document length measured in some suitable unit, such as word or a sequence of words.

Parameters k_1, k_3, b, and $avdl$ are tuned by experiments to optimize the performance. In our experiments, we set k_1=1.2, k_3=1000, b=0.75, and avd=61200.

4. Basic Content-based Retrieval

For basic content-based retrieval, only initial retrievals (i.e. without QE) were performed. Only those words in fields of Title/Subtitle/Passage were indexed. The initial retrieval results are summarized in Table 1 and 2. We can see that the ad hoc retrieval results for TREC-9 query set are very promising. It is favorably comparable to the best effectiveness achieved in the previous web track experiments. This indicates the robustness and effectiveness of our new version of the Okapi system. The results in Table 1 and 2 will also serve as the baseline in all experiments described below. The evaluation metric of ad hoc task is non-interpolated average precision. The evaluation metrics of page finding task includes average reciprocal rank, top-10 precision, and not-found rate.

Query set	Avg. P.
T9	22.08%
T10	19.42%

Table 1: Baseline results of ad hoc task

Query set	Average reciprocal rank	Top-10 precision	Not-found rate
P9	19.68%	34.00%	25.00%
P10	22.46%	44.10%	25.52%

Table 2: Baseline results of page finding task

5. Query Expansion

The average length of title-only queries is less than 3 words (non stop word). It seems that query expansion is needed to deal with word mismatching problem for web retrieval. We performed query expansion experiments on ad hoc retrieval. The procedure works as follows:

(1) For each query, retrieve 10 top ranked documents by an initial retrieval;

(2) Choose 10 expansion terms from the top ranked documents. First, stop words were discarded. Then expansion terms were ranked in decreasing order of a term selection value (TSV) of the form

$$TSV = w^{(1)} * r / R \tag{4}$$

where $w^{(1)}$, R, and r are same elements described in Equation (1) and (2). The top-10 terms were added to the initial query.

As shown in Table 3, the conventional query expansion (i.e. pseudo-relevance feedback (PFB)) result is not good. We think that there might be two reasons. First, the topics of web pages are diverse. Although the expansion terms were chosen from top ranked documents, the ranking of these terms was based on the statistics over the whole collection as indicated by Equation (4). Second, the quality of documents in the web collection is highly mixed.

We adopted two methods to solve the abovementioned two problems..

First, we introduced the local context analysis (LCA) technique as proposed in (Xu and Croft, 1996). We used statistics of documents and terms from local collection (i.e. top-10 ranked document collection retrieved by an initial retrieval) to estimate the TSV for each expansion terms. That is, we set R=10, and r is the number of relevant documents containing the term in the local collection (r<10). As shown in Table 3, although the result is a little better than PRF, it is still worse than the initial retrieval.

Second, we introduced the idea of collection enhancement (CE), which was successfully applied for TREC cross language information retrieval experiments (Kwok et al., 2000). The basic idea is: if we can refine web queries by QE using documents from an external high-quality and well-organized collection, we may able to improve the web retrieval. We used MS-Encarta collection as the external collection. That is, in our experiments, the initial retrieval was performed using MS-Encarta collection. The expansion terms were chosen from the top-1 Encarta document. Notice that we did not use top-10 documents because the MS-Encarta collection is relatively small (i.e. less than 200MB). More importantly, MS-Encarta is a well-written encyclopedia with each document discussing one specific topic. So terms from multiple documents are likely of different topics and not relevant. Documents in MS-Encarta are categorized by a set of pre-define keywords and are well-organized under a domain hierarchy structure. We think that such information will be helpful for navigating the web collection, but we have not found an effective way to make use of them.

The preliminary result shown in Table 3 is not encouraging. We found that it is largely due to the difference between the Encarta collection and the web data. But we do believe it has potential if we can make good use of rich information imbedded in MS-Encarta collection (i.e. pre-defined keywords, domain hierarchy, etc.) or figure out an effective way to fuse the web collection with MS-Encarta collection.

Initial retrieval	PRF	LCA	CE
22.08%	20.89%	21.55%	18.24%

Table 3: QE results of TREC-9 ad hoc retrieval

6. Link-based Retrieval and Content-based Retrieval

Recently, the research of web retrieval has focused on link-based ranking methods. However, none had achieved better results than content-based methods in TREC experiments. We investigated the use of two kinds of link information: link anchor and link connection. Our focus was on finding the effective ways for combining link-based retrieval with content-based retrieval.

6.1 Using anchor text

We assumed that the anchor text of a link describes its target web page (Craswell et al., 2001). We then, for each web page, built an anchor description document containing all the anchor texts of a page's incoming links.

We observed that plenty of anchor texts are names of home pages (i.e. URL, or URL-like terms), which are reasonable. Therefore intuitively, they could be very effective for page finding task, in which most queries are also a bunch of URLs or URL-like terms. Our results on TREC-10 page finding tasks confirmed the intuition. In Table 4, row 1 and row 2 show that anchor-text-based retrieval achieved much better performance than content-based retrieval, i.e. more than 96% improvements on average reciprocal rank and 48% improvements on top-10 precision.

	Average reciprocal rank	Top 10 precision	Not found	Method
1	22.46%	44.10%	25.52%	Content-based retrieval
2	44.06%	65.50%	25.52%	Anchor-text-based retrieval
3	42.40%	65.50%	13.10%	Content + anchor text (Comb-1)
4	50.50%	69.00%	15.20%	Content + anchor text (Comb-3)

Table 4: Page finding results of TREC-10 query set (P10)

We then performed experiments on ad hoc retrieval using anchor description only for indexing. The results are much worse than content-based retrieval as shown in Table 5 (row 1-3). This is due to the data sparseness problem. As indicated in Figure 1, statistics showed that about 28% web pages in the web collection have no anchor description at all. Totally 75% web pages have anchor description documents with less than 10 words. Therefore, the information (in terms of keywords) that anchor text provided for ad hoc retrieval is very limited. It is most unlikely that the title-only queries have chances to match words contained in such a short description. So even with query expansion (e.g. chose 30 terms from top-5 ranked documents), anchor-text-based retrieval was still much worse than content based retrieval although it was much better than the result without query expansion as shown in row 1-3 of Table 5.

	Avg. P using T9	Avg. P. using T10	Method
1	20.08%	19.42%	Content-based retrieval
2	3.12%	--	Anchor-text-based retrieval
3	4.85%	--	2 + query expansion
4	22.23%	19.13%	Content + anchor text (Comb-1)
5	23.27%	18.64%	Content + anchor text (Comb-3)

Table 5: Ad hoc retrieval results using anchor text

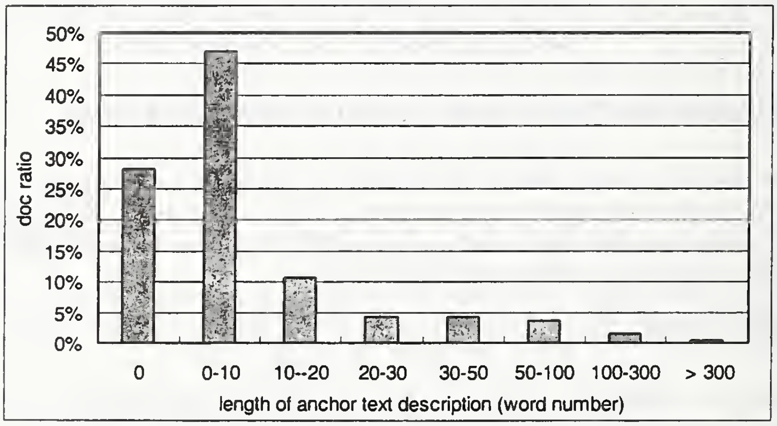


Figure 1: Anchor text description length vs. number of documents

In what follows, we examine three different ways to make use of anchor text and content text.

First, we simply combined the content text and anchor text for indexing (denoted by Comb-1 in Table 4 and Table 5).

Second, we merged the two ranking lists obtained by content-based retrieval and anchor-text-based retrieval respectively. The new score, *s*, of a retrieved page is estimated by Equation (5).

$$s = \lambda * sc + (1-\lambda)*sa \qquad (5)$$

where *sc* and *sa* are, respectively, the scores of content-based retrieval and anchor-text-based retrieval, and λ $(0 \leq \lambda \leq 1)$ is the interpolation weight tuned on a test set.

The last method we used is to re-rank the results of content-based retrieval according to the results of anchor-text-based retrieval (denoted by Comb-3 in Table 4 and 5). For each retrieved page in the ranking list of content-based retrieval, if it is also included in the ranking list of anchor-text-based retrieval, we set a new score by Equation (6), where $\lambda \geq 1$.

$$s = \lambda * sc \qquad (6)$$

The page finding results are summarized in Table 5. The ad hoc retrieval results are summarized in Table 6. Let us discuss the frustrating ad hoc retrieval results first.

As we expected, since the addition indexing words provided by anchor text are very limited, the impact of combination is neglectable, as shown in row 4 of Table 5. Similarly, in the second method, we found that the best result is obtained when λ=1. This indicates again the neglectable impact of anchor text information on ad hoc retrieval. For Comb-3, we still found that the best results are obtained when λ approached 1.

The reason that anchor text is not helpful to ad hoc retrieval is largely due to the sparseness problem we discussed above. The following Figure 2 shows the results of query by query analyses on TREC-10 ad hoc retrieval task. Because most of the anchor texts are too short, they are submerged in the content data. At the same time, what most commonly happens is that the query and the anchor text are mismatched for both of them are extremely short. Since no good result can be achieved by using anchor description of the document only, no improvement may be obtained by combining anchor text retrieval result and content text retrieval results.

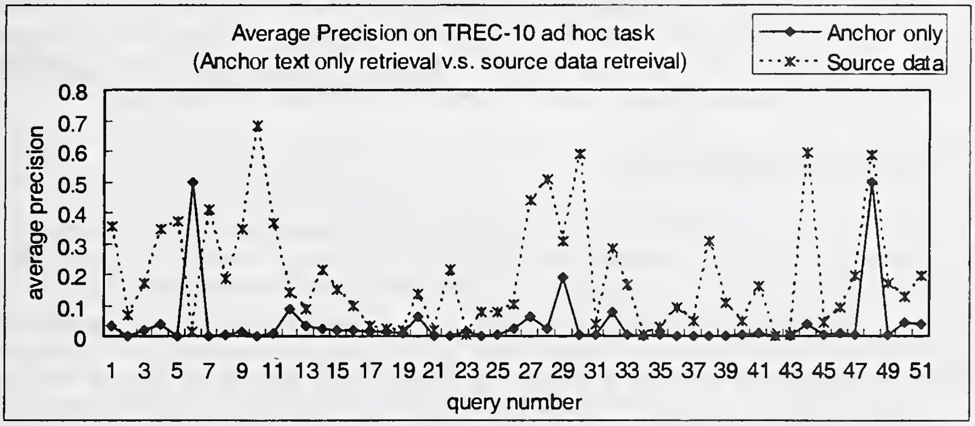


Figure 2: Average precision on TREC-10 ad hoc task (using anchor text v.s. source data)

Now let's look at the good results achieved in page finding task by using both content text and anchor text.

We applied Comb-1 and Comb-3. Unlike the ad hoc retrieval, experimental results on page finding task are much more encouraging as shown in Table 4. Row 3 shows that when using Comb-1, although we did not get any improvements on average reciprocal rank and top-10 precision, the not found rate dropped dramatically by more than 48%. As shown in row 4, by using Comb-3, we obtained even better performance. We achieved approximately 15% improvement on average reciprocal rank, and 56% improvement on top-10 precision. The not found rate also dropped substantially by more than 40%.

We give the similar query by query analyses for TREC-10 homepage finding task, the first 50 queries of which are shown in Figure 3. That is to say, we evaluate the retrieval result by non-interpolated 11 points average precision metric, which shows the performance in terms of both precision and recall. For the remaining queries, the results are most similarly. On a whole, there are 90 queries that can get better performance by using anchor description than using source data for indexing; only 38 queries are worse than source data retrieval; and for the remaining 17 queries, both anchor description retrieval and source date retrieval get the same results. Since anchor text takes limited but precise information of a homepage, especially the URL feature of the page, it can get better performance. Then it is reasonable to make improvements while combining two different ranked lists of retrieval results.

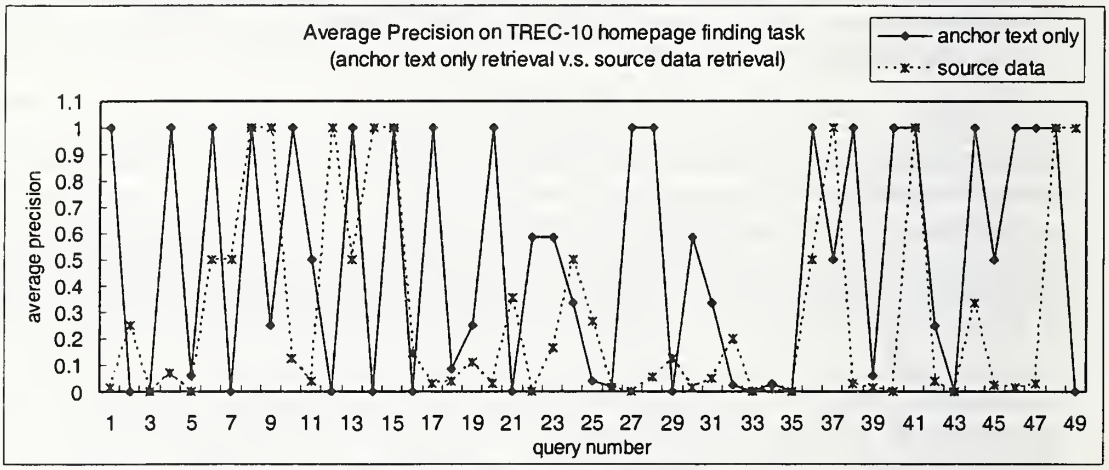


Figure 3: Average precision on TREC-10 homepage finding task (using anchor text v.s. source data)

The results indicate that (1) anchor text containing less but URL-like terms which contributed more to the precision of page finding; (2) content text with more terms might contribute more to the recall; and (3) when we combined anchor text and content text for indexing, both kinds of information really complemented each other, and achieved a better tradeoff between precision and recall.

6.2 Using link connection

We assumed that links between web pages indicate useful semantic relationships between related web pages. Especially, we tried spreading activation (SA) approach (Crestani and Lee, 2000; Savoy and Rasolofo, 2000) for ad hoc task using TREC-9 query set. In SA method, the degree of match between a web page D_i and a query Q, as initially computed by the IR system (denoted $SIM(D_i,Q)$), is propagated to the linked documents through a certain number of cycles using a propagation factor. Savoy and Rasolofo (2000) used a simplified version with only one cycle and a fixed propagation factor λ for k-best incoming links and k-best outgoing links. Our experiments showed that considering outgoing links negatively affects the retrieval results. Therefore only the top-1 similar incoming link is considered in our methods. In this case, the final retrieval value of a document D_i with m incoming linked documents is computed as:

$$\text{SAscore}(D_i) = SIM(D_i, Q) + \lambda \cdot \max\{SIM(D_j, Q) \mid j = 1,...m\} \tag{7}$$

Unfortunately, we found that the best result could be obtained only when λ approached 0.

In addition to SA, different search strategies, such as PageRank etc have been tested. However, none was able to improve the retrieval effectiveness. This result confirmed the previous results in TREC using link connections.

7. Summary of Official Retrieval Results

In TREC-10, we submitted 5 official runs for ad hoc task, and 4 runs for page finding task. In both tasks, title-only query sets were used. Table 6 and 7 show the results as well as methods we used for our submitted runs.

Run #	Avg. P	Method
Msrcn1	19.42%	Content-based retrieval
Msrcn2	19.13%	Content + anchor text (Comb-1)
Msrcn3	18.64%	Content + anchor text (Comb-3)
Msrcn4	17.79%	Content + anchor text (Comb-3) +LCA
Msrcn5	18.80%	Content + anchor text (Comb-1) + PRF

Table 6: Ad hoc official results of submitted runs

Run#	Average reciprocal rank	Top 10 precision	Not found	Method
Msrcnp1	22.46%	44.10%	25.52%	Content-based retrieval
Msrcnp2	42.40%	65.50%	13.10%	Content + anchor text (Comb-1)
Msrcnp3	44.06%	65.50%	25.52%	Anchor-text-based retrieval
Msrcnp4	50.50%	69.00%	15.20%	Content + anchor text (Comb-3)

Table 7: Page finding official results of submitted runs

8. Conclusions and Future Work

In this paper, we described our work in the Web track (ad hoc retrieval task and page finding task) evaluated at TREC-10. We used the latest version of the Okapi system (Windows 2000 version), and focused our researches on: (1) the use of traditional IR techniques (content-based retrieval) for web retrieval, (2) the use of query expansion, and (3) the use of link information.

Several conclusions are suggested by our experiments.

(1) The new version of the Okapi system was shown to be very robust and effective for web retrieval.

(2) In ad hoc task, content-based retrieval achieved much better performance than link-based retrieval. This confirmed again the previous Web track results at TREC.

(3) In ad hoc task, the impact of anchor text could be neglected. This might be due to the problem of the nature of the TREC web collection, such as the sparseness problem mentioned in Section 6. Other groups have reported that similar methods can achieve improvements on other web collection than TREC collection (Dumais and Jin, 2001).

(4) In page finding task, anchor-text-based retrieval achieved much better results than content-based retrieval in spite of much less terms contained in the anchor description. This is perhaps because terms in page finding queries and anchor text are very similar (i.e. URL, or URL-like terms).

(5) Both anchor text and content text provided useful information for page finding. In particular, anchor text contributed more to the precision of page finding, while content text contributed more to the recall. Both kinds of information complemented each other. The combination thus achieved a better tradeoff between precision and recall.

(6) Although query expansion did not show any improvement in our web retrieval experiments, we think that there are still potential for CE if we can make good use of other rich information imbedded in the well-organized high-quality external collection (MS-Encarta) or figure out an effective way to combine the web collection with the external collection.

Our future work includes

(1) Study the nature of the web collection, and exploit the use of link information on a more 'complete' web collection.

(2) Enrich the anchor description by using context information of the anchor. The context information can be a sentence or a passage that contains the anchor text. The context information may enhance the anchor description from two aspects: (1) providing clues to evaluate the relevance between the anchor text and its target web page; (2) providing richer description of the target web page.

(3) Exploit the use of the external collection for QE including the use of information of domain hierarchy, pre-defined keywords, and topics etc, and the effective combination of external collection with web collection, etc.

References

Bhrat, K., and Henzinger, M. R., (1998). Improved Algorithms for Topic Distillation in a Hyperlinked Environment. In *SIGIR-98*.

Brin, S., and Page, L. (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *WWW7*.

Craswell, N., Hawking, D., and Robertson, S.E. (2001). Effective site finding using link anchor information. In *SIGIR-01*.

Crestani, F., and Lee, P. L. (2000). Searching the web by constrained spreading activation. *Information Processing & Management*, 36(4), page 585-605.

Dumais, S., and Jin, R. (2001). Probabilistic combination of content and links. In *SIGIR-01*.

Kowk, K. L., Grunfeld, N., and Chan, M. (2000). TREC-9 cross language, web and question-answering track experiments using PIRCS. In *TREC-9*.

Robertson, S. E., and Walker, S. (1999). Okapi/Keenbow at TREC-8. In *TREC-9*.

Robertson, S. E., and Walker, S. (2000). Microsoft Cambridge at TREC-9: Filtering track. In *TREC-9*.

Savoy, J., and Rasolofo, Y. (2000). Report on the TREC-9 Experiment: Link-Based Retrieval and Distributed Collections. In *TREC-9*.

Xu, J. and Croft, W. (1996). Query expansion using local and global document analysis. In *SIGIR-96*.

Data-Intensive Question Answering

Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais and Andrew Ng
Microsoft Research
One Microsoft Way
Redmond, WA 98052
{brill, mbanko, sdumais}@microsoft.com
jlin@ai.mit.edu; ang@cs.berkeley.edu

1 Introduction

Microsoft Research Redmond participated for the first time in TREC this year, focusing on the question answering track. There is a separate report in this volume on the Microsoft Research Cambridge submissions for the filtering and Web tracks (Robertson et al., 2002). We have been exploring data-driven techniques for Web question answering, and modified our system somewhat for participation in TREC QA. We submitted two runs for the main QA track (AskMSR and AskMSR2).

Data-driven methods have proven to be powerful techniques for natural language processing. It is still unclear to what extent this success can be attributed to specific techniques, versus simply the data itself. For example, Banko and Brill (2001) demonstrated that for confusion set disambiguation, a prototypical disambiguation-in-string-context problem, the amount of data used far dominates the learning method employed in improving labeling accuracy. The more training data that is used, the greater the chance that a new sample being processed can be trivially related to samples appearing in the training data, thereby lessening the need for any complex reasoning that may be beneficial in cases of sparse training data.

The idea of allowing the data, instead of the methods, do most of the work is what motivated our particular approach to the TREC Question Answering task. One of the biggest challenges in TREC-style QA is overcoming the surface string mismatch between the question formulation and the string containing its answer. For some Question/Answer pairs, deep reasoning is needed to relate the two. The larger the data set from which we can draw answers, the greater the chance we can find an answer that holds a simple, easily discovered relationship to the query string.

Our approach to question answering is to take advantage of the vast amount of text data that is now available online. In contrast to many question answering systems that begin with rich linguistic resources (e.g., parsers, dictionaries, WordNet), we begin with data and use that to drive the design of our system. To do this, we first use simple techniques to look for answers to questions on the Web. Since the Web has orders of magnitude more data than the TREC QA document collection, simple techniques are likely to work here. After we have found suitable answer strings from online text, we project them onto the TREC corpus in search of supporting documents.

2 Answer Redundancy and Question Answering

Answer redundancy (multiple, differently phrased, answer occurrences) serves two purposes for our task of question answering. First, the occurrence of multiple linguistic formulations of the same answer increases the chances of being able to find an answer that occurs within the context of a simple pattern matching the query. For instance, it is not difficult to match the question "Who killed Abraham Lincoln" with the text "John Wilkes Booth killed Abraham Lincoln," but it is more challenging to find the answer to this question in the text "John Wilkes Booth is perhaps America's most infamous assassin. He is best known for having fired the bullet that ended Abraham Lincoln's life."

The TREC corpus has considerably less answer redundancy than the Web – the TREC QA database consists of fewer than 1 million documents, whereas Web search engines are now indexing more than 2 billion pages. By analyzing the set of documents returned by the union of all groups, we see that only 37 of the TREC 2001 queries have 25 or more documents with a correct answer, and only 138 have 10 or more documents. Given a source, such as the TREC corpus, that contains only a relatively small number of formulations of answers to a query, we may be faced with the difficult task of mapping questions to answers by way of uncovering complex lexical, syntactic, or semantic relationships between question string and answer string. The need for anaphor resolution and synonymy, the presence of alternate syntactic formulations and indirect answers all make answer finding a potentially challenging task. However, the greater the answer redundancy in the source, the more likely it is that we can find an answer that occurs in a simple relation to the question, and therefore, the less likely it is that we will need to resort to solving the aforementioned difficulties facing natural language processing systems.

The second use of answer redundancy is to facilitate answer extraction. Even if we find a simple relationship between the question and the proposed answer, the answer might be incorrect. It is possible that the source made a mistake, or that the seemingly correct answer string appears in a context that identifies it as possibly incorrect (e.g. "John thinks that Andrew Jackson killed Abraham Lincoln"). Additionally, even with a highly redundant information source, there will be questions for which no simple-relationship answer can be found. To lessen these challenges, we can use answer redundancy to combine a number of uncertain guesses into a single, much more reliable guess. This is the kind of redundancy explored in Abney et al. (2000), Clarke et al. (2001) and Kwok et al. (2001).

3 System Overview

Our system utilizes a search engine[1] to find answers on the Web, an approach similar to that described in Kwok et al. (2001). Given a question, we formulate multiple queries to send to the search engine, we ask for the 100 best matching pages for each, and then harvest the returned summaries for further processing. A set of potential answers is extracted from the summary text, with each potential answer string weighted by a number of factors, including how well it matches the expected answer type and how often it occurred in the retrieved page summaries.

Given a set of possible answers, we then perform *answer projection,* searching for supporting documents in the TREC QA document collection. The system returns the four best <answer, document ID> pairs. We made no attempt to determine when an answer did not exist in the TREC corpus; instead we always returned "NIL" in the fifth position. A flow diagram of our system is shown in Figure 1. Below we discuss each component in detail.

3.1 Query Reformulation

Given a query Q, we would like to search our document collection for possible answer strings S. To give a simple example, from the question "When was Abraham Lincoln born?" we know that a likely answer formulation takes the form "Abraham Lincoln was born on <DATE>". Therefore, we can look through the data, searching for such a pattern. While it may be possible to learn query-to-answer reformulations (e.g., Agichtein et al., 2001; Radev et al., 2001), we created these manually. We did not use a parser or part-of-speech tagger for query reformulation, but did use a lexicon in order to determine the possible parts-of-speech of a word as well as its morphological variants.

We first classify the question into one of seven categories, each of which is mapped to a particular set of rewrite rules. Rewrite rule sets ranged in size from one to five rewrite types. The output of the rewrite module is a set of 3 tuples of the form [string, L/R/ , weight], where "string" is the reformulated search query, "L/R/-" indicates the position in the text where we expect to find the answer with respect to the query string (to the left, right or anywhere) and

[1] For the experiments reported here, we used Google as the backend Web search engine.

"weight" reflects how much we prefer answers found with this particular query. The idea behind using a weight is that answers found using a high precision query (e.g. "Abraham Lincoln was born on") are more likely to be correct than those found using a lower precision query (e.g. "Abraham" "Lincoln" "born").

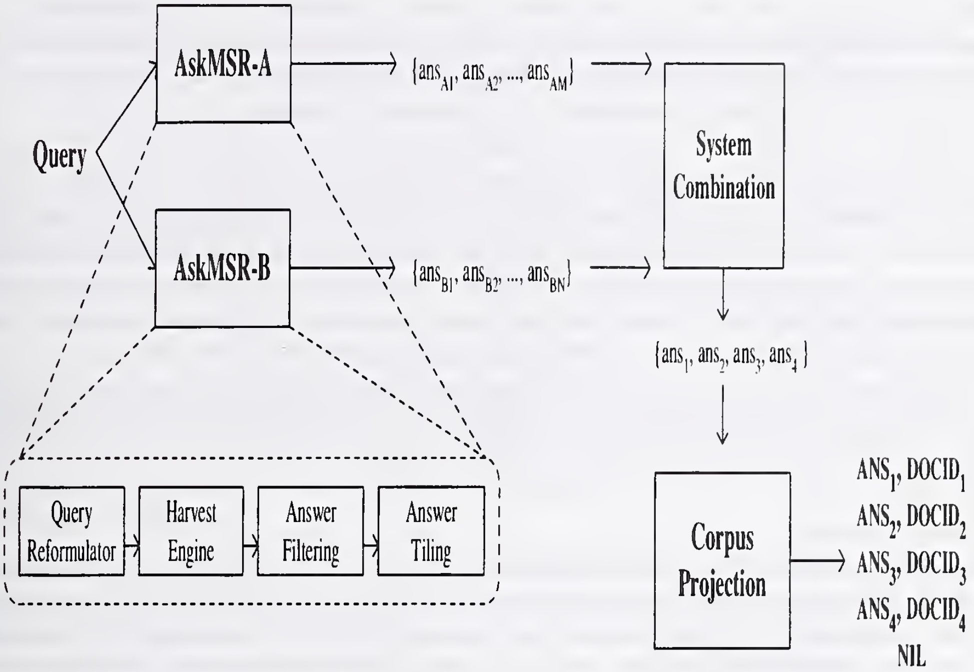


Figure 1. AskMSR System Architecture

The rewrites generated by our system were simple string-based manipulations. For instance, some question types involve query rewrites with possible verb movement; the verb "*is*" in the question "Who **is** the world's richest man married to?" should be moved in formulating the desired rewrite to "The world's richest man **is** married to". While we might be able to determine where to move a verb by analyzing the sentence syntactically, we took a much simpler approach. Given a query such as "Who is w_1 w_2 ... w_n", where each of the w_i is a word, we generate a rewrite for each possible position the verb could be moved to (e.g. "w_1 is w_2 ... w_n", "w_1 w_2 is ... w_n", etc). While such an approach results in many nonsensical rewrites (e.g. "the world's is richest man married to"), these very rarely result in the retrieval of bad pages, and the proper movement position is guaranteed to be found via exhaustive search. If we instead relied on a parser, we would require fewer query rewrites, but a misparse would result in the proper rewrite not being found. We currently use only simple string matching, but could enhance our rewrites to include richer patterns as Soubbotin and Soubbotin (2002) have done.

The rewrites for the query "*What is relative humidity?*" are:

["+is relative humidity", LEFT, 5]
["relative +is humidity", RIGHT, 5]
["relative humidity +is", RIGHT, 5]
["relative humidity", NULL, 2]
["relative" AND "humidity", NULL, 1]

3.2 N-Gram Harvesting

Once we have obtained the set of rewrites, we submit each reformulated query to the search engine. For efficiency reasons, we worked only with the summaries returned for each hit rather than retrieving the full-text of pages (as was done by Kwok et al. (2001) and Clarke et al. (2002)).

The returned summaries contain the query terms, usually with a few words of surrounding context. The summary text is then processed to retrieve only strings to the left or right of the query string, as specified in the rewrite triple. In some cases, this surrounding context has truncated the answer string, which may negatively impact our results.

We obtain 1-gram, 2-grams and 3-grams from the short summaries. We score each n-gram according to the weight of the query that retrieved that it and sum these weights across all summaries containing the n-gram. So, the weight for each candidate n-gram is given by:

$$ngram\_weight = \sum_{ngram \in summaries} rewrite\_weight$$

There are a couple of important things to note about this weighting scheme. First, we count an n-gram only once within each summary, so there is no *tf* component. Second, the more summaries an n-gram occurs in the higher weight it gets, which is the opposite of the usual *idf* approaches to term weighting. Shorter n-grams will occur more often, but we use tiling to increase the counts for longer n-grams, as described below. Because we do not use any global term weights, we do not need to index the documents directly nor maintain a local database of term weights.

When searching for candidate answers, we enforce the constraint that stop words are not permitted to appear in any potential n-gram answers. In retrospect, this was too stringent a requirement.

3.3 Answer Typing

Next, we use type filters to increment/decrement each n-gram count based on expected type (gleaned from the question) and a guess as to the type of the n-gram. The system uses filtering in the following manner. First, the query is analyzed and assigned one of seven question types, such as *who-question, what-question,* or *how-many-question.* Based on the query type that has been assigned, the system determines what collection of filters to apply to the set of potential answers found during n-gram harvesting. The answers are analyzed for features relevant to the filters, and then rescored according to the presence of such information.

A collection of approximately 15 filters were developed based on human knowledge about question types and the domain from which their answers can be drawn. Most filters used surface string features, such as capitalization or the presence of digits, and consisted of hand-crafted regular expression patterns. Some filters were driven by more sophisticated properties such as semantic features or part-of-speech assignments, and used natural language analysis (Jensen et al., 1993) capable of associating such characteristics with strings. For example, these filters indicate that the strings "Pope Julius", "Julius II", and "David" refer to people, whereas "Vatican" refers to a location, which will be helpful for correctly answering *who-* or *where-questions*.

The selected filters are applied to each candidate string and used to adjust the initial score of the string. In most cases, filters are used to boost the score of a potential answer when it has been determined to possess the features relevant to the query type. In other cases, filters are used to remove strings from the candidate list altogether. This type of exclusion was only performed when the set of correct answers was determined to be a closed set (e.g. "Which continent....?") or definable by a set of closed properties (e.g. "How many...?").

The filters were determined to yield 26.4% relative improvement in MRR on a held-out subset of TREC9 queries, compared to using no type filter re-weighting.

3.4 Answer Tiling

Finally, we applied an answer tiling algorithm, which both merges similar answers and assembles longer answers out of answer fragments. Tiling constructs longer n-grams from sequences of overlapping shorter n-grams. For example, "A B C" and "B C D" is tiled into "A B C D." The weight of the new n-gram is the maximum of the constituent n-gram weights. The algorithm proceeds greedily from the top-scoring candidate - all subsequent candidates (up to a certain cutoff) are checked to see if they can be tiled with the current candidate answer. If so, the higher scoring candidate is replaced with the longer tiled n-gram, and the lower scoring candidate is removed. The algorithm stops only when no n-grams can be further tiled.

4 System Combination

We had developed two semi-independent versions of the system, differing in the set of rewrite rules, tiling algorithm and type filters. It has been demonstrated in many settings that, given several algorithms for a prediction problem, combining their results via a voting scheme can frequently result in performance better than that of any of the individual algorithms, since different systems can reinforce each other's strengths and also help correct each other's wrong answers. Towards realizing such gains for our system, we learned an automatic method for combining the results from the two systems (AskMSR-A and AskMSR-B).

The first step in combining the answers was to run through AskMSR-A and AskMSR-B's lists of outputs to determine when two of their answers can be deemed equivalent and hence should be merged. This is a step that can frequently be omitted by most voting schemes that have to deal with only a small set of possible output values (rather than the large set of all possible strings in our setting), and we note that determining whether two answers match is a more subtle task than might appear at first blush. For instance, exact string matching will fail to recognize that ``redwood trees" and ``redwoods'' are almost certainly the same thing; simple substring matching also fails on this example.

In our system, we tested whether two answers A and B matched by checking if either every stem of every word in A matches a stem of some word in B, or vice versa. Armed with this test, we then merge AskMSR-A and AskMSR-B's lists of answers into a single, combined list, as follows: We initialize the ``combined list" to the empty list, and then repeatedly test pairs of answers (one from AskMSR-A, one from AskMSR-B) to see if they match; upon finding a match, both answers are deleted from their respective lists, and the (lexicographically) longer answer is added to the combined list. When no more pairs of matches are to be found, we then add the answers still remaining in the AskMSR-A and the AskMSR-B lists to the combined list.

Having formed a combined list of answers, we then learn a way for ranking them. For almost any learning method, the choice of features critically impacts performance. In our specific application, we desire features that can be used to characterize how ``confident" we are about each answer's correctness, so that we can rank the answers we are confident about higher. While AskMSR-A and AskMSR-B both output their own confidence scores, the absolute values of these scores were only very weakly predictive of confidence. On the other hand, the rankings of the answers output by each of the two methods were far more predictive of confidence; this can also be thought of as using the relative, rather than absolute, values of these scores.

We therefore learned a function that took as input the rankings of an answer output by either or both algorithms, and whose task it was then to output a ``score" determining how confident we are that this answer is correct. Here, these ``scores" have no intrinsic meaning (such as the probability of being correct), and our goal is only that when the results are sorted according to the scores, that the resulting expected MRR be high.

Using TREC-9 QA queries 201-400 as our training data, the parameters of our function approximator were automatically tuned to maximize the empirical MRR on the training set. On holdout test data, we estimated that this method improved our overall MRR by 11% over the better of AskMSR-A and AskMSR-B.

5 Answer Projection

At this point the system has produced a list of the n-best answers for a question. These answers were determined using web data. The next task was to find supporting documents in the TREC document collection for each answer candidate. In the projection phrase, five possible supporting documents are found for each of the five candidate answers. The Okapi IR system was used for finding the supporting documents for each candidate answer (Robertson et al., 1995). The query submitted to Okapi was just the list of query words along with the candidate answer. Documents were ranked using the standard best match ranking function, bm25. We did not use any phrase or proximity operators to increase precision nor any pseudo relevance feedback to increase coverage. We did not use Boolean operators to ensure that the candidate answer be matched.

To generate the final answers, the first supporting document for each candidate answer was chosen, unless there existed a supporting document for the candidate answer that was also retrieved as the supporting document for another candidate answer, in which case the duplicate supporting document is returned. For example, if a candidate answer had supporting documents d1, d2, etc., d2 is returned if another candidate answer is supported by d2. The reasoning behind this strategy is that candidate answers tend to be related to the correct answer, and multiple occurrences of a document suggested that the document contain either the answer or terms related to the answer. In practice, however, this mechanism was rarely used - almost all supporting documents returned were the first one.

Although we designed the answer projection component to work for the TREC QA track, we believe it is more generally applicable. For example, if one had a small reliable source like an encyclopedia, newspaper, or help documentation, one could use the same idea – first find possible answers using our simple system in a large noisy collection like the Web and then project the answers to the reliable sources for verification.

6 Results and Analysis

We present the official TREC 2001 results for our two submitted runs, AskMSR and AskMSR2, in the table below. We used exactly the system described above for the AskMSR run. For AskMSR2, we used a somewhat different projection algorithm than described above, which improved performance on our TREC9 hold-out set but had little impact on the actual test data, as shown in the table. The average answer length was 14.6 bytes for both systems, well below the 50 byte limit. Since we had no training data to calibrate the system scores, we did nothing to handle NIL queries, and simply placed a NIL response in position 5 for every query.

System	*Strict*	*Lenient*
AskMSR		
MRR	0.347	0.434
% no answers	49.2	40.0
AskMSR2		
MRR	0.347	0.437
% no answers	49.6	39.6

Table 1. TREC 2001 results

We were quite pleased with the results of our very simple system in our first participation in the TREC QA track. Although we had been working on Web QA for a few months, our entire TREC QA endeavor was done, from scratch, in two weeks. There is still a great deal that can be done to improve the system. One of the biggest weaknesses of our system was the simple strategy we used to map an answer onto a supporting document, as seen in our .09 drop in MRR from finding an answer to finding a supporting document for that answer. Clarke et al. (2002) and Buchholz (2002) also report lower TREC performance compared to Web performance. A number of projection errors came from the temporal differences in the Web and TREC collections. E.g., for query *1202: Who is the governor of Alaska?,* we return Tony Knowles, who is the governor in 2001, but not Steve Cowper who was the governor in 1989.

There were several other bugs and sub-optimal design decisions in our initial TREC QA system. One problem was our decision not to include stop words in the n-gram strings (e.g., For query *1358: In which state would you find the Catskill Mountains?,* our top answer was 'Regional York

State', but we omitted 'New' because it was a stop word. We have removed this constraint in our current system and it improves performance considerably. Other problems occurred in answer tiling (e.g., For query *1288: What is nepotism?,* our top answers were: 'favoritism shown'; 'relatives'; and 'employment' which we did not tile correctly because 'to' and 'in' were linking stop words that we removed. Other areas for improvement are: handling of quantities, answer typing, and question reformulation, which are useful more broadly than the TREC question-answering task.

There are several examples where our simple approach does quite well compared to other systems. Typically these are cases where simple rewrites work well with the large Web collection but much more complex processing is required to find answers within the small TREC document collection. Consider the following query-document pairs. (These are the only relevant documents for these queries within the TREC collection.)
1083: What is the birthstone for June? <answer: pearl>
<DOC> ... *For anyone fascinated by pearls who wants to learn more about them, a tiny but magical London jewellery shop, Manguette, is having a festival of pearls (faux and real) for two weeks during June (the pearl is the birth-stone for those born in that month)...* Only three groups find this document. There are two difficulties in finding this document in the TREC collection -- pronominal reference must be used to know that 'that month' refers to June, and the query term birthstone needs to be rewritten as birth-stone which occurs in the document. With the wealth of data available on the Web, we can find the answer without solving either of these problems.
1340: What is the rainiest place on Earth? <answer: Mount Waialeale>
<DOC> ... *In misty Seattle, Wash., last year, 32 inches of rain fell. Hong Kong gets about 80 inches a year, and even Pago Pago, noted for its prodigious showers, gets only about 196 inches annually. ... (The titleholder, according to the National Geographic Society, is Mount Waialeale in Hawaii, where about 460 inches of rain falls each year.)...* Again, only three groups find this document. This is a much more interesting case. Some fairly sophisticated processing needs to be done to know that titleholder means rainiest.

After submitting our TREC run, we continued to improve the system for general web QA capabilities. After receiving the TREC relevance judgments, we tried the new system on the TREC queries, and were pleased to see some sizable improvements. We analyzed our new system on the 30 "worst" questions for our system – that is, the questions with the greatest difference between mean score across groups and our score for a question. On these 30 questions, our official submission attained an MRR of 0. The improved system attained an MRR of 0.390 on these 30 questions. There would be improvements on other queries as well although we have not scored the full set by hand.

We believe that data redundancy is a readily available and valuable resource that should be exploited for question answering in much the same way as linguistic resources often are. The performance of our system shows promise for approaches to question answering which make use of very large text databases, even with minimal natural language processing.

References

S. Abney, M. Collins and A. Singhal (2000). Answer extraction. In *Proceedings of ANLP 2000.*

E. Agichtein, S. Lawrence and L. Gravano (2001). Learning search engine specific query transformations for question answering. In *Proceedings of WWW10.*

M. Banko and E. Brill; Scaling to very very large corpora for Natural Language Disambiguation. In *Proceedings of ACL*, 2001.

S. Buchholz (2002). Using grammatical relations, answer frequencies and the World Wide Web for TREC question answering. To appear in *Proceedings of the Tenth Text REtrieval Conference (TREC 2001).*

C. Clarke, G. Cormack and T. Lyman (2001). Exploiting redundancy in question answering. In *Proceedings of SIGIR'2001.*

C. Clarke, G. Cormack and T. Lynam (2002). Web reinforced question answering. To appear in *Proceedings of the Tenth Text REtrieval Conference (TREC 2001).*

C. Kwok, O. Etzioni and D. Weld (2001). Scaling question answering to the Web. In *Proceedings of WWW'10.*

Jensen, K., Heidorn, G.E., and Richardson, S.D. eds. 1993. *Natural Language Processing: The PLNLP Approach.* Kluwer Academic Publishers.

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 *ACM CIKM 2001: Tenth International Conference on Information and Knowledge Management.*

S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu and M. Gatford (1995). Okapi at TREC-3. In *Proceedings of the Third Text REtrieval Conference (TREC-3).*

S. E. Robertson, S. Walker and H. Zaragoza (2002). Microsoft Cambridge at TREC-10: Filtering and web tracks. To appear in *Proceedings of the Tenth Text REtrieval Conference (TREC 2001).*

M. M. Soubbotin and S. M. Soubbotin (2002). Patterns and potential answer expressions as clues to the right answers. To appear in *Proceedings of the Tenth Text REtrieval Conference (TREC 2001).*

E. Voorhees and D. Harman, Eds. (2001). Proceedings of the Ninth Text REtrieval Conference (TREC-9).

E. Voorhees and D. Harman, Eds. (2002). Proceedings of the Tenth Text REtrieval Conference (TREC 2001).

Qanda and the Catalyst Architecture

Pranav Anand David Anderson John Burger[*]
John Griffith Marc Light Scott Mardis Alex Morgan
The MITRE Corporation
Bedford, MA 01730 USA

Introduction to Qanda and Catalyst

Qanda is MITRE's entry into the question-answering (QA) track of the TREC conference(Voorhees & Harman 2002). This year, Qanda was re-engineered to use a new architecture for human language technology called *Catalyst*, developed at MITRE for the DARPA TIDES program.

The Catalyst architecture was chosen because it was specifically designed for fast processing and for combining the strengths of Information Retrieval (IR) and Natural Language Processing (NLP) into a single framework. These technology fields are critical to the development of QA systems.

The current Qanda implementation serves as a prototype for developing QA systems in the Catalyst architecture. This paper serves as an introduction to Catalyst and the Qanda implementation.

What is Catalyst?

Catalyst is a framework for creating and experimenting with Human Language Technology (HLT) systems. It attempts to address several problems typical of current approaches to component-based HLT systems. The principal problems that Catalyst is designed to ameliorate are these:

- Systems do not scale easily to handle today's information processing needs. Systems are needed to process human language very quickly or in very large amounts.
- Experimenting with a variety of potential system configurations is difficult because each pair-wise component interaction typically requires specialized integration code for smooth operation.

The approach that we are using in the Catalyst framework to address these problems is to combine *standoff annotation* and *dataflow*.

Standoff Annotation

The Catalyst data model, like those of both the TIPSTER and GATE architectures(Cunningham, Wilks, & Gaizauskas 1996; Grishman 1996), is annotation based. A signal (text, audio, etc.) is augmented with annotations that mark up portions of the signal with supplemental or derived information.

Catalyst annotations are standoff (versus inline) which means that the underlying signal is unmodified and annotations are maintained and communicated separate from the signal. By separating the signal from the annotations and annotations of different types from each other, Catalyst is able to automatically construct customized streams of annotation for each component in a system. The set of annotations, attributes and their names can all be transparently modified between each language processing component without modifying any component code or inserting additional scripts.

Every standoff annotation has an annotation type identifier, a start position, an end position, and zero or more attributes. The attributes are named fields that provide information derived from or associated with the annotated text. For example, a tokenizer might emit **word** annotations, with **text**, **stem** and **part-of-speech** attributes. The start and end of each such annotation would indicate where in the text the tokenizer found the words.

Dataflow Processing

In order to support distributed, scalable systems, Catalyst is based on a dataflow model of language processing components. We refer to these components as language processors (LPs). The dataflow model allows us to describe an HLT system directly in terms of the data dependencies between LPs. Furthermore, we are able to use the natural ordering properties of the standoff annotation indices to synchronize the operation of the various components.

Each LP in a Catalyst system is connected to others by annotation streams consisting of a flow of standoff annotations serialized according to the following predicate.

$$A_1 < A_2 \text{ if } \begin{cases} A_1.\text{start} < A_2.\text{start} \\ \vee ((A_1.\text{start} = A_2.\text{start}) \wedge (A_1.\text{end} > A_2.\text{end})) \\ \vee ((A_1.\text{start} = A_2.\text{start}) \wedge (A_1.\text{end} = A_2.\text{end}) \\ \quad \wedge (A_1.\text{annotation-type} < A_2.\text{annotation-type})) \end{cases}$$

As a node in a dataflow network, each LP has a declaration that defines its annotation input requirements and annotation outputs. A system declaration identifies the required language processors and the desired annotation stream connections between them (connections that satisfy each LP's input requirements). From these declarations Catalyst can arrange to deliver to each component only the annotations that are expected. Thus, components do not need to forward

[*] john@mitre.org

annotations unrelated to their specific function. For example, a sentence tagger may consume the bf word annotations produced by the tokenizer described above, and emit annotations indicating the boundaries of sentences. The sentence tagger need not copy the words to its output—if a third component requires both sentences and words, Catalyst will arrange to deliver the outputs of the tokenizer and the sentence tagger, suitably merged.

Catalyst's dataflow approach to building HLT system has a number of advantages.

- Error dependencies between components are limited to the precisely specified data dependencies.
- By using dedicated peer-to-peer channels Catalyst eliminates the cost of parsing and generating generic annotation interchange formats (such as XML) between independently constructed components.
- Component developers may work directly with an annotation model, rather that with particular data interchange formats. (Catalyst will support the exchange of data in XML for system I/O and for use with components not prepared for direct use in a Catalyst system.)
- A system can be run on a single machine or distributed across many. Individual components can be replicated to increase throughput.
- Component code can be simplified because the data presented is always consistent with the LP specification.

Distributability

Catalyst annotation streams are transported over sockets and can be connected between processes on many machines, permitting a wide range of processing strategies for optimizing system performance without having to rewrite component code. Once properly working on a single host, distributing a system across many machines requires only starting a server on each machine and editing a few lines in the system configuration file.

Control

A network of Catalyst servers exchange information for the purpose of creating and maintaining Catalyst-based systems. Connections are negotiated by servers and then handed-off to component processes. A single script, compiled from a static dataflow description of the system, works in concert with the servers to create each system. Servers also route and deliver control commands to each language processor.

Logging and Monitoring

A multi-process, distributed system can be difficult to debug and maintain. To assist component and system developers in this regard, Catalyst has both distributed logging and distributed monitoring capabilities.

The Catalyst log capability allows logger processes to collect information from some or all of the processes in a Catalyst system. Logging information includes events such as when language processors are stopped or started, user log messages, command events, errors, etc. Logs may be created at the same time the system is instantiated or may be added later as needed. Multiple loggers can be created simultaneously to record several views of the log at different levels of detail and can be used to create logs at multiple destinations.

The Catalyst monitor is used to examine the configuration and state of a Catalyst system. Using the monitor, a component or system developer can obtain snapshots of the current system configuration and track the flow of data through a system. The monitor provides information such as the list of current running components, the connecting annotation streams, the amount of information buffered with the system, the current indices for the various streams, etc. Debugging multicomponent systems such as Qanda requires a facility to examine the global system state easily. The monitor is an important tool for quickly diagnosing component interaction problems and identifying performance bottlenecks.

Information Retrieval in Catalyst

In addition to addressing some of the general problems of HLT system construction, Catalyst is also an experiment in developing a framework for combining NLP and IR in a single system. Standoff token annotations, grouped by term, form the basis of an inverted index for terms in a large corpus, similar to those used by traditional IR engines. By extending this usage to all other types of annotation, Catalyst permits the development of fast information retrieval techniques that query over NLP-generated products (see examples below).

Catalyst's dataflow model, combined with flexible inverted index streams, makes it possible to develop systems that can utilize both pre- and post-index NLP to improve the speed of query responses. Also, retrieval engines can be built that directly answer complex queries as needed for question answering (e.g., retrieve all paragraphs containing a person and one of terms A or B).

Implementing Qanda using Catalyst

Our previous TREC efforts have used inline-XML pipeline architectures, where all components monotonically added XML markup to retrieved documents. This approach had a number of problems:

- Components downstream had to understand (or at least parse) all upstream annotations, in order to ensure that these earlier annotations were properly replicated on output.
- This led to an inflation of the markup on documents: Often the final documents comprised 99% markup and 1% underlying character data.
- Some components' only purpose was to rewrite markup to make it more palatable to downstream components.
- It is difficult to parallelize such an architecture.

Our current Catalyst-based architecture suffers from none of these problems. Every component is delivered only the annotations that it requires to do its job. If a component is producing annotations that no other component currently

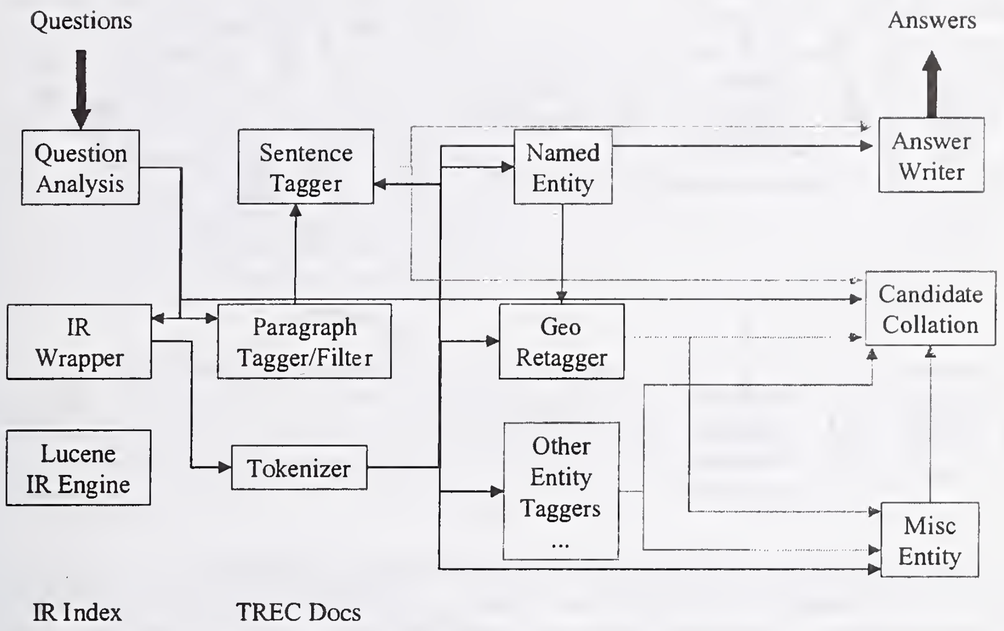


Figure 1: Qanda as a Catalyst System—Wide arcs indicate file system IO, narrow arcs are lightweight Catalyst annotation streams.

requires, Catalyst quietly drops them on the floor.[1] If necessary, we can instruct Catalyst to map between different annotation types in order to accommodate differences in the natural representations of different components.

Catalyst also allows us to lay the system out in a more natural manner than a single pipeline. Figure 1 shows our TREC system, which is naturally expressed as a directed graph. Note that many components do not need to communicate with each other, even indirectly, and can thus run in parallel, e.g., most of the entity taggers. Although we have not yet taken advantage of it, Catalyst will allow us to run language processors on different machines, even replicating slower components in order to increase throughput.

Using Catalyst

Working with Catalyst entails developing Catalyst-enabled components and assembling them into a system. Catalyst is designed to simplify this second task. First we describe the general way in which Catalyst systems are constructed; next we show two possible paths for integrating existing technology with Catalyst.

Building a System in Catalyst

Figure 2 shows how the major components of Catalyst (the library, the server and the configuration compilers) are used in creating a running system. A configuration file is written for each language processing component (LP). It specifies which annotation streams it is able to process and which it generates. A component compiler transforms the configuration file into header files and other static information that are used to create each Catalyst-enabled executable. A system configuration file, referring to LP configuration files, defines which LPs are needed in a system and the stream connections that are required between the LPs. The system compiler transforms the system configuration file into a startup script that is used to instantiate the system. The script (presently a PERL 5.0 script) communicates only with Catalyst servers, whose function is to create the operating system processes that will contain the LPs, establish connections between them, and forward configuration and control information from the script to each process.

The Catalyst library (linked into each component process) handles annotation communication between components and passes control information to and from the servers. The library can merge annotations from many different components and produce a single annotation stream specialized

[1] Of course, the preferred behavior would be for the component to neglect computing such annotations in the first place, but at least they are not further processed.

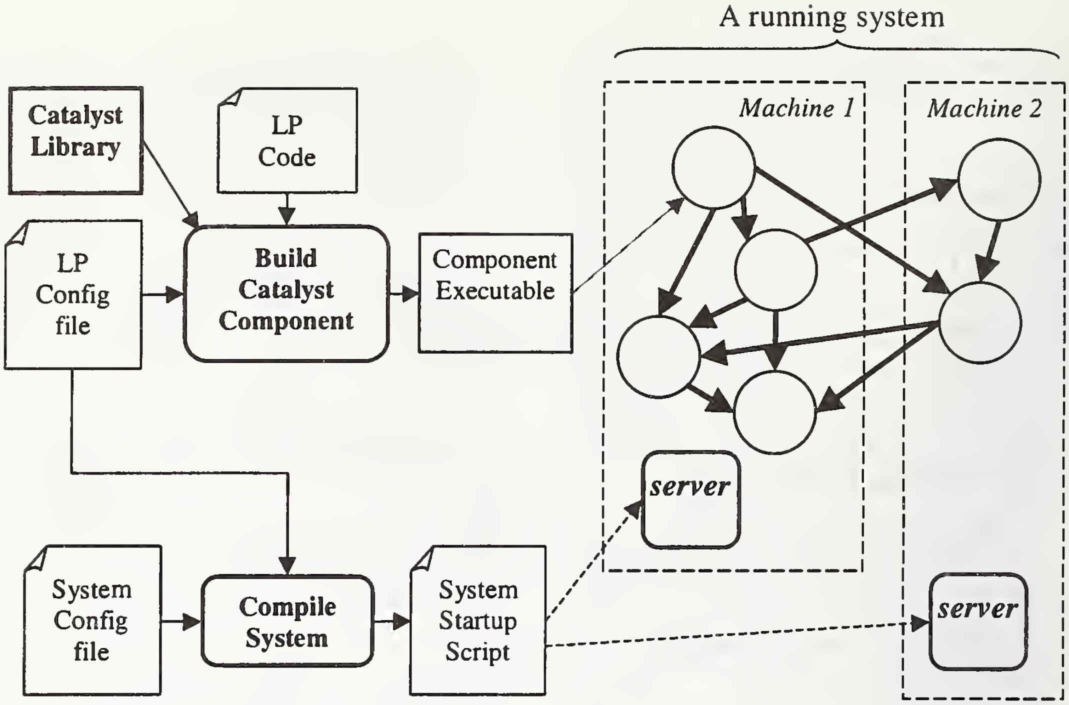


Figure 2: How Catalyst Builds a System—Configuration compilers transform configuration files into header, data, and script files that are used to create language processors[LPs] and systems. A compiled start-up script uses the network of Catalyst servers to create a running system. The system, once started, passes annotations via peer-to-peer communication managed by the Catalyst library.

to each component's input declaration. Similarly, a single stream of output from a component can be broken down into its constituent annotations and attributes and delivered piece-meal to many destinations. In this way, Catalyst delivers to each component, the precise set of annotations required by the component's declaration.

Integrating with Catalyst

There are two basic ways of integrating existing components with Catalyst: writing a Catalyst wrapper process and using the Catalyst API. The purpose of a Catalyst wrapper process is to convert the Catalyst annotations streams to and from a data format that an existing black box component uses. The Catalyst API, of course, provides direct access to all of Catalyst's features and provides maximum benefit.

A wrapper process allows one to connect existing technology into Catalyst without having to modify the component code. This is the only choice for components for which the code is unavailable. It would provide the advantage of delivering a precise set of annotations to the wrapped component but it would suffer from the cost of transformation to and from the appropriate interchange format. Also, transforming annotations from standoff to inline formats and back again (as most component technologies would require) can be difficult. The Catalyst project is, however, planning direct support for inline annotations in XML to facilitate integration via wrapper processes.

The Catalyst API defines a standoff annotation model and provides methods for sharing data via annotation streams. Standoff annotations allow for overlap in ways that cannot be constructed in an inline format such as XML, permitting, for example, components to output many possibly overlapping noun phrase bracketings or answer candidates. Additionally, a component can receive an annotation stream that contains the combined outputs of several components that all share the same task (e.g., it is simple to develop a component that looks at N different tagger outputs and selects the best tags by combining the results).

Future Directions for Qanda within Catalyst

Currently, work is proceeding within the Catalyst project on two important technologies: Persistent annotation archives and annotation indexes. The goal of archiving annotations is to store and then later reuse a stream of annotations. For instance, the tokenization and entity tagging in Qanda could be done on the entire TREC corpus ahead of time, then pulled from an archive at question-answering time. Consumer language processors will not be aware that their input annotations are being read from disk rather than being created by a "live" producer.

The goal of annotation indexing is to invert arbitrary text "containers", not just documents or paragraphs. Thus, one might query for archived **Location** entities containing the term *Berlin*, to answer a question such as *When did the Berlin wall come down?*. In addition, we want to index all annotations not just on the terms they contain, but on all of their other contained annotations as well. This is similar to the work of (Prager *et al.* 2000; Kim *et al.* 2001) but intended to be more general and comprehensive. With both of these capabilities in place, we imagine that complex queries might be formulated, such as:

> Retrieve **Sentence**s containing **Date**s and also containing the term *wall* and also containing **Location** annotations containing the term *Berlin*.

We believe that such targeted queries will allow for very fast and accurate question answering systems. Optimizing such queries is admittedly complex, however, as is determining appropriate scoring mechanisms. Deciding how best to use archived coreference information is also an issue. Nonetheless, we believe that Catalyst provides a valuable framework for such sophisticated language processing systems.

References

Cunningham, H.; Wilks, Y.; and Gaizauskas, R. 1996. GATE – a general architecture for text engineering. In *Proceedings of the 16th Conference on Computational Linguistics (COLING96)*. http://citeseer.nj.nec.com/43097.html.

Grishman, R. 1996. Tipster architecture design document version 2.2. Technical report, DARPA TIPSTER.

Kim, H.; Kim, K.; Lee, G. G.; and Seo, J. 2001. MAYA: A fast question-answering system based on a predictive answer indexer. In *Proceedings of the Workshop on Open-Domain Question Answering*.

Prager, J.; Brown, E.; Coden, A.; and Radev, D. 2000. Question-answering by predictive annotation. In *Proceedings of ACM SIGIR*. Athens.

Voorhees, E., and Harman, D., eds. 2002. *Proceedings of the Tenth Text Retrieval Conference (TREC-10)*. http://trec.nist.gov/pubs.html.

Description of NTU System at TREC-10 QA Track

Chuan-Jie Lin and Hsin-Hsi Chen
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, TAIWAN, R.O.C.
E-mail: cjlin@nlg2.csie.ntu.edu.tw; hh_chen@csie.ntu.edu.tw
Fax: +886-2-23628167

1. Introduction

In the past years, we attended the 250-bytes group. Our main strategy was to measure the similarity score (or the informative score) of each candidate sentence to the question sentence. The similarity score was computed by sums of weights of co-occurred question keywords.

To meet the requirement of shorter answering texts proposed in this year, we adapt our system, and experiment on a new strategy that is focused on named entities only. The similarity score is now measured in terms of the distances to the question keywords in the same document. The MRR score is 0.145. Section 2 will deal with our work in the main task.

We also attended the list task and the context task this year. In the list task, the algorithm is almost the same as the one in the main task except that we have to avoid duplicate answers and find the new answers at the same time. Positions of the candidates in the answering texts should be considered.. We will talk about this in Section 3.

In the context task, how to keep the context, and what the answers of the previous questions can help are the main issues. In our strategy, the answers of the first question are kept when answering the subsequent questions, but the answers of the other ones (denoted by question *i*) are kept only if question *i* has a co-referential relationship to its previous one. Section 4 will describe this strategy in more detail.

2. Main Task

In the previous 250-bytes task, we measured the similarity of the question sentence and each sentence in the relevant documents, and reported the top 5 sentences with the highest scores and with the question focus words. In our experiment, the real answer sometimes lies in the sentence that is not so "similar" to the question. It becomes harder to extract text shorter than 50 bytes and containing the answer in this manner. Therefore, we experiment on another strategy, which is "candidate-focused" rather than "sentence-focused".

After reading a question, the system first decides its question type and keywords as usual. Now every named entity in the relevant documents becomes our answer candidate. For each candidate, we find out its distances to the question keywords in the same document, and sum up the reciprocals of these distances. One question keyword only contributes once, i.e., if a keyword occurs more than once, only the one nearest to the candidate contributes the score. Moreover, we assign higher weights to the keywords that are named entities. After scoring all the candidates, the highest top five are proposed, together with the texts surrounding the candidates within 50 bytes. The texts are extracted in such a way that the candidates can be placed in the middle.

In our experiment, we found that if there is a question keyword right preceding or following the candidate, it will dominate the score despite of the other question keywords. To solve this problem, we divide the distance by three, i.e., we consider three words as a unit to measure the distance. The scoring function is shown as follows:

$$score(x) = \sum_{t \in Q \cap D} \frac{1}{\left\lceil \min\left(\left|pos_D(t) - pos_D(x)\right|\right)/3 \right\rceil} \times weight(t) \tag{1}$$

where x is an answer candidate, Q is the question sentence, D is the document currently examined, t is a term occurring in both Q and D, and $pos_D(t)$ is one of the occurrence positions of t in D.

The algorithms of deciding question type and extracting named entities are the same as those in last year, which was proposed in Lin and Chen (2000). If we cannot tell which question type a question belongs to, or the question type is not concerned with a named entity, we consider every kind of entities as candidates. To extract different answers as more as possible, we ignore those answering texts whose named entity answers have appeared in the previous answering texts.

Two runs were submitted this year. When question keywords were prepared in the first run *qntuam1*, variants of ordinary words (inflections of verbs, plural forms of nouns, etc.) and named entities (adjective forms of country names, abbreviations of organization names, *etc.*) are added into the keyword bag. Stems of keywords are also added with a lower weight. Note that no matter how many variants or stems of a keyword are matched in a document, only one of them contributes the score. We select the one that can contribute the highest score.

In the second run *qntuam2*, the synonyms and explanations provided by WordNet (Fellbaum *Ed.*, 1998) are also added, with lower weight to reduce the noise.

Moreover, if there are m words in an explanation text, and n words occur in the document, the matching score of this explanation is defined as $\sqrt{n/m} \times weight(e)$, where $weight(e)$ is the weight of this explanation.

MRRs of these two runs are 0.145 and 0.101 under strict strategy, respectively.

3. List Task

List task is a new task beginning in this year. A question does not only ask for its information need but also a specified number of answers. Therefore, the system has to offer different answers to the specified number. An example is Question 1:

Question 1: Name 20 countries that produce coffee.

In this case, the system is asked to provide 20 names of different countries. Besides deciding which country produces coffee, the system also has to decide if the answer is duplicated, or if two answers are identical to each other.

The main algorithm to this task is almost the same as the main task. The only difference is that we extract the answering text in the manner that the candidates will be located at the beginning. By this way, if more than one answer appears in the same sentence, the previously proposed candidates will not appear again in the subsequent answering texts. The algorithm of the main task has already ignored the same answers (which is lexically identical), so we do not do other things to check answer identity.

Two runs were submitted as the same as those in the main task. Scores of the average accuracy are 0.18 and 0.14, respectively.

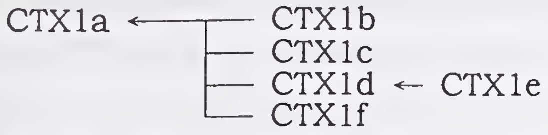
4. Context Task

There is another new task this year. A series of questions are submitted, which are somewhat relative to the previous questions. For example, in Question CTX1:

a. Which museum in Florence was damaged by a major bomb explosion in 1993?
b. On what day did this happen?
c. Which galleries were involved?
d. How many people were killed?
e. Where were these people located?
f. How much explosive was used?

Question CTX1a asks the name of the museum. Question CTX1b continues to ask the date of the event mentioned in Question CTX1a, so this question and its answer are important keys to Question CTX1b. Question CTX1c asks more details of

Question CTX1a, but irrelevant to Question CTX1b. So is Question CTX1d. But Question CTX1e refers to both Question CTX1a and CTX1d. We can draw a dependency graph of this series of questions as below:



If a question is dependent on one of its previous question, it is obvious that the information relative to this previous question is also important to the present question. Thus the system has to decide the question dependency.

We proposed a simple strategy to judge the dependency. Because the first question is the base question of this series, every subsequent question is dependent to the first one. After reading a question, if there is an anaphor or a definite noun phrase whose head noun also appears in the previous question, we postulate that this question is dependent on its previous question.

Next issue is that how we can use the dependency information in finding answers as well as its context information. After answering a single question, the system has located some answering candidates together with documents and segments of texts in which these candidates appear. Such information can be used to answer its subsequent dependent questions, as well as the keywords of the question itself. Note that context information can be transitive. In the above example, Question CTX1e consults the information that Question CTX1d itself owns, and Question CTX1d refers to, i.e., Question CTX1a.

In our experiment, we only consider the keywords and their weights as the context information. Furthermore, we assign lower weights to the keywords in the context information so that the importance of recent keywords cannot be underestimated. The answers to the previous question remain their weights because they are new information. The question type is decided by the present question.

The accompanying issue is that how confident an answer is included in the context information. This is because we may find the wrong answers in the preceding questions and those errors may be propagated to the subsequent questions. Moreover, do these five answers have the same weight? Or we trust the answers of the higher ranks than those of the lower ones, or only the top one is considered?

These issues are worthy of investigating, but not yet implemented in the experiment of this year. We assign weights to the previous answers according to the following equation:

$$weight(x) = weight\_NE(x) \times \sqrt{(6 - rank(x))/5} \times weight\_PreAns(x) \qquad (2)$$

where *weight_NE*(x) assigns higher weight if x is a named entity; *rank*(x) is the rank of x, and *weight_PreAns*(x) is a discount to the previous answers because they may be wrong. The square root part tries to assign higher weights to the higher-ranked answers.

Because only relevant documents to the first questions are provided, and we do not implement an IR system on TREC data, we cannot do a new search when answering the subsequent questions. Our solution is to search the same relevant set of the first question.

We submitted one run this year. Its main algorithm followed the first run of the main task.

There is still no formal evaluation of this task. The MRR of all 42 question of our result is 0.139. 4 of the first questions are correctly answered. Answers of at least one of the subsequent questions can also be found in each of these 4 series. Only one of the series is fully answered.

5. Discussion

Comparing the results of two runs of the main task and the two runs of the list task, we can find that synonyms and explanations introduce too much noise, so that the performance is worse. However, paraphrase is an important problem in question answering. Explanation provides only one of the paraphrases, thus we have to do more researches on paraphrases.

After investigation of the results of the list task, we found that there is a small bug when reporting answers. Although duplicate answers were neglected, equivalent answers were not. In other words, adjective forms of country names were regarded as different answers to their original names, which produced redundancy and lowered the performance.

In this year, the question types of many questions are not named entities. Many of them in the main task are "definition" questions. For example,

Question 896: Who was Galileo?
Question 897: What is an atom?

In our system, we only take named entities as answer candidates, so we cannot answer such type of questions, and the performance is rather worse than that of last year.

The same problem happened in the context task, too. Therefore, it is not obvious that our proposed model to the context task is good or bad. Further investigation and experiment are needed to verify this point.

References

Lin, C.J. and Chen, H.H., "Description of NTU System at TREC-9 QA Track," *Proceedings of The Ninth Text REtrieval Conference* (*TREC-9*), 2000, pp. 389-406.

Fellbaum, C. *Ed.* (1998) *WordNet: An Electronic Lexical Database.*

The NexTrieve Search System in TREC 2001

Gordon Clare and Kim Hendrikse

The NexTrieve Search System used in TREC

The NexTrieve search system is a combination fuzzy and exact search engine.

All document words are indexed. The exact (word) index comprises approximate document position information, along with "type" information indicating if the word is part of specially tagged text (such as a title or heading). At the time of the TREC runs, word presence (including type) at the document level was also recorded, but word frequency within a document was not.

The fuzzy index comprises text n-grams including "type" and originating word information, and their approximate document positions.

An "exact" search uses only the exact-word indexed information (namely word position and type information, and word-presence in document).

A "fuzzy" search uses the fuzzy-indexed information, and is assisted by a simultaneous exact word search. The "fuzziness" of a fuzzy search arises from the fact that not all query n-grams need be present in a hit for it to generate a good or winning score.

A score of a hit during searching was comprised of two parts -- a "document level" score and a "proximity" score.
The document level score is most important but simply collected word-presence-in-document information and, as such, does not vary on documents that contain the same set of search words with the same types.

The proximity level score of a document is the score given to the highest scoring region of the document containing the most search words (with most valuable types) in the smallest area. The position of this highest-scoring area is later used on winning documents to simplify preview generation.

Both levels of scoring had small multipliers in effect that increased as more search words were found in a particular document or region. Both levels also made use of the same scores applied to the originating words. These word level scores are generated from inverse frequency values in the database, augmented with word "type" information (giving an increase or decrease of the basic value). A "derived" penalty is also present, on words that have been automatically stemmed from an original query word.

Parameterization of TREC runs

A few technical details of the parameterization of the NexTrieve search engine for the TREC runs follows.

Four runs were submitted. Two were exact searches, and two were fuzzy.

- All runs were title-only, with stop words removed.
- All runs made use of a very simple stemming procedure (basically adding or removing a trailing 's' where necessary, and marking the modified word as "derived").
- All runs except ntvenx2 used a 45% increase in word score for words found in titles. Ntvenx2 used a 100% increase in word score, but this was only applied at the proximity level, not at the document level.

ntvenx1:
An exact search with a 45% increase in word score for words found in titles.

ntvenx2:
An exact search with a 100% increase in word score for words found in titles. This word score increase was only applied at the proximity level, not at the document level. Recalling that the document level score is the more important score, this has the effect of removing any "type" bias at the document level, but still preserving it at the proximity level where it is nominally more important.

ntvfnx3:
A fuzzy search with a setting of "minimal fuzzy". A 45% increase in word score for words found in titles was in effect. "Minimal fuzzy" has the effect of reducing the permitted word variation that can occur, and increasing the score degradation that is applied on the variation that does occur. Ie, same-letter trigrams from words who are more different from original query words get a correspondingly lower score.

ntvfnx4:
A fuzzy search with a setting of "maximal fuzzy". A 45% increase in word score for words found in titles was in effect. "Maximal fuzzy" has the effect of increasing the permitted word variation that can occur, and decreasing the variation-difference score degradation that is applied.

CONCLUSIONS

The NexTrieve TREC results were not as good as expected. The best-scoring run of the four runs submitted was ntvenx2, with an average precision of 0.13. After some analysis of the NexTrieve search system results for TREC 2001 several points became readily apparent.

- The lack of word frequency information at the document level and the lack of a document length component in the scoring significantly harmed the results. Simply by adding a suitable document length metric, and adding the word frequency information, the mean average precision was increased by around 50%, to a current best of 0.19 for an exact word search.

- The presence or absence of a title-text-scoring-improvement makes very little difference to the TREC scores of NexTrieve. This is possibly due to the fact that there is, in fact, not a lot of information in titles.

- Having local ("proximity") information take part in the scoring doesn' t seem to significantly change the TREC results either. Also, the "small multipliers" affecting scores as more words were present has been removed.

- Pure fuzzy searching has several problems getting good TREC results. This is possibly due to the fact that, by its very nature, a fuzzy search uses less document-level information than is used by an exact-word search.

In short, for TREC the NexTrieve search engine must focus on document-level information in order to obtain good results. That being said, however, the other aspects of the NexTrieve search engine (fuzzy search, title text weighting, good proximity scoring) while not achieving high TREC scores are nevertheless valuable for other reasons.

Having a local (or "proximity") score take part in the overall score increases the "user-friendliness" of the system by having nicer previews arrive at the top of the result list. Ie, previews containing more of the search words appear first. This feature doesn' t improve TREC retrieval scores, but doesn' t harm them either.

Indexing title information is still valuable -- using NexTrieve it is possible to perform a search restricted to only title text (or subject text or whatever the type happens to be), resulting in significantly quicker searches. This feature doesn' t improve TREC retrieval scores, but doesn' t harm them either.

Fuzzy searching, although not providing good TREC results, still allows the user to perform searches that find information not otherwise obtainable by exact search methods. It should be noted that NexTrieve does not use any (language-dependant) stemming operations, and that the fuzzy search method employed by NexTrieve is language-independant.

NTT Question Answering System in TREC 2001

Hideto Kazawa, Hideki Isozaki and Eisaku Maeda
NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seikacho, Sorakugun, Kyoto, Japan
{kazawa,isozaki,maeda}@cslab.kecl.ntt.co.jp

Abstract

In this report, we describe our question-answering system SAIQA-e (System for Advanced Interactive Question Answering in English) which ran the main task of TREC-10's QA-track. Our system has two characteristics (1) named entity recognition based on support vector machines and (2) heuristic apposition detection. The MPR score of the main task is 0.228 and experimental results indicate the effectiveness of the above two steps in terms of answer extraction accuracy.

1 Introduction

To design a QA system, there are several choices to make. One is about what kind of technology the system should be based on. To date, some research works have attempted the Information Retrieval (IR) approach, assuming that the most relevant passages include the answers of questions. Generally speaking, this IR approach is fast and robust, but is unable to specify the 'exact answer', i.e., what part of the passage is really the answer. Another approach is the Information Extraction (IE) approach, where the system extracts candidate strings from documents and evaluates the validity of the candidates. This approach has the advantage of being able to specify the locations of exact answers although it is usually slow and often fails because of complicated natural language processing.

For TREC-10's QA track, we adopted the IE approach because we think that knowing the exact locations of answers is one of the important goals of QA. It also seems easier to reach the goal with the IE approach. To avoid 'deep' natural language processing, we decided to use only shallow linguistic analysis, i.e., part-of-speech tagging and base noun phrase (NP) chunking. To proceed with this decision, we mainly focused on the following problems.

1. **Learning extraction rules**

   Shallow linguistic analysis only gives 'low level' information and writing extraction rules manually with this information is quite a complicated job. Additionally, the written rules often lack readability and are hard to maintain. Therefore, we applied a machine learning method, support vector machines (SVM), to learn some extraction rules. SVM has shown a high performance in many pattern recognition and natural language processing tasks.

2. **Detecting apposition**

   To answer a certain kind of question, detecting an appositive relation is very important. As we looked further into this issue, however, we found that such detection is not easy because apposition is often determined by long-range constraints in sentences and cannot be identified only by neighborhood information. We therefore created a simple but effective heuristics to detect appositive relations.

This paper is organized as follows. In Section 2, we briefly describe the overall structure of our QA system SAIQA-e (System for Advanced Interactive Question Answering in English). Then, we

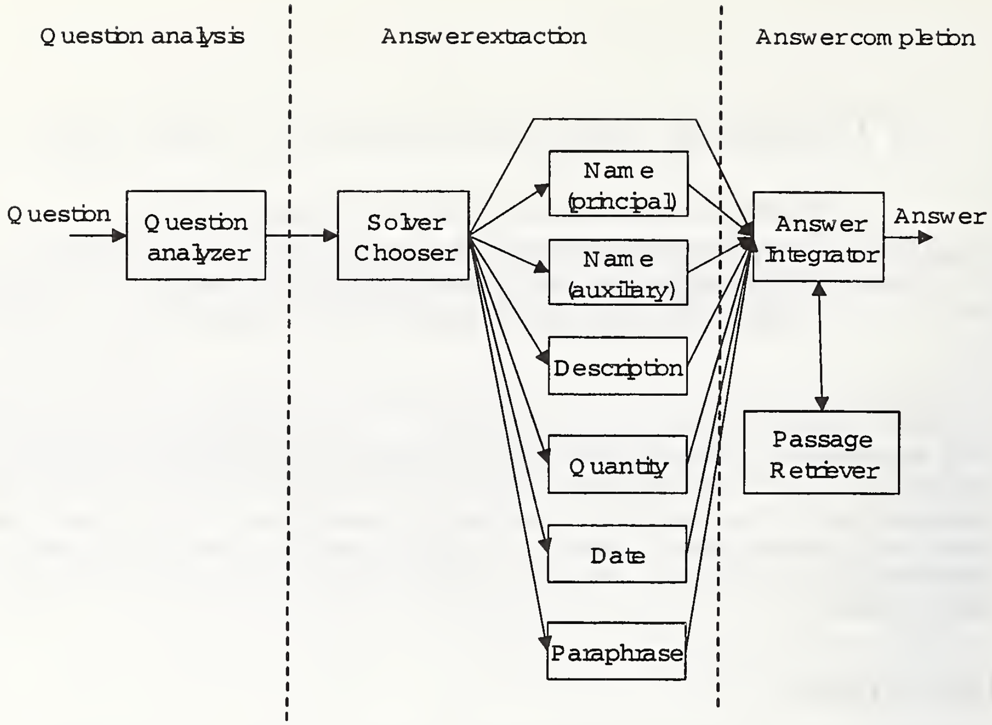


Figure 1: SAIQA-e overview

explain the approaches to our main problems, i.e., learning extraction rules and detecting apposition, in detail in Sections 3 and 4. Finally, in Section 5, we analyze results of SAIQA-e on the main task of TREC-10's QA track.

2 System overview

SAIQA-e first classifies a question into several categories (Figure 1). Then, it passes each question to an answer extraction subsystem, one that is specific to the question's category. Each extraction subsystem extracts candidate answers from a document set and scores them by heuristic measures. Finally, these candidate answers are merged when multiple candidates are located within a 50-byte length. If a question is categorized as 'unknown' or a specific solver does not extract five candidates, the system evokes a passage retrieval system, which extracts 50-byte passages. These passages are added to the candidates.

In the following, we describe the details of each component.

2.1 Question analysis

In the question analysis stage, each question is classified into one of the categories shown in Figure 2. The category is determined by a manually created decision tree and the following features of the question: question words (who, what, when,...), positions of the question words (start/middle/end of the question), first verb, head word of the first NP, head word of the second NP, and the word between the first and the second NPs.

Here, we explain the question categories.

- **Name**

Question type		Example
Name	Person	Who discovered x-rays? (Q 945)
	Location	Where is John Wayne airport? (Q 922)
	Organization	What is the name of the chocolate company in San Francisco? (Q 943)
	Others	What is the Ohio state bird? (Q 1001)
Description	Person	Who was Galileo? (Q 896)
	Others	What is an atom? (Q 897)
Date		When was Lyndon B. Johnson born? (Q 1060)
Quantity		What are the speed hummingbirds fly? (Q 953)
Paraphrase	Abbreviation	What is the abbreviation for Texas? (Q 1172)
	Fullname	What does I.V. stand for? (Q 1176)
	Nickname	What is Shakespeare's nickname? (Q 1294)
Unknown		What is the easiest way to remove wallpaper? (Q 1386)

Figure 2: Question categories

This category has four subcategories: Person (including entities treated like person, such as gods and comic characters), Location, Organization, and Others (including class names like animal species).

- **Description** This category has two subcategories, Person and Others.

  The distinction between Name-Person and Description-Person might be a little confusing, so let us present examples. "Who is the U.S. president?" is a Name-Person question because it asks about the name of a person who is the U.S. president. On the other hand, "Who is George W. Bush?" is a Description-Person question because it requires descriptive information about a person whose name is George W. Bush.

- **Quantity** This category has nine subcategories: Count, Age, Duration, Length, Money, Ratio, Size, Speed, and Others. As you can see, this category is rather broad and contains few related concepts. However, the expressions of these concepts are usually associated with numerical words and accordingly their extraction steps are expected to be similar. Based on this, we grouped these subcategories into one Quantity category.

- **Date** This category has four subcategories: Day, Day of the week, Year, and Others.

- **Paraphrase** This category has three subcategories: Abbreviation, Fullname, and Nickname. The category is created because these expressions are often related to the original expressions in unique ways (for example, an abbreviation follows an original expression in parentheses) and can be identified in a unified fashion.

The questions that are not classified into any above categories are labelled as 'Unknown' questions.

2.2 Answer extraction and evaluation

After the question is categorized, the answers are extracted from a document set and evaluated by heuristic measures. We have several extraction subsystems, each of which is intended to deal with only restricted types of questions. Each question is passed to the corresponding subsystem, while 'Unknown' questions skip this extraction step and are passed directly to the next answer integration step. Here, we describe the subsystems.

Principal Name Solver

This subsystem deals with Name-Person, Name-Location, and Name-Organization questions. It was separated from the other Name solver because detecting names of person/location/organization in documents is harder than other name detection and we wanted to focus our resources on this problem.

The subsystem first retrieves articles ranked by the fequency of keywords and their proximity. Then, an SVM-trained named entity detection module extracts person/location/organization names from these articles. These names are evaluated by heuristic measures such as the proximity to the keywords.

The issue of SVM learning of named entity detection is discussed in Section 3.

Auxiliary Name Solver

This subsystem deals with Name-Others questions. The extraction and evaluation are similar to Principal Name Solver's, but the name detection rules are manually created. Additionally, the evaluation heuristics are less accurate because Name-Others questions cover such diverse kinds of entities that it is hard to develop accurate category-specific measures such as those used in Principal Name Solver.

Description Solver

This subsystem accepts Description-Person and Description-Others questions. The extraction and evaluation are quite different from the name solvers'.

The subsytem first retrieves all articles including the name of the requested entity. (It is easy to identify the name in the question.) Then, the NPs appositively connected to the name are extracted as the descriptive answers. The answers with the same head are grouped as variant expressions of the same description. Finally, the most specific expressions of the groups are scored by the number of group members. (That is, a more frequent description is considered to be more trustable.)

Apposition detection plays the main role in Description Solver. We discuss this in Section 4.

Quantity/Date Solver

These subsystems deal with Quantity and Date questions. They are almost the same as Auxiliary Name Solver and the differences are in the extraction rules.

Paraphrase Solver

This solver deals with Paraphrase questions and the subsystem is quite different from other solvers.

For example, for Paraphrase-Abbreviation questions (for example, "What is the abbreviation for the United Nations"), it retrieves all articles in which the fullname (United Nations) appears. Then, a regular expression is used to extract all abbreviations from the articles. Finally, a sequence of upper characters in the fullname (UN) is compared to a sequence of upper characters in the abbreviations. This comparison is done approximately so that some missing characters are tolerated and the matching degree is translated into the score of the abbreviation.

2.3 Answer integration

After the answer extraction and evaluation stage, the answers are extended to 50 bytes and merged when the 50-byte passages contain multiple answers. Then, we add 'no answer' in the following manner.

1. If a question is classified into a category other than 'unknown' and the specific solver does not return as many as five answers, then 'no answer' (i.e., 'NIL') is added to the candidates. After that, the output of the passage retrieval system is added.

2. If a question is classified into the 'unknown' category and the passage retrieval system does not return as many as five answers, then 'no answer' (i.e., 'NIL') is added to the candidates.

We set all 'final answers' as '1', because SAIQA-e's outputs have already been sorted according to their relevance and we consider the first-ranked answer as the most trustable one.

3 Named Entity Recognition based on Support Vector Machines

Named entity (NE) recognition systems are useful for determining whether a certain pronoun designates a person or an organization or location. Although we have had our own Japanese NE systems, we did not have any experience on developing English NE systems. Therefore, we decided to develop one by using a corpus-based approach. Since we did not have any training data for English NE tasks, we prepared our own training data.

We employed support vector machines (SVM) for the English NE system. Such a system was proposed by Yamada et al. [YKM01] for Japanese NE recognition. His system is a simple application of Kudo's chunking system [KM01] that shows the best performance for the CoNLL-2000 shared task. We also implemented an SVM-based NE system for Japanese. This SVM-based NE system employs a different approach, but according to our experiments, this system is better than the other Japanese NE systems we have (a C4.5-based rule generation system [Iso01] and a system based on maximum entropy (ME) modelling). In the following sections, we describe our English NE systems.

3.1 Support Vector Machines

First, we introduce SVM briefly. The non-linear SVM classifier [SBS99] uses a decision function for an input vector $\vec{x}$ given by

$$f(\vec{x}) = \mathrm{sign}(g(\vec{x}))$$

where $\mathrm{sign}(y) = -1$ for $y < 0$ and $\mathrm{sign}(y) = 1$ for $y > 0$, and

$$g(\vec{x}) = \sum_{i=1}^{\ell} w_i k(\vec{x}, \vec{z}_i) + b.$$

$k(\vec{x}, \vec{z})$ is called a kernel function. Several kernel functions are known. By considering Japanese NE results, we decided to use a second-order polynomial kernel function $k(\vec{x}, \vec{z}) = (1 + \vec{x} \cdot \vec{z})^2$. The $\vec{z}_i$s are called support vectors that are representatives of training examples. w_is and b are constants determined by the training examples.

3.2 The first NE system

The first English NE system we implemented was a simple variation of ME-based NE systems proposed by Borthwick [Bor99] and Uchimoto [UMM$^+$00]. In this sytem, each word is classified into 21 classes: {PERSON,ORGANIZATION,LOCATION,FACILITY,ARTIFACT} × {SINGLE,BEGIN,MIDDLE,END}

$\cup$ {OTHER}. Here, (PERSON,SINGLE) is a label for a one-word person name like "George." (PERSON,BEGIN) is the first word of a certain multi-word expression for a person's name (e.g., "George" in "George Bush"). (PERSON,MIDDLE) indicates an internal word (e.g., "Walker" in "George Walker Bush"). (PERSON,END) is the last word (e.g., "Bush" in "George Bush"). When a word does not belong to any of the named entities defined above, it is labeled as OTHER.

In ME-based NE systems, the Viterbi algorithm is employed to get the best combination of labels. Since the ME model gives conditional probabilities, this is easy.

However, SVM does not tell us such probabilities. In addition, ordinary SVM can only solve two-class problems. Therefore, we built 21 SVM classifiers, i.e., one SVM for each class. For the application of the Viterbi algorithm, we used the sum of $g(\vec{x})$ instead of the sum of logarithms of probabilities. We used Kudo's TinySVM because of its faster speed over the well-known SVM light [SBS99] for this kind of task.

Since this first NE system classifies every word in a given document, the training data for each class has 10^5-10^6 examples. As a result, its training.took a very long time.

In our case, we applied the NE system to the TREC data after the training. It turned out that it was also too slow in the application phase. Because of this slowness, we could not try various combinations of possible features. In addition, we could not improve the QA system, which depends on the NE system. Therefore, we abandoned the first NE system.

3.3 The second NE system

We implemented another NE system in which hand-crafted rules were designed to detect NE candidates (roughly, noun phrases containing capitalized words) and then SVMs classified them into four classes: $C = \{\text{PERSON}, \text{ORGANIZATION}, \text{LOCATION}, \text{OTHER}\}$. For efficiency, we removed two classes, i.e., FACILITY and ARTIFACT, because they had only small numbers of positive training examples and their results were not very good.

In the second NE system, the features for classification include word strings, their memberships in word lists, their part-of-speech tags, word counts, neighbor words, appositive information, information about preceding occurrences in the same documents, and other surface features usually used in other NE systems. Since SVM allows only numerical values in the input, we have to convert features into a set of numerical vector components.

One example is represented by one numerical vector. Suppose an NE candidate's head word is *Washington*. Then, we introduce an axis for the feature *head_word_is_Washington*, and its value is 1. At the same time, the vector's incompatible axes like *head_word_is_University* have 0 as their values. In TinySVM, we have only to enumerate non-zero components.

For each candidate, the outputs of four functions, g_{PERSON}, $g_{\text{ORGANIZATION}}$, g_{LOCATION}, g_{OTHER}, are compared and the function that gives the largest value is chosen as class ($\text{argmax}_{c \in C} g_c(\vec{x})$).

The second NE system was found to be much faster than the first NE system, but it was still too slow for application to all TREC documents. Instead, we embedded the second NE system into the QA system and to be called on demand.

4 Description solver and Apposition detection

To determine which parts of documents contain descriptions of entities is difficult even for humans, but we provisionally adopted the following assumption.

- The description of an entity is expressed as the appositive modifier of the entity.

For example, in the sentence "George W. Bush, the U.S. president, said...", 'George W. Bush' is appositively modified by 'the U.S. president'. Therefore, 'the U.S. president' should be some description of 'George W. Bush'. This assumption makes the detection of an appositive relation the principal task in answering a description question.

Q. category	Num. of Q.	MPR (strict)
Name-{Per,Loc,Org}	131	0.349
Name-Others	76	0.096
Desc-{Per,Others}	128	0.247
Quantity	68	0.219
Date	48	0.119
Paraphrase	14	0.193
Others	27	0.151
Total	492	0.228

Table 1: Results of TREC10 QA track (main task)

In detecting appositive relations, punctuation disambiguation plays an important role. By 'punctuation disambiguation', we mean distinguishing the syntactic roles of commas. For example, in the sentence "When I was a kid, things were simple.", the comma is used as a marker of syntactic movement. On the contrary, in the sentence "George, the son of the former president, is a popular man.", the comma shows an appositive relation between 'George' and 'the son of the former president'. Note that in both examples, the commas are placed between noun phrases. This indicates that we cannot disambiguate this kind of comma usage only from neighbor information and punctuation disambiguation requires 'long-range' information.

We first used some off-the-shelf parsers to detect apposition. Unfortunately, we found that these parsers often failed around commas. We then created several heuristics to disambiguate punctuations and then to identify appositive relations. These heuristics classify punctuations into appositive markers, movement markers, and coordination markers (such as in "cats, dogs and birds").

Here are examples of the heuristics.

1. If a sentence starts with a subordinating conjunction, the leftmost comma in the sentence is a movement marker. (For example, "When I was a kid, TV was not popular.")

2. If a sentence contains the sequence of '(NP ,)+ NP CC NP', these commas are coordination markers.

5 Main task results

Table 1 shows the evaluation returned by NIST for each question category. These categorizations were manually done after the result was submitted.

Name-Person/Location/Organization result in the highest score (0.349) among all categories. This provides moderate but convincing evidence that our machine learning approach in NE recognition improves the answer extraction accuracy. The second highest is Description. Actually, this result was a little surprising for us because the extraction of the description was based on only a simple assumption (See Section 4).

References

[Bor99] Andrew Borthwick. *A Maximum Entropy Approach to Named Entity Recognition.* PhD thesis, New York University, 1999.

[Iso01] Hideki Isozaki. Japanese named entity recognition based on a simple rule generator and decision tree learning. In *Proceedings of Association for Computational Linguistics*, pp. 306–313, 2001.

[KM01] Taku Kudo and Yuji Matsumoto. Chunking with support vector machines. In *Proceedings of NAACL*, 2001.

[SBS99] Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors. *Advances in Kernel Methods.* MIT Press, 1999.

[UMM$^+$00] Kiyotaka Uchimoto, Qing Ma, Masaki Murata, Hiromi Ozaku, Masao Utiyama, and Hitoshi Isahara. Named entity extraction based on a maximum entropy model and transformation rules (in Japanese). *Journal of Natural Language Processing*, Vol. 7, No. 2, pp. 63–90, 2000.

[YKM01] Hiroyasu Yamada, Taku Kudo, and Yuji Matsumoto. Japanese named entity extraction using support vector machines (in Japanese). In *IPSJ SIG Notes*, 2001. NL-142-17.

Oracle at TREC 10: Filtering and Question-Answering

Shamim Alpha, Paul Dixon, Ciya Liao, Changwen Yang
Oracle Corporation
500 Oracle Parkway M/S 4op8
Redwood Shores, CA 94065 USA
trec@us.oracle.com

Abstract:
Oracle's objective in TREC-10 was to study the behavior of Oracle information retrieval in previously unexplored application areas. The software used was Oracle9i Text[1], Oracle's full-text retrieval engine integrated with the Oracle relational database management system, and the Oracle PL/SQL procedural programming language. Runs were submitted in filtering and Q/A tracks. For the filtering track we submitted three runs, in adaptive filtering, batch filtering and routing. By comparing the TREC results, we found that the concepts (themes) extracted by Oracle Text can be used to aggregate document information content to simplify statistical processing. Oracle's Q/A system integrated information retrieval (IR) and information extraction (IE). The Q/A system relied on a combination of document and sentence ranking in IR, named entity tagging in IE and shallow parsing based classification of questions into pre-defined categories.

1. Filtering based on Theme Signature

As a first time filtering track participant, Oracle submitted runs for adaptive filtering, batch filtering and routing this year. Only linear-utility optimized runs were submitted for adaptive filtering and batch filtering. The filtering system is built based on the Oracle 9i database with PL/SQL - an Oracle supported database access language. Since the routing sub-task outputs the top 1000 ranked documents per category, and the training process and similarity score calculation algorithm are the same for batch filtering and routing, we will focus our discussion on batch filtering and adaptive filtering.

The filtering system can be divided into three parts based on functionality:

a. Theme Vector Generation
b. Training
c. Classification

Theme Vector Generation

Theme vector generation generates a theme vector for each document. It is built-in functionality of Oracle Text, the information retrieval component of the Oracle database[2]. A theme vector containing a list of themes (concepts) and associated weights carries all information of a document used in classification. Themes are normalized words having meanings individually and extracted based on the Oracle Text knowledge base. The knowledge base is built in-house and contains about 425 thousand concepts classified into 2000 major categories. These categories are organized hierarchically under six top terms: *business and economics, science and technology, geography, government and military, social environment, and abstract ideas and concepts*. This knowledge base is built to support concept search and retrieval. For this TREC work, the ConText knowledge base was employed in our filtering system to preprocess documents and generate concept terms. Although the Oracle Text user extensible knowledge base functionality allows users to modify the built-in knowledge base using user specified thesaurus, we used the knowledge base without any modification. We believe augmenting the knowledge base using domain specific information could improve filtering performance. In the theme generation, known phrases are recognized using a greedy algorithm, unknown words and proper name phrases are recognized and treated as themes. Words and phrases are normalized to their canonical forms. Every normalized term is a potential theme for a document.

Theme weights are used to rank the semantic significance of themes to the aggregate document content. Themes are assigned initial weights based on their lexical flags in the knowledge base. Next, several factors derived from the structure of a document and the frequency of the theme in the document are employed to modify the initial weights of the themes. For example, the first few terms inside a sentence have higher weights than the terms at the end of sentences to account for "fronting" and sentence focus.

Generated theme vectors are normalized to have unity length before being sent to the training or classification process. This normalization can be written as :

$$w_j^n = \frac{w_j}{\sqrt{\sum w_j^2}}$$

where $w_j n$ and w_j are the j-th component (j-th theme term) weight of theme vector w after and before unity normalization respectively.

Our prior experience demonstrates that themes are superior to text tokens in representing text documents of medium to large size for classification purposes. Oracle Text first tokenizes documents and then processes these tokens using a greedy maximal match algorithm to generate themes. A brief description of the process to generate themes from tokens may shed some lights on the reason why themes are superior to tokens in classification. After finding a token, Oracle Text gets the part of speech information from the knowledge base or finds phrases based on the greedy algorithm and lexical knowledge base. If the token is a noun, a canonical form is used as a normalized form for this token, such as "tastefulness" with canonical form of "tasting" and "dull-headedness" with canonical form of "stupidity". If the token is a non-noun, a base form is found based on the knowledge base or morphology if the token does not exist in knowledge base. After that, a normalized noun form is used as the theme form for the non-noun base form. For example, "steadied" has a base form of "steady" which corresponds to a normalized form of "steadiness". The following differences between themes and tokens may contribute to the different behaviors in classification:

1. Themes can handle phrases while tokens can not without a lexicon.
2. Themes are represented with normalized forms of concepts, while tokens are forms with or without stemming. Word normalization is mostly based on lexical knowledge, while stemming of a token is mostly based on morphology.
3. The weight of a theme expresses the lexical information of a term, locations in a document, and term frequency. The weight of a token typically only includes the information of term frequency.

For the classification task no parent themes (broader terms) were used. Whether or not the parent themes improve the learning quality is actually an open question. One side says a specific word should be more important for representing a document and a parent theme may act as a common word. On the other hand, one of the parent themes may tell exactly what a document is about. However, that might depend on the level of parent theme and depend on whether or not the hierarchy of the knowledge base represents the same knowledge hierarchy in the classification application. We intend to investigate this issue thoroughly in the future.

Training

The training process calculates the summation of all relevant theme vectors for each category. The summation result serves as the original theme vector for one category. Because of accumulation, the number of themes in the category theme vector can be large. Experiments show that reducing some common themes and less frequent themes for the category theme vector can improve classification accuracy. Theme reduction can also reduce the resource usage and improve classification performance. We adopt a two-step theme reduction. The first step is to choose the top 400 themes with highest theme weights in the category theme vector. As mentioned earlier, the theme weight obtained from Oracle Text combines information about the lexical significance, word position inside one sentence, and occurrence frequency inside the document. Those top 400 themes in the category theme vector are the most frequently occurring and significant words to the category. Another rationale for choosing the theme by weights is that words with little meaning have lower weights and therefore can be removed.

The first step of theme selection based on the theme weight may choose some themes which are common in lot of categories. These common themes are not specific to one category and may produce extra noise to the classification process. The second step of theme reduction is to choose themes which are more specific to one category. We use a chi-square test for theme selection [3]. In specific, we choose a theme if the null hypothesis that this theme is independent of the considered category can be proved not true. The themes will be chosen if:

$$\frac{N(Nr_t - n_t R)^2}{Rn_t(N-R)(N-n_t)} > 3.84$$

where N is the total number of training documents

R is the number of training documents in this category

n_t is the number of training documents containing this word

r_t is the number of training documents in this category and containing this word.

value 3.84 is chosen because the confidence of chi-square test is 0.95.

By chi-square test, the average theme vector size can be reduced to 280. In the original category theme vector, the weight is the summation of each document's theme weights; those weights help us to choose the top 400 themes for the category. However, during the classification process, we use Robertson-Sparck Jones weights [4] as term weights in category theme vectors. The weights are calculated based on the statistical characteristics of the training set and relevant category:

$$\log\frac{(r_t+0.5)(N-R-n_t+r_t+0.5)}{(n_t-r_t+0.5)(R-r_t+0.5)}$$

This formula is obtained from the Bayesian statistical model. The Robertson-Sparck Jones weight is the component weight for one term to estimate the log-odds of an given document belonging to the considered category in the assumption that terms are independent [5].

Classification

Before classification, category theme vectors are normalized to have unity length. In classification, the similarity scores S between the incoming document and each category are calculated as a dot product between the document theme vector vd and category theme vector vc, that is S = vd.vc. The document is classified to the categories in which the similarity scores are larger than the corresponding category thresholds. The predefined thresholds are determined from the relevance information either from the training set in batch filtering or from feedback in adaptive filtering.

Threshold Determination

Batch filtering

Each category has its own threshold to determine if a document can be classified to it based on the similarity score. In order to determine the threshold for one category, we use the classification module to calculate the similarity score between all training documents and the considered category. For any given threshold x, we can get the following contingency table as we know the actual categories of each training document.

	Relevant	Not Relevant
Retrieved	R_+	N_+
Not Retrieved	R_-	N_-

We can define a utility (goal) function of the about 4 numbers, say $f(R_+,N_+,R_-,N_-,x)$. x appears explicitly in the function because R+,R-,N+ and N- are all functions of the threshold x. The threshold is chosen to maximize the function f.

$$\text{Threshold} = x: \max_{x} f(R_+,N_+,R_-,N_-,x)$$

In TREC-10, we submit the batch filtering run based on optimization function of linear-utility, which is $f(R_+,N_+)=T10U=2R_+ - N_+$.

In implementation, one can generate a sorted array of training documents ordered by similarity scores to the given category with a decreasing sequence. The relevance information of documents in the sorted array before any given document can determine R_+, N_+ at the threshold value equal to the similarity score of this document. For each document in the sorted array, one then can calculate the T10U function value at the threshold value equal to the similarity score of this document based on calculated R_+, N_+. Because the array is sorted such that the similarity scores are decreasing, one therefore can draw a curve of T10U vs threshold. As threshold decreases from the largest value, the T10U values first increase because more relevant documents are located at the positions having larger similarity scores, and decrease after reaching a peak. The peak position corresponds to a similarity score , whose value is the optimized threshold value to maximize T10U function. This calculation makes the assumption that the training set similarity score distribution and T10U quantity is similar to that of the test set.

Adaptive training

In adaptive filtering, we first built initial category theme vectors from training process of an initial training set, which contains two relevant documents per category. The training process is the same as we discussed above. The initial category threshold is set to be 40% of the minimum similarity score of the two relevant documents with the considered category. We then classify the test documents in a batch mode with each batch containing 2000 documents coming from the test set stream. After classification of each batch, feedback information including the relevance judgments and the similarity scores is sent to adaptive training, see Fig.1.

Adaptive training includes updating category theme vectors and category thresho!ds. In order to update the category theme vector, we have to maintain the original category theme vectors which are the theme vectors before any theme selections and has the theme weights from summation of Oracle Text theme weights. To keep the number of themes in the category theme vector from becoming too large, we limit the size of each original category theme vector to a maximum of 2000. The extra feedback training document theme vectors are added to the original category theme vectors using Widrow-Hoff algorithm [6].

$$w_j^n = w_j - 2z(\mathbf{w} \bullet \mathbf{x}_i - y_i)x_{i,j}$$

where w_j, $w^n{}_j$ are the weights for j-th component of the category theme vector before and after adaptive training, respectively. x_i is the theme vector of i-th feedback document, y_i the relevance judgment of the i-th feedback document with the considered category with y_i =0 denoting not relevant, y_i=1 denoting relevant. $\mathbf{w}.\mathbf{x}_i$ denotes the dot product between the theme vector **w** and $\mathbf{x}_i$. z>0 is learning rate and is set to 0.2.

The Widrow-Hoff algorithm generates a list of updated themes and weights. We maintain only the top 2000 highest weight themes for each category. The weights here are calculated quantities from Oracle Text theme weights. We apply theme selections and employ Robertson-Sparck Jones weights as category theme vector weights for classification as discussed in the above training section.

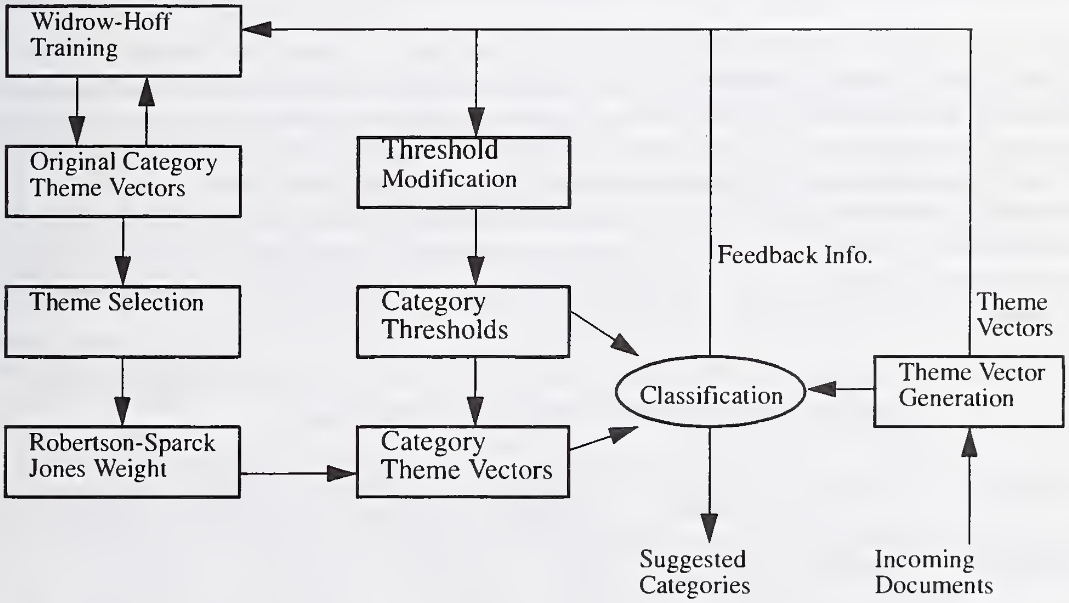


Figure 1. Adaptive filtering diagram

Thresholds can be calculated based on the relevance information and similarity scores of all previous feedback documents in the way we discussed in the threshold determination section. However, that calculation may take unacceptably long time. Instead we adopt a simple method to adjust the existing thresholds only based solely on current feedback information.

Thresholds can be adjusted by calculating the optimal threshold for the extra feedback training set as discussed in threshold determination section. We denote the optimal threshold as optimal_threshold_extra_training, then the updated threshold is :

updated_threshold = old_threshold + C (optimal_threshold_extra_training - old_threshold)

where C is a learning parameter and is set to 0.3. We note that the feedback batch size and the learning parameter C are relevant parameters, if the feedback batch size is small, the optimal threshold for the extra feedback documents may vary a lot, one then choose a smaller C. C has to be chosen such that the updated thresholds change with the feedback process in a systematic and stable way.

Submission Result and Discussions

Oracle submitted three runs. They are listed in the Table 1, and Table 2, with adaptive, batch runs in table 1 and routing in table2, respectively. The numbers in the parenthesis are the median value of all participants. The median values are the (N/2+1)-th value in sorted decreaseing list if the number of participants N is even. Except the precision for batch filter, all numbers in our submitted runs are above median.

We note that the routing behaves better than batch filtering. The fact that batch filtering system has only one more component: thresholding, than routing implies that our threshold determination is not quite good for batch filtering. In batch filtering, the threshold can not be adjusted. Once a threshold is determined, it is used to classify the whole test set without any adjustment. So the initial threshold determination is critical. However, it is interesting to note that the same simple method of determining threshold behaves quite well in adaptive filtering when comparing our adaptive filtering result with others.

Our training, classifying, and thresholding methods are all well-known methods, but our system behaves better than medians, especially in adaptive filtering. One explanation for this might be the linguistic suite in Oracle Text and knowledge base we used to process documents. The theme vector we get from Oracle Text contains more information than just text token and occurrence frequency in the document. Theme vector have a list of normalized terms. This term normalization could reduce the size of collection thesaurus, and make it easier to match different terms with the same concept. The weight of the theme contains not only the occurrence frequency information, but lexical information. In conclusion, the combination of these linguistic functionalities and appropriately engineering some well-known learning methods are believed to make our system successful.

Table 1: Adaptive and batch filtering result with T10U optimization. The numbers in the parathesis are the median value for all participants.

Run label	Run type	Optimization	Precision (median)	Recall (median)	T10SU (median)	F-beta (median)
oraAU082201	adaptive	T10U	0.538 (0.462)	0.495 (0.213)	0.291 (0.137)	0.519 (0.273)
oraBU082701	batch	T10U	0.556 (0.618)	0.353 (0.293)	0.249 (0.247)	0.450 (0.448)

Table 2: Routing result. The number in the parathesis is the median value for all participants.

Run label	Run type	Mean average precision (median)
oraRO082801	Routing	0.104 (0.082)

2. Question Answering based on Information Retrieval and Information Extraction

Questions can be classified into pre-defined categories. Typical categories are: person names, organization names, dates, locations (cities, countries, states, provinces, continents), numbers, times, meaning of acronyms and abbreviations, weights, lengths, temperatures, speed, manner, duration, products, reasons etc.[7][8]

Information extraction (IE) techniques allow us to extract lists of semantic categories from text automatically[9], such as person names, organization names, dates, locations, duration, etc., which are subsets of the whole pre-defined question categories. If a question category is covered by IE, finding the locations of answer candidates becomes easier: the task remains is to rank the list of answer candidates extracted by IE. Otherwise, a number of heuristics are employed to locate the answer candidates and rank them.

Overview of Oracle Q/A system:

Our Q/A system consists of three major components shown in figure2: (1) question processor (2) sentence ranking (3) answer extraction.

Question Processor:

Its role is to: (a) classify a question into a list of pre-defined semantic categories (b) extract content words from a question and send them to Oracle to retrieve relevant documents.

To classify a question, the first step is to determine its question type. The following wh-words are used to determine the question types: *who, why, where, whom, what, when, how much money, how much, how many, how* (rich, long, big, tall, hot, far, fast, large, old, wide, etc.).

A list of heuristics will help to map the question types to the pre-defined semantic categories:
(1) who is (was) "person name" => occupation
(2) other "who" types => personal name
(3) how rich, how much money, how much + VBD(VBP, VBZ, MD) => money expression
(4) other "how much" types => number
(5) how hot (cold) => temperature
(6) how fast => speed
(7) how old => age
(8) how long => period of time or length
(9) how big => length or square-measure or cubic-measure
(10) how tall (wide, far) => length

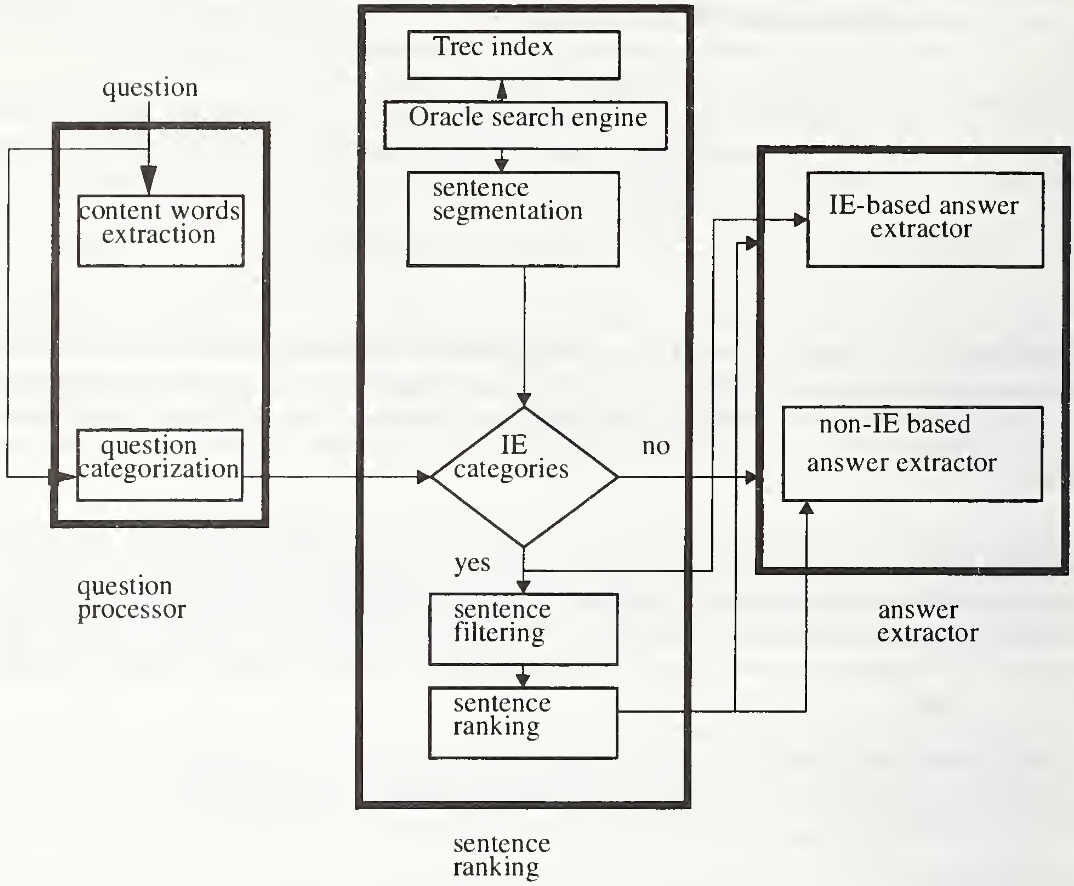


Figure 2: Architecture of the Oracle Q/A System

A complicated problem is to map the question type "what" to its semantic category. Here, a part-of-speech (POS) tagger is used to assign the most appropriate part-of-speech for each word in a question based on the contextual information [10]. The head noun of the first noun phrase in a question is used to decide its semantic category. For example, "What costume designer decided that Michael Jackson should only wear one glove?" The head noun of the first noun phrase is "designer". Using WordNet's lexicon [11], one finds that "designer" is a person, so, the semantic category of this question is "person name". If the head noun of the first noun phrase in a question is a stop word, then, the head noun of the second noun phrase is used to decide the semantic category. For example, "What was the name of the first Russian astronaut to do a spacewalk?" The head noun of the first noun phrase is "name" (a stop word), so, the head noun of the second noun phrase "astronaut" is used to decide the semantic category. Similarly, WordNet's API can tell that its semantic category is "person name".

When extracting a list of keywords from a question, our principle is to extract all content words, but ignore all non-content words. The distinction between these two types of words is that content words should appear in the relevant documents, but non-content words should not appear in the relevant documents. At lease, stop words and stop phrase (such as: how much, what time, what country) belong to non-content words. Furthermore, a list of heuristics is helpful to distinguish content words from non-content words. For example, "What is the length of coastline of the state of Alaska?", and "What is the Illinois state flower?" Word "state" is a non-content word in the first question, but a content word in the second question. Removing non-content words as many as possible makes retrieved documents more focusing on the subject topic of the question and is very helpful for extracting right answers from retrieved documents.

Sentence Ranking:

After the query processor extracts a number of content words from a question, two queries are formulated: one uses proximity operator “near" with maximum span size 25 to connect these words, the other uses “accum" operator to connect them. Near opearator find all query terms within specified span. Documents are ranked based on the frequencies and proximity of query terms in the document. Accum (accumulate) operator finds documents matching one or more query terms. Documents are ranked based on the sum of weights of the terms matched and frequency of the terms in the document. The first query has higher priority than the second one, because “near” operator always retrieves more relevant documents, but usually, the number of documents retrieved by “near” is not big enough, so, “accum” query is used to supplement it. Oracle Text retrieves a list of relevant documents (60 documents in trec10) based on the two queries. Then, the relevant documents are broken into paragraphs, the paragraphs are segmented into sentences. According to our experiments, it is suitable to extract long answers (250 bytes) from ranked paragraphs, but to extract short answers (50 bytes), the paragraphs must be further segmented into sentences.

Ranking the segmented sentences is based on the following information: (1) the number of unique content words in a sentence (2) tf and idf of each content word (3) total number of content words in a query (4) the smallest window size which contains all the unique content words in the sentence.

Our information extractor (IE) has two modules: one used for sentence filtering, the other used for answer extraction (IE-based answer extractor). If the semantic category of a question is covered by the IE, the IE is used for sentence filtering. Only selected sentences which satisfy the IE, are the candidates of the sentence ranking. For example, if the semantic category of a question is "person name", only the sentences which include at least one person name will participate the sentence ranking, all the rest of sentences are filtered out from answer extraction, because they do not include answers of the question. The IE was also integrated with sentence segmentation algorithm. The standard sentence delimiters are "?!.", followed by one or more spaces, then followed by a word whose first letter is a capital letter. There are many exceptional cases, such as Mr. Steve, St. Louis. The IE could recognize these exceptional cases, and guarantee the success of the sentence segmentation.

Answer Extraction:

After the sentences are ranked, top five of them are used to extract the answers. From previous description, our IE only covers a subset of the whole semantic categories. If the answer type of a question belongs to the subset, it is easy to extract answers using the IE. Otherwise, we concluded a number of heuristics, which help to extract answers. The sentence ranking algorithm can find the smallest window in a sentence, which contains all the content words in the sentence. This window divides the sentence into three parts: (1) the words in front of the window, (2) the words after the window and (3) the words inside of the window. According to our observation, the priorities of the three parts are (1) (3) (2). We further observed that in (1) and (3), the words closer to the windows have higher priority than others. Based on these observations, we picked up certain percent of words from each part of the sentence according to their priorities to form the final answers.

Other Linguistic Processing:

(1) acronyms and abbreviations: like other advanced search engines, our system also does limited automatic query expansion, mainly for queries with acronyms, abbreviations, etc. It expanded (a) acronyms of geographical terms, such as "U.S. = United States", "N.C. = North Carolina" (b) abbreviations of organization names, such as "YMCA = young mens christian association", "NBS = national bureau of standards"

(2) stemming: Oracle's search engine does not use Porter's stemmer. Our stemmer is more conservative, which obtains good precision, may hurt recall a little bit. To remedy this problem, extra stemming was

added in rare situations. For example, "When did Hawaii become a state?", the main verb was stemmed as "$become".

(3) Information Extractor (IE): an information extractor was created over the last few months to recognize (a) person names (b) organization names (c) dates (d) number (e) locations (f) money expression (g) time (h) temperature (I) speed (j) weight (k) length (l) square measure (m) cubic measure (n) age, etc.

Performance Evaluation:

A question answering system was created based on information retrieval and information extraction. Our study shows that traditional IR technique are not only useful to rank documents, but also to rank paragraphs and sentences. Finding the smallest window from a sentence which contains all the content words in it, is very helpful to extract answers when its semantic category is not covered by the IE, the window size is also an important factor to decide the sentence rank.

The following table shows the evaluation result provided by NIST for our system

	strict	lenient
NIST score	0.477	0.491
% of correct answers	60.77%	62.60%
% of correct first answers	40.04%	40.85%

The current (Oracle 9i) knowledge base is designed for information retrieval; for Q/A track, we found it necessary to expand the lexicon to cover wh-focus ontological facets.

3. Web Track

As preparation, we investigated the TREC-10 web task using TREC-9 web track documents and queries. We also attempted to productize lessons learnt from our participation in Trec8 adhoc manual task. A set of different collections including TREC Web and Adhoc collections helped us in our effort to formulate generic techniques applicable across domain. Due to resource constraints, we were unable to work on Trec10 web track. Here we summarize our findings based on older collections.

Our experiments in link analysis using Oracle intranet data indicate that link analysis adds little value to intranet search. Link analysis is a technique that helps bring order to an unorganized collection lacking central authority (such as web) by using popularity measure. A organized intranet will have clearly defined authorities for different subject matters.

IDF weighting used in tf-idf scoring is not very effective when the collection is pretty large (a couple of million documents) and number of terms in the queries is pretty high. If the queries are free-text queries, IDF weighting fails to distinguish between important and unimportant terms. Weighting techniques which weight terms inversely proportional to a factor of the frequency ratios (x times as rare terms get y times as much weight) seem to perform better in this situation. We saw significant improvement in R-precision by adopting this technique.

As the number of documents increases, the number of distinct score values supported by a system becomes important. Until recently Oracle Text used 100 distinct integers in the range of 1 to 100 for scoring. We found that allowing a million distinct values improves system IR quality computed in average precision by improving tie splitting. Even though number of relevant documents retrieved did not increase very significantly (about 3-4%), average precision increased by 10-15% (for example, Trec9 web track average precision improved from 0.11 to 0.125).

Using Trec8 and Trec9 collections, we identified a few simple flaws in our system which have been removed. On average, recall has increased by about 25% and precision at 10 has improved by more than 50%. We ran this out-of-box automatic system against TREC-8 adhoc task. Oracle TREC-8 manual task submission received an average precision score of 0.42. Out of 50 benchmark queries, performance (number of relevant retrieved) is tied for 10 queries, 20 won by manual and 20 won by automatic.

References

[1] Oracle Technet Text Homepage (*http://technet.oracle.com/products/text*)

[2] K.Mahesh, J. Kud, and P. Dixon, Oracle at Trec8: A Lexical Approach, *in Proceedings of the Eighth Text Retrieval Conference (TREC-8)*, 1999.

[3] C.D. Manning and H. Schutze, Foundations of Statistical Natural Language Processing, the MIT press, 2000.

[4] S. E. Robertson and K. Sparch Jones, Relevance weighting of search terms, *Journal of the American Society for Information Science*, **27**, 129-146, 1976.

[5] K. Sparck Jones, S. Walker and S.E.Robertson, A probabilistic model of information retrieval: development and status, *Technical Report TR446 Cambridge University Computer Laboratory*, 1998.

[6] D. D. Lewis, R. E. Schapire, J.P. Callan and R. Papka, Training algorithms for linear text classifications, in SIGIR'96.

[7] Dan Moldovan, Sanda Harabagiu. Lasso: A tool for surfing the Answer Net. In the *Proceedings of the Text Retrieval Conference (TREC-8)*, 1999.

[8] Sanda Harabagiu, Dan Moldovan. Falcon: Boosting Knowledge for Answer Engines. In the *Proceedings of the Texi Retrieval Conference (TREC-9)*, 2000.

[9] Rohini Srihari and Wei Li. Information Extraction Supported Question Answering. In the *Proceedings of the Text Retrieval Conference (TREC-8)*, 1999

[10] Eric Brill, Some Advances in Transformation-Based Part of Speech Tagger. In the *Proceedings of AAAI*, 1994

[11] G.A. Miller, WordNet: A Lexical Database. *Communication of the ACM*, **38**, 39-41, November 1995

Observations of Searchers: OHSU TREC 2001 Interactive Track

William Hersh
Lynetta Sacherek
Daniel Olson

Division of Medical Informatics & Outcomes Research
Oregon Health & Science University
Portland, Oregon, USA

{hersh, sacherek, olsondan}@ohsu.edu

Abstract

The goal of the TREC 2001 Interactive Track was to carry out observational experiments of Web-based searching to develop hypotheses for experiments in subsequent years. Each participating group was asked to undertake exploratory experiments based on a general protocol. For the OHSU Interactive Track experiments this year, we chose to perform a pure observational study of watching searchers carry out tasks on the Web. We found users were able to complete almost all the tasks within the time limits of the protocol. Future experimental studies aiming to discern differences among systems may need to provide more challenging tasks to detect such differences.

Background

At the SIGIR 2000 workshop, Interactive Retrieval at TREC and Beyond, the Interactive Track decided on a number of new directions for TREC 2001 and beyond [1]. One of these decisions was to move the track to a two-year cycle, which would provide time to refine the track protocol and collect adequate amounts of data. In addition, it was agreed upon that the TREC 2001 track activities would consist of observational studies aiming to view searchers in realistic searching situations and to generate hypotheses that could assessed when the track returned to experimental studies in the following year.

For TREC 2001, participants in the Interactive Track carried out observational studies that aimed to maximize the realism of the searching by allowing the use of data and search systems/tools publicly accessible via the Internet. Within the framework of a broad protocol, groups were encouraged to allow their searchers to choose tasks and systems/tools for accomplishing those tasks. Groups were also asked, however, to maximize the likelihood they would find in their observations a hypothesis they could test for TREC 2002.

The OHSU group chose to undertake a purely observational study for TREC 2001, watching searchers carry out the assigned tasks with Web tools they desired to use. Users were allowed to choose whatever tools they desired for each questions, with their actions logged via the browser history files. They were also administered a short questionnaire after each question about their knowledge of the topic, ease of searching, and perceived success of searching.

Methods

The OHSU Interactive Track group performed an observational study that adhered to the guidelines of the track protocol. Each subject was asked to perform a search for each of four domains:

1. Finding consumer medical information on a given subject
2. Buying a given item
3. Planning travel to a given place
4. Collecting material for a project on a given subject

Per the track protocol, two of the questions would be fully-specified (i.e., user given the complete task) and the other two would be partially-specified (i.e., the user would decide part of the task). As the protocol had four domains, two search types (fully-specified and partially-specified), and two questions for each domain and type, a total of 16 questions were available (see Table 1). For data analysis, a three letter/number identifier was developed to identify each question based on the domain, type, and question number. The questions and their order were permuted for each searcher to insure each searcher would have one each from the four domains that varied in terms of type, question number, and the order administered.

Searchers were asked to search on each question as they would normally search on the Web, using the search engines, catalogs, etc. that they would typically use. They were instructed to provide a pertinent answer to each question along with the URL(s) to justify it. After they finished searching, they were asked to answer five questions on a Likert scale of 1 (not at all) to 5 (extremely). These statements were:

1. Are you familiar with this topic?
2. Was it easy to get started on this search?
3. Was it easy to do the search on this topic?
4. Are you satisfied with your search results?
5. Did you have enough time to do an effective search?

The searchers were given up to 20 minutes to complete each question but were asked to stop if they completed it sooner, with the elapsed time recorded. Each Web page they viewed was tracked by the Netscape History file, which we cleared before the searcher arrived and saved when the session was over.

For each question, we obtained a number of variables. We first determined whether the task was completed successfully. We did not check whether the user answered the question "correctly," but rather determined whether he or she listed an appropriate answer (or number of answers) pertinent to the task. We also measured the time taken for the question, the number of pages viewed, and the number of pages listed as justifying the answer.

Similar to previous track years, we recruited experienced searchers who were librarians or information professionals in the Pacific Northwest or medical informatics graduate students at OHSU. The experiments took place in computer training rooms at OHSU where searchers could choose to use either a Windows or Macintosh computer connected to the Internet running Netscape Navigator. Searchers were given a ten-minute orientation describing the nature of the experiment and the plan for the two-hour session. They also performed a practice search which asked them, "Find universities that offer a graduate degree in computer science in Oregon."

Because our experiment was exploratory in nature, we had no a priori hypotheses and thus performed no statistical analyses.

Results

A total of 24 searchers were recruited. All were highly experienced Web searchers, either as information professionals or graduate students. All subjects completed with experiments without difficulty. Four questions were discarded due to incomplete data collection, so a total of 92 questions were analyzed. A wide variety of topics were chosen for the partially-specified questions, as shown in Table 2.

Table 3 shows the overall results at the per-question level and two levels of aggregation. As seen in the first section of Table 3, virtually all subjects completed the tasks correctly. The average time taken was well under the 20 minutes allowed. The lower sections of Table 3 show aggregation of the results by domain-search type and further by domain. The differences across domains are small, but the buying task did take the most time and required the most page views.

Table 4 shows the results of the post-searching questions. While searchers were equivocal about their prior knowledge of the topic, they were consistent in their high belief that the search was easy to do and provided satisfactory results within an adequate amount of time.

We also looked at the use of search engines. A total of 68 questions (74%) employed the use of one search engine. Sixteen questions (17%) used no search engines, while seven (7%) employed two different ones and one (1%) used four. Table 5 shows the search engines used. Google was the most commonly used search engine, with Yahoo being the only other one used more than five times.

Further analysis planned after the TREC meeting for the final proceedings paper will focus on analyzing the search engines and catalogs used along with the queries posed to them.

Conclusions

Our study demonstrates that experienced Web users are able to perform relatively challenging tasks successfully using the search engines and catalogs that are available. This result is not new, but does demonstrate that such information tasks can be performed by a wide cross-segment of experienced and educated users.

This study does not provide any major new insights into user searching, but it clearly does indicate that experiments aiming to discern differences among systems will need to employ tasks that are more difficult. Otherwise users will be able to complete tasks no matter what systems they use, and any differences among systems will not be detected. This study also shows that users employ a variety of high-quality, widely-available tools in their searching. Experiments that focus on single systems or user interfaces may not provide the diversity of approaches that would accommodate all searchers.

References

1. Hersh W, Over P, SIGIR workshop on interactive retrieval at TREC and beyond, SIGIR Forum, vol. 34, no. 1.

Table 1 - Searching questions with domain (medical, buying, travel, or project), search type (fully-specified or partially-specified), number, and text. Blank line indicated area for user choice in partially-specified questions.

Domain	Search Type	Question Number	Question Text
M	F	1	Tell me three categories of people who should or should not get a flu shot and why.
M	F	2	Find a website likely to contain reliable information on the effect of second-hand smoke.
M	P	1	List two of the generally recommended treatments for _____.
M	P	2	Identify two pros or cons of taking large doses of _____.
B	F	1	Get two price quotes for a new digital camera (3 or more megapixels and 2x or more zoom).
B	F	2	Find two websites that allow people to buy soy milk online.
B	P	1	Name three features to consider in buying a(n) _____.
B	P	2	Find two websites that will let me buy a(n) _____ online.
T	F	1	I want to visit Antarctica. Find a website with information on organized tours/trips there.
T	F	2	Identify three interesting things to do during a weekend in Kyoto, Japan.
T	P	1	Identify three interesting places to visit in _____.
T	P	2	I'd like to go on a sailing vacation in _____, but I don't know how to sail. Tell me where can I get some information about organized sailing cruises in that area.
P	F	1	Find three articles that a high school student could use in writing a report on the Titanic.
P	F	2	Tell me the name of a website where I can find material on global warming.
P	P	1	Find three different information sources that may be useful to a high school student in writing a biography of ________.
P	P	2	Locate a site with lots of information for a high school report on the history of _____.

Table 2 - Topic designations for partially-specified question.

Searcher ID	Question ID	Partial Topic Text
4	BP1	TV
5	BP1	SUV
12	BP1	DVD Player
13	BP1	bicycle
20	BP1	house
21	BP1	hybrid car
3	BP2	Laptop computer
7	BP2	book
11	BP2	airplane
15	BP2	bassinet
19	BP2	set of dishes
23	BP2	(4 door) car-compact
3	MP1	Heart Disease/Atrial fibrillation
8	MP1	Achalasia
11	MP1	eczema
16	MP1	breast cancer
19	MP1	sunburn
24	MP1	scabies
2	MP2	Steroids
10	MP2	pseudoephedrine HCI
14	MP2	aspirin
18	MP2	Vitamin C
22	MP2	vitamin C
2	PP1	Shakespeare
7	PP1	Abraham Lincoln
10	PP1	Virginia Woolf
15	PP1	Amelia Earhart
18	PP1	Ayn Rand
23	PP1	Mark Twain
1	PP2	Turkey
4	PP2	cat worship
5	PP2	the Great wall of China
9	PP2	Mesopotamia
13	PP2	computers
17	PP2	Eleanor Roosevelt
21	PP2	WWII
1	TP1	Santa Fe, NM
9	TP1	Greece
14	TP1	Burma
17	TP1	Berlin
22	TP1	Oregon
4	TP2	Tahiti
8	TP2	wales
12	TP2	Hawaii
16	TP2	The Galapagos
20	TP2	the Bahamas
24	TP2	San Juan Islands

Table 3 - Overall results by individual question and averaged by full-partial status and domain.

Question ID	Number of Searchers	Number Correct	Percent Correct	Time (minutes)	Pages Viewed	Pages Listed
By Individual Question						
BF1	5	4	80%	11.00	15.20	2.00
BF2	6	6	100%	9.50	29.83	2.17
BP1	6	6	100%	7.00	15.17	3.17
BP2	6	6	100%	8.67	15.50	2.67
MF1	6	6	100%	8.50	9.50	2.00
MF2	6	6	100%	5.83	10.00	2.33
MP1	6	5	83%	6.00	11.17	1.50
MP2	5	4	80%	11.80	12.60	2.00
PF1	5	5	100%	5.80	12.40	2.80
PF2	5	5	100%	3.80	5.80	2.40
PP1	6	6	100%	8.17	12.17	3.17
PP2	7	7	100%	7.14	13.29	2.29
TF1	6	6	100%	6.00	13.83	1.17
TF2	6	5	83%	7.50	15.00	1.67
TP1	5	5	100%	6.80	14.20	2.80
TP2	6	6	100%	7.83	12.00	3.00
By Full-Partial Question						
BF	5.5	5	91%	10.25	22.52	2.08
BP	6	6	100%	7.83	15.33	2.92
MF	6	6	100%	7.17	9.75	2.17
MP	5.5	4.5	82%	8.90	11.88	1.75
PF	5	5	100%	4.80	9.10	2.60
PP	6.5	6.5	100%	7.65	12.73	2.73
TF	6	5.5	92%	6.75	14.42	1.42
TP	5.5	5.5	100%	7.32	13.10	2.90
By Domain						
B	5.75	5.5	96%	9.04	18.93	2.50
M	5.75	5.25	91%	8.03	10.82	1.96
P	5.75	5.75	100%	6.23	10.91	2.66
T	5.75	5.5	96%	7.03	13.76	2.16

Table 4 - Questionnaire results by individual question and averaged by full-partial status and domain.

Question ID	Familiar with topic	Easy to get started	Easy to do search	Satisfied with results	Enough time to search
By Individual Question					
BF1	2.80	5.00	4.80	3.80	4.80
BF2	2.17	4.33	3.67	4.33	4.83
BP1	2.33	4.50	4.50	4.67	4.83
BP2	3.17	4.67	4.50	4.33	4.83
MF1	2.67	4.83	4.33	4.83	5.00
MF2	2.67	4.33	4.17	4.67	4.83
MP1	3.33	3.83	3.83	4.17	4.00
MP2	3.20	4.20	3.80	3.40	4.00
PF1	2.80	4.60	4.60	4.40	4.80
PF2	2.60	4.40	4.80	4.20	4.80
PP1	3.17	4.83	5.00	4.67	4.67
PP2	3.00	4.71	4.43	4.43	4.71
TF1	1.17	3.83	3.83	3.83	4.67
TF2	1.67	4.67	4.33	4.33	4.67
TP1	3.00	4.60	4.80	4.80	5.00
TP2	1.67	4.33	4.33	4.33	4.67
By Full-Partial Question					
BF	2.48	4.67	4.23	4.07	4.82
BP	2.75	4.58	4.50	4.50	4.83
MF	2.67	4.58	4.25	4.75	4.92
MP	3.27	4.02	3.82	3.78	4.00
PF	2.70	4.50	4.70	4.30	4.80
PP	3.08	4.77	4.71	4.55	4.69
TF	1.42	4.25	4.08	4.08	4.67
TP	2.33	4.47	4.57	4.57	4.83
By Domain					
B	2.62	4.63	4.37	4.28	4.83
M	2.97	4.30	4.03	4.27	4.46
P	2.89	4.64	4.71	4.42	4.75
T	1.88	4.36	4.33	4.33	4.75

Table 5 - Frequency of use of search engines.

Search Engine	Times used
Google	45
Yahoo	20
Ask Jeeves	5
Alta Vista	4
Ask	4
Metacrawler	4
Netscape	3
Excite	2
Lycos	2
AOL	1
LookSmart	1

SiteQ: Engineering High Performance QA system Using Lexico-Semantic Pattern Matching and Shallow NLP

Gary Geunbae Lee, Jungyun Seo*, Seungwoo Lee, Hanmin Jung, Bong-Hyun Cho, Changki Lee, Byung-Kwan Kwak, Jeongwon Cha, Dongseok Kim, JooHui An, Harksoo Kim**, Kyungsun Kim**

Dept. of Computer Science & Engineering, POSTECH
Dept. of Computer Science, Sogang University*
DiQuest.com, Inc.**

Abstracts

In TREC-10, we participated in the web track (only ad-hoc task) and the QA track (only main task).

In the QA track, our QA system (SiteQ) has general architecture with three processing steps: question processing, passage selection and answer processing. The key technique is LSP's (Lexico-Semantic Patterns) that are composed of linguistic entries and semantic types. LSP grammars constructed from various resources are used for answer type determination and answer matching. We also adapt AAD (Abbreviation-Appositive-Definition) processing for the queries that answer type cannot be determined or expected, encyclopedia search for increasing the matching coverage between query terms and passages, and pivot detection for the distance calculation with answer candidates.

We used two-level answer types consisted of 18 upper-level types and 47 lower-level types. Semantic category dictionary, WordNet, POS combined with lexicography and a stemmer were all applied to construct the LSP knowledge base. CSMT (Category Sense-code Mapping Table) tried to find answer types using the matching between semantic categories and sense-codes from WordNet. Evaluation shows that MRR for 492 questions is 0.320 (strict), which is considerably higher than the average MRR of other 67 runs.

In the Web track, we focused on the effectiveness of both noun phrase extraction and our new PRF (Pseudo Relevance Feedback). We confirmed that our query expansion using PRF with TSV function adapting TF factor contributed to better performance, but noun phrases did not contribute much. It needs more observations for us to make elaborate rules of tag patterns for the construction of better noun phrases.

1. Introduction

The goal of the QA track is to foster research on systems that retrieve answers rather than documents in response to a question [11][12]. The focus is on systems that can function in unrestricted open domains [11].

The web track features ad hoc search tasks on a document collection that is a snapshot of the World Wide Web. The main focus of this track is to form a Web test collection using pooled relevance judgments. We will describe our systems and experiences for both QA and Web tracks in this paper.

2. QA track: Systems and Experiences

In TREC-10, the QA track consisted of three separate tasks: the main task, the list task and the context task. We participated in only the main task.

The main task is similar to the task in previous QA tracks (TREC-8, TREC-9). NIST provided 500 questions that seek short, fact-based answers. Some questions may not have a known answer in the document collection. In that case, the response string "NIL" is judged correct. This differs from the previous QA tracks and makes the task somewhat more difficult. The answer-string should contain no more than 50 bytes; 250-byte runs were abandoned this year. Participants must return at least one and no more than five responses per question ranked by preferences.

The document collection consists of the following six data sets: AP newswire, Wall Street Journal, San Jose Mercury News, Financial Times, Los Angeles Times, and Foreign Broadcast Information Service. The documents are SGML tagged, and each document in this collection has a unique identifier in the field.

Distinguished from an information retrieval, a QA system must retrieve answers rather than documents as responses to a question. As an ordinary course of step, we focused on what can be a possible answer, how our system can determine the answer type of a question, and how our system can detect instances of each answer type in a document. We classified possible answers and designed a method for determining the answer type of each question and detecting instances of it in a document. We have not constructed the index of document collection this time and instead used the ranked document list provided by NIST for each question.

Our QA system, SiteQ, consists of three important steps; question processing, passage selection and answer processing, which will be explained in detail.

2.1 Question Processing

In general, a question answering system analyzes an input question at first step. It is important to understand what a user wants to find; whether it is person's name, location, organization, or any other types. To do so, we first classified the types of possible answers [1][2][3][6] and used Lexico-Semantic Patterns (LSP) to determine the answer type of a question.

2.1.1 Answer Type

We classified the type of answers to fact-seeking questions [12]. Referring to the types used in FALCON [3], we analyzed the questions used in the previous QA tracks and their answers judged correct and constructed 2-level hierarchy of answer types. Hierarchical structure of answer types is useful since only YEAR is available for 'what year' question, but YEAR, MONTH, DAY, or TIME is available for 'when' question. Our answer type has 18 types at top level as shown in the box.

QUANTITY DATE TIME PROPERTY LANGUAGE_UNIT LANGUAGE SYMBOLIC_REP ACTION ACTIVITY LIFE_FORM NATURAL_OBJECT LOCATION SUBSTANCE ARTIFACT GROUP PHENOMENON STATUS BODY_PART

2.1.2 Lexico-Semantic Patterns

Usually an interrogative in a question is an important factor but it is not enough to determine the answer type. LASSO first determined the question class and the question focus, and then determined the answer type by using them [6]. The question class is defined as an interrogative and the question focus is defined as the main information required by the interrogation.

We used Lexico-Semantic Patterns (LSP) to determine the type of answer expected. Usually in addition to an interrogative in a question, its surrounding words or their senses are expressed in LSP, which substitutes the question class and focus word.

LSP grammar is composed of condition part and conclusion part. The conclusion part is the type of answer expected if the LSP in condition part is matched. LSP is composed of lexical entries, POS tag, semantic category and their sequence, and is expressed in regular expression. For example, a grammar *"(%who)(%be)(@person) → PERSON"* can be constructed from a question *"Who was President Cleveland's wife?"*. '*%who*' and '*%be*' is lexical entries and '*@person*' is a semantic category for representing the position of a person. We have manually constructed LSP grammar from the questions used in the previous QA tracks and the questions gathered from the Web by ourselves. Among them 361 entry LSP grammar was used for this year's QA track.

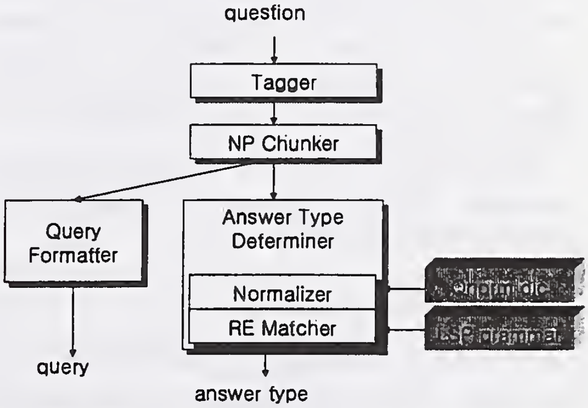


Figure 1 Question processing

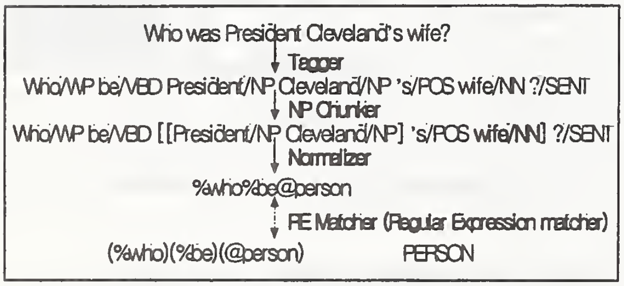
2.1.3 Determining The Answer Type

Figure 1 explains the procedures to determine the expected answer type of an input question. At first, an input question is POS-tagged using POSTAG/E English tagger and, at the second step, noun phrases are detected by NP chunker. Scanning the tagged question from right to left, this module detects the boundary of noun phrase and its head noun. To do this, we collected the POS patterns for noun phrases from the questions. A noun phrase almost always ends with a noun, usually starts with a pre-determiner, a determiner, an adjective, a possessive pronoun, or a noun. The rightmost noun in a noun phrase is selected as a head noun. Two noun phrases can be combined into a larger noun phrase by connecting them using a preposition '*of*' or a possessive ending. In case of a preposition '*of*', the head of its left-side noun phrase is selected as a head of the combined noun phrase, but in case of a possessive ending the head of its right-side noun phrase is selected.

It is important that detecting a head in a noun phrase since the sense of the head noun plays an important role in determining the expected answer type but its modifiers are useful for justifying final answers. In the above question, "*President Cleveland's wife*" is detected as a noun phrase, and '*wife*' is its head and clarifies the answer type of the question is PERSON. In contrast to this question, the expected answer type of a question "*Who is Cleveland?*" will be POSITION, which means the position of Cleveland (i.e., president) will be an answer.

At the third step, based on normalization dictionary (Qnorm dic) and WordNet, each word in a question is converted into LSP code to be matched with the condition part of LSP grammar by regular expression. "*President Cleveland's wife*" is converted into '*@person*' since it is a noun phrase and its head is '*wife*', of which semantic category is '*@person*'.

The following box shows how the answer type of a question "*Who was President Cleveland's wife?*" is determined as PERSON.


Who was President Cleveland's wife?
Tagger
Who/WP be/VBD President/NP Cleveland/NP 's/POS wife/NN ?/SENT
NP Chunker
Who/WP be/VBD [[President/NP Cleveland/NP] 's/POS wife/NN] ?/SENT
Normalizer
%who%be@person
RE Matcher (Regular Expression matcher)
(%who)(%be)(@person)
PERSON

2.2 Passage Selection

We have not constructed an index database from the document collection since we had no enough time and computing resources this year. Therefore we couldn't help using only the document list provided by NIST and selecting relevant passages from them by scanning the whole documents and matching the keywords. The documents were ranked by document similarity because they were retrieved by the PRISE [7], a document retrieval system rather than a passage retrieval system. Generally, however, a document does not fit for detecting candidate answers within itself since it is too large and contains too much extra information. By analyzing the previous questions and their answers, we can assume that answers to a question usually occur comparatively near to the matched keywords in a document. This means that the answer can occur in any ranked documents and we had better select passages from each document and rank them by passage similarity. Then we can use top passages to find candidate answers. To do so, we first must define passage and keywords to be used in selecting relevant passages.

2.2.1 Keywords

We define keywords to be used in selecting passages from the retrieved documents. We first remove useless words in a question and then use the remained words as three types of keywords considering lexical normalization and semantic similarity. Finally we assign weights to each keyword.

- Removing stop words

The useless words in a question are removed first by POS tag and stop word list, which has 568 entries. Then the following five heuristics are applied to the remaining words.

a. When a word like 'kind', 'sort', 'one', 'most', etc. occurs in the left side of a preposition 'of', it is removed; eg) What *kind* of dog ...? Name *one* of the Seven Wonders ...?
b. When a word like 'name', 'nickname', etc. occurs in the right side of a possessive ending, it is removed; eg) What was the man's *name* who was killed ...? What is Shakespeare's *nickname*?
c. When a question is expressed in imperative sentence, the imperative verb is removed; eg) *Tell* me what city ...?
d. When a verb needs a to-infinitive, the verb is removed; eg) Where do lobsters

like to live?

e. When an adjective or an adverb follows an interrogative 'how', the adjective or adverb is removed; eg) How *wide* is the Atlantic Ocean?

- The type of keyword

After removing all stop words, the remaining words are considered as question keywords. We define following three types of keyword to solve the mismatching problem of keywords caused by lexical variants and synonyms.

a. *Lemma form*

The lemma form of a word is used as a keyword except the superlative adjective or adverb, in which case the word itself is used as a keyword; eg) invented → *invent*, inventers → *inventer*, smallest → *smallest*

b. *Stemmed form*

Though the lemma form solves somewhat of the mismatching problem, it is not enough to solve the mismatch between 'inventer' and 'invented'. This can be resolved by using a stemmer like the Porter's stemmer [8].

c. *WordNet sense (in case of noun or noun phrase)*

To match a word 'ship' in a question with a word 'steamship' in a document, we must compute semantic similarity between a question keyword and a document word. Using the WordNet [5], the synonym or hyponym of a question keyword occurring in documents is matched with the question keyword.

- The weight of the keyword

The lemma form is weighted by its part of speech. A proper noun, a common noun starting with a capital letter, and a superlative has higher weight than a verb, an adjective and an adverb. The stemmed form has some of the weights its lemma form has. The keyword (noun or noun phrase) matched by WordNet sense has the lowest weight relative to the number of its component words.

2.2.2 Passages

A passage is composed of more than one sentence segmented by punctuation. We make adjacent two sentences into a passage if they have a lexical chain, which indicates that a sentence has a noun and the other sentence has its anaphora. We however limited a passage to maximum three sentences since the more sentences have the more extra information, which may increase incorrect candidate answers.

Each sentence from a document gets scored by matching its terms with query terms (Score$_1$) and by considering the distance and number of the matched terms (Socre$_2$). Score$_1$ consists of sum of the weights of matched terms. Each query term is tried to be matched with document terms in the order of lemma form, WordNet sense and stemmed form, and gets assigned the weight of the first matched term type. Passages are ranked by sum of their sentence scores.

$$Score_{sent} = Score_1 + Score_2 \qquad (1)$$

$$Score_1 = \sum_i wgt(qw_i) \qquad (2)$$

if qw_i appears in a stentence

$$Score_2 = \frac{\sum_{j=1}^{k-1} \frac{wgt(dw_j) + wgt(dw_{j+1})}{\alpha \times dist(j, j+1)^2}}{k-1} \qquad (3)$$

$$\times matched\_cnt$$

$wgt(qw_i)$: weight of query word i

$wgt(dw_j)$: weight of query word i,
with which document w ord j was matched

$dist(j, j+1)$: distance between
document w ord j and $j+1$

$matched\_cnt$: number of query word s
matched in a sentence

α : constant

Our system selected 1000 passages from 1000 retrieved documents per question.

2.3 Answer Processing

Answer processing selects answer candidates matching the answer type from each passage and ranks them. It uses stemmer[8], thesaurus (WordNet) [5], encyclopedia for its performance elevation. Answer processing is composed of four steps: Answer Matching, Pivot Detection, AAD Processing and Answer Ranking.

2.3.1 System Architecture

Figure 2 shows components of answer processing system. Answer matching (detection) finds answer candidates in POS-tagged passages selected by passage selection using the answer type determined by question processing. A query term, which shows up in various forms in the passage, is called "pivot". Answer ranking uses these pivots in scoring answer candidates. When the answer type of a question is "LANGUAGE_UNIT", AAD processing finds context-based answer candidates that are in abbreviated, appositive and definitive relation with the pivots. Answer ranking calculates the score of each answer candidate with various parameters, filters them according to the range and the type of answer, and finally sorts them.

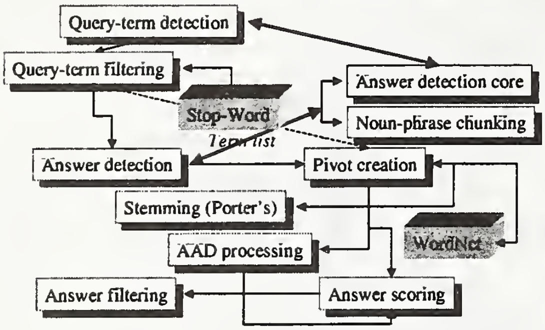


Figure 2 Answer processing

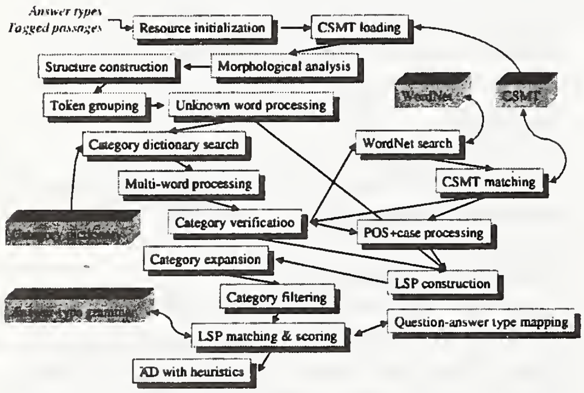


Figure 3 Answer matching

2.3.2 Answer Matching (detection)

Figure 3 shows the procedures of answer matching. Answer matching assigns semantic categories to each answer candidate by matching between LSP grammar and the normalized answer form from the following procedure. The procedure first searches semantic category dictionary. In case of its failure, it tries thesaurus matching between the sense-code from WordNet and the semantic categories in the CSMT (Category to Sense code Mapping Table), and then uses POS combined with lexicography.

- Searching semantic category dictionary
Semantic category dictionary has about 80,000 entries including single word and compound one. Each entry is assigned a semantic category among 65 ones which are components of LSP abstraction.

- Trying thesaurus matching
Sense code retrieved from WordNet [5] is mapped to each category among 65 semantic categories if it has a similarity greater than a threshold value.

- POS combined with lexicography
In case of failure of searching semantic category dictionary, POS combined with lexicography is used to build normalized form. If "Newton" has "np" (proper noun) POS tag, "Np" is used for normalization. It is because capitalization is important for detecting candidate answers, especially named entities.

When a normalized form matched with a LSP of the answer type, its terms are chosen as an answer candidate. The followings show some examples of LSP and its actual instances.

cd@unit_lengthcd@unit_length *length|1|4|4*
10 feet 5 inches
cd@unit_length%per@unit_time *speed|1|4|4*
3 km per hour

2.3.3 Pivot Detection

Pivots corresponding with query terms emerge in the passage in various way: full matching terms, partial matching ones for multi-words, stem matching ones for inflections and semantic synonyms using WordNet. When answer ranking scores answer candidates, pivots are weighted according to these normalized representations of query terms in a passage. When an answer candidate itself is a pivot, it is excluded from answer candidate set.

2.3.4 AAD Processing

In the case that no answer type can be

determined in question processing due to short of information ("LANGUAGE_UNIT" answer type), AAD processing finds context-based answer candidates that are in abbreviated, appositive and definitive relation with the pivots. It uses lexicographic patterns for abbreviation, and noun phrase chunking and clue words such as "so-called" and "stand for" for apposition and definition. The followings are examples of questions, of which answer type is LANGUAGE_UNIT.

Why does the moon turn orange?
What is epilepsy?
What imaginary line is halfway between the North and South Poles?
What is done with worn or outdated flags?

For more improvement of performance, AAD processing uses encyclopedia information extracted from WordNet glossary [5]. We gathered descriptions of about 110,000 words from WordNet glossary and removed stop words from the descriptions. Answer ranking reweighs each answer candidate through its semantic similarity with remaining terms in the descriptions.

2.3.5 Answer Ranking

Score of each answer candidate is mainly calculated by distance between pivots within some window in each selected passage. In addition to basic distance measure, the type and ratio matching each pivot with query terms, mean distance between pivots, and semantic type of answer candidate (especially in case of AAD processing) are all used for scoring each answer candidate:

$$Score_i = R_{pivot} \cdot (1 - \frac{dist_{avg.pivot}}{dist_{max.pivot}}) \cdot \frac{S_i}{AADfactor} \cdot \frac{1}{N_p} \cdot \sum_{j=1}^{N_p} r_j \cdot (1 - \frac{dist_j}{dist_{max}}) \quad (4)$$

R_{pivot}: ratio of matched pivots
$dist_{avg.pivot}$: average distance between pivots
$dist_{max.pivot}$: maximum of distance between pivots
S_i: intermediate score of *i*th Answer Candidate
AADfactor:
if question type is language-unit,
if NE type is AAD, 1
otherwise 4
otherwise 1
$Score_i$: final score of *i*th Answer Candidate
N_p: number of Pivots
r_j: weight factor of match type of *j*th Pivot
$dist_j$: distance between *j*th Pivot and *i*th Answer Candidate
$dist_{max}$: max value of $dist_j$

This formula (Eq. 4) reflects some of the following assumptions: (1) Reliable answer candidates would appear near query terms, so called pivots, in a passage. (2) Reliable answer candidates would show up around pivots which matched with query terms more exactly. (3) In the case of "LANGUAGE_UNIT" answer type, answer candidates extracted from AAD processing are more reliable than the others. (4) The smaller mean distance between pivots is, the more reliable an answer candidate around them would be. (5) If most of query terms appear in a passage, an answer candidate around their pivots is more reliable. (6) Finally, reliable answer candidates show up in some limited distance between pivots. After scoring all answer candidates, answer ranking filters less reliable answer candidates according to the range and type of the answer, sorts remaining answer candidates by their scores and presents *N* most reliable answer candidates.

2.4 Experiments in TREC-10

We participated in the main task of QA track. 500 questions were given to each participant to evaluate their QA systems. After all evaluation, it was known that 49 questions among them have no known correct answers in the document collection. Eight questions were excluded from the evaluation due to various problems with those questions.

Table 2 shows that, unlike the questions used in the previous QA tracks, questions like "what is X?" were remarkably increased. So, the task became more difficult since the answer types of such questions are often not specified definitely.

For each question, SiteQ used the top 1000 documents provided by NIST (PRISE search engine [7]), selected top 1000 passages from those documents, detected top five candidate answers from those passages and picked out 50-byte string including the candidate answer as an answer string. When the score of a candidate answer was lower than a threshold value or less than five candidates were detected, we added "NIL" string in the appropriate rank, which means that there might be no answer.

We submitted only one run (posqa10a) and it was evaluated by mean reciprocal rank (MRR) like the previous QA tracks [13]. The unsupported answers were judged incorrect in strict judgment but correct in lenient judgment. Table 1 shows the number of questions judged correct in each judgment and the mean reciprocal rank of 492 questions. Comparing with the average MRR of the 67 other runs submitted this year, our system located correct answers at rank 1 for relatively many questions. The difference between the strict and the lenient MRR arises because a word of the same answer type was added to 50-byte string when we picked out the answer string including a candidate answer.

Rank	# of Qs (strict)	# of Qs (lenient)	Avg. of 67 runs
1	121	124	88.58
2	45	49	28.24
3	24	29	20.46
4	15	16	12.57
5	11	14	12.46
No	276	260	329.7
MRR	0.320	0.335	0.234

Table 1 The number of questions judged correct and MRR

Q-type	**freq**	**MRR (strict)**	**MRR (lenient)**
how + adj/adv	31	0.316	0.332
how do	***2***	***0.250***	***0.250***
what do	***24***	***0.050***	***0.050***
what is	**242**	**0.308**	**0.320**
what/which noun	*88*	*0.289*	*0.331*
when	26	0.362	0.362
where	27	0.515	0.515
who	46	0.464	0.471
why	***4***	***0.125***	***0.125***
name a	2	0.500	0.500
Total	492		

Table 2 The frequency and MRR in each type of question

Table 2 shows the MRR for each type of question. For the questions like "What is X?", our system shows relatively good performance. This means that AAD processing was effective for those questions.

According to table 3, we know that the systems in TREC-10 show slightly higher performance than the systems in TREC-9. But this does not necessarily refer to the improvement of the systems.

	TREC-10 67runs	TREC-9 35runs
Avg. MRR	0.234	0.22
Median MRR	0.121	0.115
# of Qs with no answer(%)	67.01 %	68.54 %

Table 3 The comparison between TREC-10 and TREC-9

3. Web track: Systems and Experiences

This is our first participation in the Web track of TREC. Our system is based on POSNIR/K, Korean natural language information retrieval system [4]. For TREC-10, we focused on effectiveness in both noun phrase extraction and PRF (Pseudo Relevance Feedback). While query expansion using PRF turned out to contribute to the performance significantly, the noun phrases were used with single terms actually didn't contribute much.

3.1 Keyword Extraction

For keyword extraction, we tagged the document collection, wt10g, and queries using POSTAG/E, the English POS (Part-Of-Speech) tagger based on HMM. The output of POSTAG/E is composed of lexis, POS tag, and lemma. From the result of the tagger, we selected keywords using two-phase extraction. If the lemmas were registered in the dictionary, they were selected. On the other hand, lexes were stemmed by Porter's stemmer[8] and then the stemmed lexes were selected as keywords. Stop words were eliminated using two kinds of stop list: common stop list containing 569 words, and query-specific stop list containing 28 words which must be removed from the query.

For constructing noun phrases, we made lexico-syntactic rules based on the POS-tag patterns. Some of the rules are described below.

Term1/{*NN* | *NP*} Term2/{*NN* | *NP*}
→ Term1_Term2

Term1/{*NN* | *NP*} (*'s/POS* | *of/IN*) Term2/{*NN* | *NP*}
→ Term1_Term2
Term1/*JJ* Term2/{*NN* | *NP*} Term3/{*NN* | *NP*}
→ Term1_Term2_Term3

3.2 Initial Retrieval

Our retrieval system uses 2-poisson model based on the probabilistic term distribution. The system retrieves top-raked documents after giving scores to each document of a target data collection with each query term list made from the keyword extraction process. For scoring, a rank system uses Okapi BM25 formula [9] as shown below.

$$w^{(1)} = \log\left(\frac{N - n + 0.5}{n + 0.5}\right) \tag{5}$$

$$Score(d,q) = \sum_{t \in q}\left(\frac{(k_1 + 1) \times tf_t}{k_1 \times ((1-b) + b \times \frac{dl_d}{avdl}) + tf_t} \times w^{(1)} \times \frac{(k_3 + 1)tf_q(q,t)}{k_3 + tf_q(q,t)}\right) \tag{6}$$

, where N is the number of documents in the collection, n is the number of documents containing the term, tf_t is the term frequency of term t in a document d, dl_d is the document length, *avdl* is the average document length, $tf_q(q,t)$ is the term frequency of query term t in the query q, and k_1, b, k_3 are tunable constant parameters.

3.3 Query Expansion

Query expansion is achieved through PRF (Pseudo Relevance Feedback). In the process of PRF, top-ranked documents are regarded as relevant and TSV (Term Selection Value) is given to all single terms except stop words in them. Then, top-ranked single terms are expanded and added to the original query term list. In this process, the weights of both original and expanded query terms are reweighted by Eq.(7) reflecting relevance and non-relevance information [10].

$$w^{(1)} = \frac{k_5}{k_5 + \sqrt{R}}(k_4 + \log\frac{N}{N-n}) + \frac{\sqrt{R}}{k_5 + \sqrt{R}}(\log\frac{r+0.5}{R-r+0.5}) - (\frac{k_6}{k_6 + \sqrt{S}}\log\frac{n}{N-n}) - (\frac{\sqrt{S}}{k_6 + \sqrt{S}}\log\frac{s+0.5}{S-s+0.5}) \tag{7}$$

, where N, n is the same as in the Eq.(5), R is the number of documents known to be relevant to a specific topic, r is the number of relevant documents containing the term, S is the number of documents known to be non-relevant, s is the number of non-relevant documents containing the term, and k_5, k_6 are tunable constant parameters.

For TSV function, we developed and compared some TSV formulas adapting diverse TF (Term Frequency) factors.

$$TSV = \left(\sum_{d \in R} 0.5 + 0.5 \times \frac{tf_{t,d}}{dl_d}\right) \times w^{(1)} \tag{8}$$

$$TSV = \left(\sum_{d \in R} \log(tf_{t,d} \times \frac{avgdl}{dl_d}\right) \times w^{(1)} \tag{9}$$

$$TSV = \left(\sum_{d \in R} \frac{tf_{t,d}}{k_1((1-b) + b\frac{dl_d}{avgdl}) + tf_{t,d}}\right) \times w^{(1)} \tag{10}$$

, where $w^{(1)}$ is Eq. (7).

3.4 Final Retrieval

Final retrieval process is the same as the initial one except that, this time, each query term has the new weights given by Eq. (7) and the expanded query term list is used.

3.5 Experiments in TREC-10

Table 4 summarizes the TREC-10 results. The results indicate that when a query was expanded using PRF, the performance was better, but noun phrases didn't give much contribution to the performance. As for TSV function in using PRF, Eq. (10) which is adapting TF factor of the weight formula of Okapi was better than any others.

In order to further validate the results, the t-test was performed on the data (Table 5). The table shows the mean difference, the standard deviation difference, the t-statistics and the probability of average precision and recall-precision for no-PRF (baseline) versus PRF (using Eq. (10)) case. Though there are no significant differences for average precision in TREC-9 topics, the table shows the rest of the performance are all significantly improved when PRF was used.

	No query expansion			Query expansion			
	title only		title+desc	title only			
	no phrases	phrases	phrases	phrases			
		baseline		Eq. (7)	Eq. (8)	Eq. (9)	Eq. (10)
TREC-9							
Precision	0.1740	0.1747	0.2188	0.1758	0.1740	0.1781	0.1837
R-Precision	0.1962	0.1967	0.2399	0.1954	0.1940	0.2049	0.2082
TREC-10							
Precision	0.1535*	0.1521**	0.1877***	-	-	-	0.1771****
R-Precision	0.1853	0.1760	0.2240	-	-	-	0.2081

Table 4 Average precision & R-Precision for TREC topics (* : posnire01st **: posnire01pt *: posnire01ptd ****: posnire01rpt)**

		Mean difference	STD difference	T	Prob > |T|
TREC-9	Precision	0.0091	0.0393	1.6279	0.1100
	R-Precision	0.0116	0.0430	1.9071	0.0624
TREC-10	Precision	0.0356	0.0939	2.6841	0.0099
	R-Precision	0.0480	0.0836	4.0577	0.0002

Table 5 T-test: Avg. Precision & R-Precision - no-PRF (baseline) vs. PRF (Eq. (10))

4. Conclusion

In TREC-10, we participated in the QA track and the Web track.

We submitted a run for the main task of the QA track and it was judged and evaluated by the reciprocal rank. The MRR for 492 questions is 0.320 (strict), which is considerably higher than the average MRR of other 67 runs.

In the Web track, we confirmed that our new query expansion using PRF with TSV function adapting TF factor contributed to better performance.

5. References

[1] Lehnert, W. (1978) The process of question answering: A computer simulation of cognition. Lawrence Erlbaum Associates.

[2] Graesser, A. C., Lang, K., & Horgan, D. (1988) A taxonomy of question generation. Questioning Exchange, 2, 3-16.

[3] Harabagiu, S., Moldovan, D., at al. (2000) FALCON: Boosting knowledge for answer engines. In Proceedings of the 9th Text REtrieval Conference (TREC-9).

[4] Lee, S. and Cho, B. (2000) Statistical Natural Language Query System THE IR based on P-Norm Model; Korean TREC-1, In *Proceeding of The 5th Korea Science & Technology Infrastructure Workshop*, 189-202. (*in Korean*)

[5] Miller, G.A. (1995) WordNet: A lexical database, Communication of the ACM, vol 38: No11, pp 39-41.

[6] Moldovan, D., Harabagiu, S., at al. (1999) LASSO: A tool for surfing the answer net. In Proceedings of the 8th Text REtrieval Conference (TREC-8).

[7] NIST PRISE Search Engine: http://www.itl.nist.gov/div894/894.02/works/papers/zp2/main.html

[8] Porter, M.F. (1980) An algorithm for suffix stripping. Program, 14(3), pp 130-137.

[9] Robertson, S.E. et al. (1995) Okapi at TREC-3. In Overview of the Third Text Retrieval Conference (TREC-3). 109-126.

[10] Robertson, S.E and Walker, S. (1997) On relevance weights with little relevance information, In *Proceeding of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 16-24.

[11] TREC 2001 Question Answering Track Guidelines,

http://trec.nist.gov/act_part/guidelines/qa_track_spec.html

[12] Voorhees, Ellen M. and Dawn M. Tice, (2000) Building a question answering test collection. In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval

[13] Voorhees, Ellen M. and Dawn M. Tice, (1999) The TREC-8 question answering track evaluation. In Proceedings of the 8th Text REtrieval Conference (TREC-8).

TREC2001 Question-Answer, Web and Cross Language Experiments using PIRCS

K.L. Kwok, L. Grunfeld, N. Dinstl and M. Chan
Computer Science Department, Queens College, CUNY
Flushing, NY 11367

1 Introduction

We applied our PIRCS system for the Question-Answer, ad-hoc Web retrieval using the 10-GB collection, and the English-Arabic cross language tracks. These are described in Sections 2,3,4 respectively. We also attempted to complete the adaptive filtering experiments with our upgraded programs but found that we did not have sufficient time to do so.

2 Question-Answering (QA) Track

The QA Track requires obtaining 50-byte answer strings to 500 questions (later truncated to 492). The answers are to be retrieved from documents made up from the TREC collections: AP1-3, WSJ1-2, SJMN-3, FT-4, LA-5 and FBIS-5.

2.1 Approach

Our QA system is constructed using methods of classical IR, enhanced with simple heuristics. It does not have natural language understanding capabilities, but employs simple pattern matching and statistics. We view QA as a three-step process: 1) retrieving a set of documents that are highly related to the topic of the question; 2) weighing sentences in this document set that are most likely to answer the question according to the query type and its description; and 3) selecting words from the top-scoring sentences to form the answer string. This approach was quite successful for the 250-byte answer task at TREC-9 [1]. This year we added more heuristics, better pattern recognition and entity recognition.

2.2 Methodology

For the first step, retrieving a set of documents related to the question under focus, we employ both the NIST supplied document list as well as one generated by our PIRCS system. We also use a combination of these two lists that prove to be the best.

For the second step, weighting prospective sentences in the top ranked list of documents, we continue to employ the methods introduced in TREC-9, which are summarized below:

1) Coordinate Matching: counting words in common between the question and a document sentence.
2) Stemming: counting stems as opposed to words in 1). We use Porter's algorithm for stemming.
3) Synonyms: matching based on a manually created dictionary of common synonyms. Its size has increased to 420 terms from 300. It also contains unusual word forms, which are not handled well by stemming. Most of the entries were taken directly from Wordnet
4) RSV: use of the retrieval score of a document from PIRCS to resolve ties for sentences that have the same weight based on word or stem matching.
5) ICTF: use of Inverse Collection Term Frequency to give more credit to less frequently occurring words. For practical reasons, the collection used to obtain the frequencies is the N top retrieved documents.
6) Exact: giving extra credit for matching certain important words which must occur in the answer. At present, these are the superlatives: first, last, best, highest etc. However, one must be careful: 'best' is good but 'seventh best' is not.
7) Proximity: giving extra credit for query words in close proximity in a sentence. They are likely to refer to the same concept as the query. This is done only if all query content words are matched.
8) Heading: giving credit for query words in the headline tag even if they do not occur in a sentence.

9) Phrases: giving extra credit if consecutive words in the query occur in consecutive order in a sentence.
10) Caps: giving extra credit to matching of capitalized query words, assuming they are more important.
11) Quoted: giving extra credit to matching of quoted query words, assuming they are more important.

The query analyzer recognizes a number of specialized query types. 'Who', 'Where' and 'What name' queries are processed by the capitalized answer module, while 'When', 'How many', 'How much' and 'What number' are processed by the numerical answer module.

For 'Name' answers, heuristics were included to identify the following:
a) Persons: capitalized word not preceded by 'the'.
b) Places: capitalized words preceded by 'on', 'in', 'at'. Place names are also recognized by cue words such as 'located', 'next to', 'east of', neighboring, 'borders', etc.
c) Capitalized words: when no other clues are available.
d) Date entities, such as days, months and currency are screened out as incorrect answers.

For 'Numeric' answers, heuristics were included to identify the following:
a) Units: there are classes of queries, which require units. Our system recognizes common units of: length, area, time, speed, currency, temperature and population.
b) Date: there are some queries that have a date or year in the question. We require this date to occur in the sentence or within the Date Tag of a document.
c) Other entities are recognized such as time, address, telephone number, zip codes and percent.
d) Numbers: when no other clues are available.

Selecting a 50-byte answer from the top sentences is quite a challenge as the third step. We used the proximity to query words criterion in most cases, which misses many answers.

We also compiled several lists for countries, states, continents and oceans. We felt it may be useful for the list retrieval task.

2.3 Results and Discussions

Three runs named pir1Qqa{1,2,3} were submitted: pir1Qqa1 utilized the 50 top documents of the PRISE system; pir1Qqa2 used the top 400 subdocuments retrieved by our PIRCS system; pir1Qqa3 combines the two retrievals. PIRCS preprocesses the original documents and returns subdocuments of about 500 words long. Historically, tag information such as heading and (some) date were not captured in our system, which may result in some small degradation in the final score. Table 2.1 compares the submitted runs to the TREC overall median.

As shown in Table 2.1, our best entry pir1Qqa3 scored 0.326, 39% above the TREC median. It also demonstrates that combining retrievals is useful and improves over the results from individual retrievals pir1Qqa1 or pir1Qqa2. A new feature of TREC2001 is that a system might mark as NIL for a query that has no definite answer [2]. Since most correct answers occur at the top positions, a promising strategy is to mark all position 5 answers as NIL. We contemplated doing this but did not do so. The bottom 3 lines of the table show the improvement gained by this NIL strategy.

	All Queries	Compare to TREC	not NIL Queries	NIL Queries
TREC2001	0.234	+0%	0.239	0.193
Official:				
pir1Qqa1	0.300	+28%	0.333	0.000
pir1Qqa2	0.314	+34%	0.348	0.000
pir1Qqa3	0.326	+39%	0.362	0.000
NIL Strategy:				
pir1Qqa1	0.317	+36%	0.330	0.200
pir1Qqa2	0.328	+40%	0.342	0.200
pir1Qqa3	0.340	+45%	0.355	0.200

Table 2.1 QA Results: MRR Values and Comparison with Median

Pir1Qqa3 has 126 questions with rank 1 answers correct, 39 with rank 2, 22 rank 3, 14 rank 4, and 5 rank 5 correct. Since there are 49 questions for which the correct answer is NIL, the aggressive strategy of making every rank 2 answer NIL would do even better!

Question type	Number	Trec Med	pir1Qqa1	pir1Qqa2	pir1Qqa3	pirQqa3 compared to Trec
what	117	0.26	0.36	0.35	0.38	50%
what long	201	0.21	0.28	0.30	0.31	47%
stands for	4	0.42	0.88	0.63	0.75	77%
who	44	0.23	0.27	0.31	0.32	40%
who short	2	0.33	0.00	0.25	0.00	-100%
date	42	0.25	0.32	0.35	0.32	26%
where	26	0.24	0.27	0.24	0.25	7%
population	5	0.15	0.25	0.24	0.25	62%
why	4	0.25	0.25	0.33	0.21	-16%
what unit	29	0.24	0.21	0.24	0.26	7%
unknown	18	0.26	0.28	0.27	0.32	24%

Table 2.2 MRR Performance by question type.

Table 2.2 shows we did well for 'what' questions, both the definition and the longer types, and 'who' questions. The results are not as good for date ('when'), 'what unit' and 'where' type of questions.

The queries may be ranked by the overall performance by all the participants. It is instructive to look at some easy queries that we missed. It happens, that in many cases we retrieved the correct sentences but did not select the correct string. In many cases the correct answer is within the selected answer string, but the other words added (such us names and numbers) make the answer ambiguous.

2.4 Context and List Tasks

A week before the deadline we decided to try the context and list tracks by making minor changes. For the context track, we submitted two runs, pir1Qctx2 and pir1Qctx3. They are essentially the same as our main QA system. pir1Qctx1 (un-submitted) used the PRISE retrieval, pir1Qctx2 used PIRCS retrieval and pir1Qctx3 is a combination as before. The PIRCS retrieval is different in that it combines the series of questions into one query, aiming to retrieve documents that have all or many of the words in the series.

	MRR Score	Compare to TREC Med
TREC2001 average	0.298	0%
pir1Qctx1 (unofficial)	0.310	+4%
pir1Qctx2	0.314	+5%
pir1Qctx3	0.329	+10%

Table 2.3: Context Task Results

Considering all questions to be independent and evaluate as in main QA, we get the results shown in Fig.2.3. It seems retrieving on all query words for pirQctx2 did not substantially improve the results. Combination of retrievals again proved its usefulness as pir1Qctx3 outperformed its individual retrievals. The context task is an interesting and important task and more intelligence must be crafted into a system to take advantage of the knowledge gained from a succession of previous questions (which we did not do).

We made two changes in the QA system with an eye towards improving performance in the list task. We added a list of countries, states and oceans, and we improved our duplicate answer detection, so that similar forms will be considered equivalent and suppressed. We submitted two runs, pir1Qli1 based on PRISE retrieval and pir1Qli2 based on PIRCS retrieval. There was a bug in the second run output routine that truncated all results to the first word.

	> med	= med	< med
pir1Qli1	14[2]	10(1)	1
pir1Qli2	7[1]	11(6)	7(7)

Table 2.4: List Task:. Comparison with Median

Table 2.4 shows the performance of the submitted runs compared with the median of all runs. The un-bracketed values are the actual number better, worse or same as the median; the numbers in square brackets denote best, and the numbers in parenthesis denote worst scores.

3 Web Track

The target collection for the Web track is the W10g disks used last year. We submitted three runs: two for title only queries pir1Wt1 and pir1Wt2, and one for all-section query pir1Wa, which is a long query. Last year [1], we noticed that several queries returned no documents because the query words are common words and screened out by our Zipf threshold. Returning a random set of documents usually is fruitless. This year, for these 'zero' queries, we did special processing to bring back words that were screened out due to high frequency, hoping that we might restore some precision value. Documents having these terms within a distance of 5 words in a sentence are considered. For ranking, the minimum distance and the number of such repeats are used, and no second stage retrieval was performed on these queries. This year, there were only 3 such queries (509, 518, 521), but the process was unsuccessful. This is pir1Wt2. For pir1Wt1, we additionally do this process for queries left with one term below threshold. This turns out to depress effectiveness rather than help. Also, we had no spell-check nor punctuation processing, so that queries like #509 ("steroids;what does it do to your body") was not corrected. Query #531 ("Who and whom") contains all stop words and also returns zero precision. Results of our runs are tabulated in Table 3.1 and 2.

← Query Type →

	Title: pir1Wt1	Title: pir1Wt2	All sections: pir1Wa
Relv.Ret (at most)	2263 0 (3363)	2275 2 (3363)	2284 6 (3363)
Avg.Prec	.1660 0	.1742 5	.1715 12
P@10	.2220 0	.2160 3	.2780 11
P@20	.2070 0	.2110 4	.2370 15
P@30	.2013 0	.2040 8	.2220 18
R.Prec	.1700 0	.1894 5	.1968 9

Table 3.1: Automatic Web Results for 50 Queries

Query Type

	Title: pir1Wt1	Title: pir1Wt2	All sections: pir1Wa
	> = <	> = <	> = <
Avg.Prec	24,4 1 25,5	25,4 1 24,5	13,3 2 26,7
prec at 10	13,3 17 20,13	14,4 18 18,14	10,2 17 23,13
prec at 20	22,5 10 17,10	23,6 12 15,11	14,4 10 26,14
prec at 30	21,5 10 19,9	23,7 9 18,10	16,5 10 21,11

Table 3.2: Web Results - Comparison with Median

The result for pir1Wt2 is about median. Using all sections of a query pir1Wa does not perform better – we suspect there may be some parameters set wrong in our processing. With respect to high precision, Table 3.2, it appears our system perform better at precision 20 & 30 compared to median.

4 Cross Language Track

For Arabic utf-8 coding, the most prevalent two-byte coding is similar to Chinese GB. We think that our Chinese processing can support Arabic with few changes. A student who knows Arabic expressed interest to help us in forming a stopword list and try to find stemming algorithms from the web. A number of such programs were examined, and we eventually discovered that none can process large volumes in reasonable time without drastic re-programming. We also tried to locate an Arabic-English dictionary without success. However, the website for English to Arabic translation (http://tarjin.ajeeb.com) seems useful and good. We had the given English queries translated by using this site. To meet the deadline, we finally decided to use a mixture of n-grams for indexing so that we do not have to rely on linguistic processing. Our representation is to mix 4-gram, 5-gram and single words without stemming or stopword removal.

We submitted four runs two for monolingual Arabic: pirXAtdn and pirXAtd using all sections, and title with description section respectively. The corresponding runs for English-Arabic cross language runs are: pirXEtdn and pirXEtd. Results are tabulated in Table 4.1.

Query Type

	Mono tdn	Cross tdn	Mono td	Cross Td
Relv.Ret (at most)	1254 (4122)	899 (4122)	974 (4122)	802 (4122)
Avg.Prec	.1036	.0440	.0852	.0360
P@10	.2440	.1280	.1720	.1040
P@20	.2120	.1220	.1540	.0920
P@30	.2000	.1200	.1520	.0867
R.Prec	.1602	.0768	.1405	.0647

Table 4.1: Automatic Mono andCross Language Results for 25 Queries

The results are way below median. Apparently, there was an error in the retrieval in that no year 2000 documents were returned in our retrieval list. We corrected the error but result still does not materially change. It also seems that we may have some system problem related to LINUX v7 where we ran this experiment. We did not pursue this cross language track further.

5 Conclusion

We continued experimenting with our QA system based on classical IR methods enhanced with simple heuristics for locating good sentences. It achieved above average results. This year we used better pattern and entity recognition. In the future, more heuristics, increased use of knowledge bases, exploring part-of-speech information and more careful query analysis will be needed for further progress. The context and list tasks were also prepared using the same methodology. They also give respectable average. It may be because the average is low, or it may perhaps show that an IR-based system is quite robust although it may be less intelligent.

Our web and cross language results are not up to expectation. For the web track, we did not employ more advanced processing such as collection enrichment, term variety, etc. because of time constraints. This year we transferred these two tasks to work on a Linux-PC platform instead of Solaris-SUN. It is possible that some system error may creep in during processing of the Arabic coding.

Acknowledgments

This work was partly supported by the Space and Naval Warfare Systems Center San Diego, under grant No. N66001-1-8912. We like to thank Khalid Yehia and Peter Deng for helping us in the Arabic processing.

References

[1] Kwok, K.L., Grunfeld, L., Dinstl, N. & Chan, M. TREC-9 cross language, web and question-answering track experiments using PIRCS. In: The Ninth Text Retrieval Conference (TREC-9). NIST SP 500-249, pp.417-426, 2001.

[2] Voorhees, E. Overview of the TREC-9 Question-Answering Track. In: The Ninth Text Retrieval Conference (TREC-9). NIST SP 500-249, pp.71-79, 2001.

RICOH at TREC-10 : Web Track Ad-hoc Task

Hideo Itoh, Hiroko Mano and Yasushi Ogawa
Software Research Center, RICOH Co., Ltd.
1-1-17 Koishikawa, Bunkyo-ku, Tokyo 112-0002, JAPAN
{hideo,mano,yogawa}@src.ricoh.co.jp

1 Introduction

This year we participated in the Web track and submitted four title-only runs {`ricMM`, `ricAP`, `ricMS`, `ricST`} which were automatically produced for ad-hoc task. This is our third participation in TREC. Last year in the TREC-9 main web track, our system achieved the best performance in automatic results. However the following problems could be pointed out at the same time.

- Our system uses many parameters in term-weighting and document-scoring. The value of each parameter should be tuned to improve retrieval effectiveness using test collections. However we cannot use enough relevance information in most of practical situations.
- Automatic query expansion using pseudo-relevance feedback occasionally produces a negative effect. For example in TREC-9, the performance for title-only query was hurt by query expansion using pseudo-relevance feedback.

In TREC-10, we tackled the above problems in addition to taking account of practical retrieval efficiency.

2 System description

Before describing our approach, we give the system description as background. For the TREC-10 experiments, we revised query processing although the framework is the same as that of TREC-9 [4]. The basic features of the system are as follows :

- Effective document ranking based on Okapi's approach [10] with some modifications.
- Scalable and efficient indexing and search based on the inverted file system which can be used for both Japanese and English text [6]
- Originally developed English tokenizer and stemmer for word indexing and query processing.

In what follows, we describe the full automatic process of document ranking retrieval for the TREC-10 web track ad-hoc task.

2.1 Query term extraction

We used only the "title" field of each topic. Input topic string is transformed into a sequence of stemmed tokens using the English tokenizer and stemmer. Query terms are extracted by matching some patterns against the sequence. We can flexibly control term extraction using the patterns which are described in regular expression on each token's string or tag assigned by the tokenizer. The tag indicates the character type (alphabetical, numeral, and so on) of the token string. Stop words are eliminated using the Fox's [3] word list.

For initial retrieval, both "single term" and "phrasal term" are used. A phrasal term consists of two adjacent terms and represented by a proximity operator to match only adjacent occurrences in target documents.

In order to moderate the effect of the phrasal term, a "combination value" is assigned to each phrasal term and multiplied by the weight in the process of term weighting. The value is estimated by ψ/b, where ψ is a tuning parameter and b is the number of extracted phrasal terms.

2.2 Initial retrieval

Each query term is submitted one by one to the ranking search system, which assigns a weight to the term and scores documents including it. Retrieved documents are merged and sorted on the score in descent order. The specification of term weighting and document scoring is explained in the section 3.1.

2.3 Seed document selection

Top ranked documents are assumed to be pseudo-relevant to the topic and selected as a "seed" of query expansion. The maximum number of seed documents is ten.

To avoid duplication in the seed documents, the document which has the same length and score as that of the document previously selected is skipped [4].

We also skipped any document the length of which exceeds the average in the collection, since the lengthy document may include many heterogeneous topics and may not be effective as a seed.

2.4 Query expansion

Candidates of expansion terms are extracted from the seed documents by pattern matching as in query term extraction mentioned above. Each candidate is pooled with its seed document frequency and term frequency in each seed document.

We do not use phrasal terms for query expansion because phrasal terms may be less effective to improve recall and risky in case of pseudo-relevance feedback.

The weight of initial query term is re-calculated with the Robertson/Spark-Jones formula [7] if the term is found in the candidate pool.

Before selecting expansion terms among the candidates, we assign to each candidate the Robertson's Selection Value [8] and Robertson/Spark-Jones relevance weight. In this step, candidates the document frequency of which is less than one thousand are eliminated because these term may be less effective for query expansion.

We did not mix the weight based on the seed documents with the prior weight based on the document collection as in [10][4]. Instead, the rank of a document is mixed between two document rankings in the data fusion manner (see the section 4).

The candidates are ranked on the RSV and top-ranked terms are selected as expansion terms. In consideration of practical retrieval time, we used only ten expansion terms.

As in the case of phrasal query terms, a "combination value" is assigned to each expansion term. The value is estimated by ξ/e, where ξ is a tuning parameter and e is the number of expansion terms.

2.5 Final retrieval

Each query and expansion term is submitted one by one to the ranking search system as in initial retrieval. Since the submitted term is already weighted in the previous phase, the system simply multiplies the weight by the document score to give a final score to the document including the term.

3 Automatic parameter estimation

Many parameters have been used in our retrieval process and the system performance heavily depends on whether the set of the values fits with the target query and document collection. Therefore parameter tuning is inevitable to maintain retrieval effectiveness. However in most practical settings such as in commercial use, we cannot use enough information for the parameter tuning. In the TREC-10 experiments, we tried to automatically adapt some parameter values to any given query and document collection.

3.1 Term weighting

The ranking system uses the following term-weighting formula (1).

$$w_t = \log\left(k_4' \cdot \frac{N}{n_t} + 1\right), \tag{1}$$

where w_t is the weight of the term t, N is the number of documents in the collection, n_t is the number of the documents in which t occurs and k_4' is the parameter of the formula.

The formula (1) is based on the Robertson/Spark-Jones formula (2).

$$w_t = \log \frac{p_t\ (1-q_t)}{q_t\ (1-p_t)} \tag{2}$$

where $p_t = \mathrm{P}(t \text{ occurs} \mid \text{relevant document})$ and $q_t = \mathrm{P}(t \text{ occurs} \mid \text{non-relevant document})$. While q_t is estimated by n_t/N as usual, p_t is estimated in our formula (1) as follows:

$$p_t = p0 + (1-p0)q_t \tag{3}$$

where $p0$ is an unknown constant called a "base probability" in this report [1]. Using the formulas (1), (2) and (3), we get the relation between k_4' and $p0$ as :

$$k_4' = \frac{p0}{1-p0} \tag{4}$$

Instead of conventional parameter k_4', we adapt the base probability $p0$ to a given query using following the heuristics :

> The average of p_t may monotonically decrease as the number of query terms u increases. Especially, if u equals 1 then the p_t nearly equals to 1.

To implement this heuristics, we give the following estimation :

$$p0_i = \frac{\rho}{u_i} \tag{5}$$

where $p0_i$ is a base probability for topic i, u_i is the number of single terms in topic i and ρ is a constant the value of which nearly equals to 1 (the actual value is 0.9). As a result, we estimate the weight for term t in topic i with the following formula (6)

$$w_{t,i} = \log(\frac{\rho}{\rho - u_i} \cdot \frac{N}{n_t} + 1) \tag{6}$$

Fig.1 shows the relation between the average of p_t and the number of query terms in test collections of the TREC-7, TREC-8 and TREC-9 ad-hoc retrieval task. For each "desc" field which includes n terms, we got the average of p_t of the term set using relevance data. We think the data has supported the heuristics mentioned above because in most desc fields (about 87%), the number of query terms is less than eight.

3.2 Document scoring

We also tried to adapt parameters used for document scoring. Because the effect was negative in the result of the official run (`ricST`), we describe the method briefly.

[1] If $0 \leq p0 \leq 1$ then the weight never gets negative. It enables us to treat term-weights simply and consistently in any case [4].

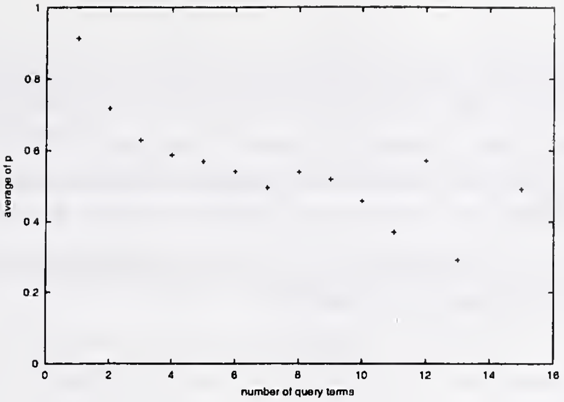


Figure 1: Average of p and number of query terms

In the process of conventional document ranking, each document d is given the score $s_{d,t}$ for the term t with the following formula (7) :

$$s_{d,t} = \frac{f_{d,t}}{f_{d,t} + \kappa\ ((1-\lambda) + \lambda\frac{l_d}{L})} \cdot w_t \tag{7}$$

where $f_{d,t}$ is within-document frequency of t in d, l_d is the length of d, L is the average length of documents. κ and λ are tuning parameters for document scoring and the actual values used in the official runs (`ricMM, ricMS, ricAP`) are 0.5 and 0.2 respectively.

For a query term t, the value of λ is estimated using the median m_t and the variance v_t of the lengths of retrieved documents as follows :

$$\lambda_t = \frac{m_t}{m_t + v_t} \tag{8}$$

In addition to the above estimation, we used m_t instead of L in the formula (7).

The value of κ is fixed (the actual value is 1) and we applied the following non-linear score transformation :

$$newscore(d,t) = \frac{w_t}{max^\mu - min^\mu}(score(d,t)^\mu - min^\mu) + \epsilon \tag{9}$$

where ϵ is a constant with a very small value. The value of μ is given by

$$\mu = \frac{min + ave}{2} \tag{10}$$

where max, ave and min are those of document scores given by the formula (7). Using the transformation and fixing the value of κ, we tried to normalize the document score distribution.

4 Rank merging

Automatic query expansion using pseudo-relevance feedback occasionally produces a negative effect [12][2]. In Fig.2 we draw a comparison of the average precision between initial retrieval (unexpanded) and final retrieval (expanded) on the TREC-9 title-only ad-hoc retrieval data. Each dot represents one of the 50 topics. If the dot is above (below) the bisecting line, then the performance is improved (hurt) by query expansion.

In case of a short query, the information need is often under-specified and the effect of pseudo-relevance feedback becomes very unstable.

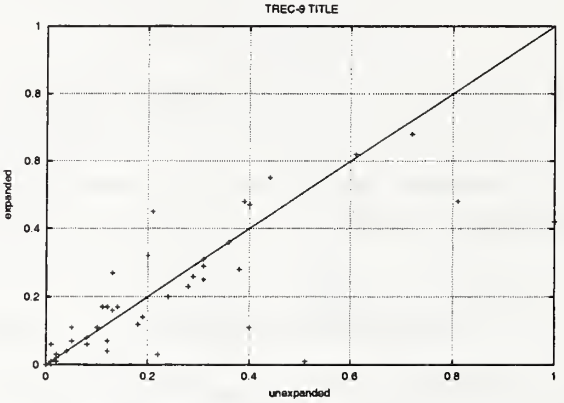


Figure 2: Inconsistency of query expansion

To remedy the inconsistent effect of pseudo-relevance feedback, we construct a final ranking by merging first ranking with query terms and second ranking with both query and expansion terms. We call the method "rank merging" in this report. This is a kind of the data fusion problem [1][11].

Specifically we give scores in a final ranking by the following formula :

$$score(d) = \frac{1}{(1-\beta)\cdot first\_rank(d) + \beta \cdot second\_rank(d)} \tag{11}$$

where β is a tuning parameter with the actual value 0.6. $first\_rank(d)$ ($second\_rank$) is the rank of the document d in first (second) ranking. The target of this merging is restricted to only documents the rank of which in first ranking is smaller than a threshold $maxRank$. The actual value of the threshold is 20.

5 Results

Results of our submitted runs are summarized in Table 1. These runs were automatically produced using only title field. No link information in a HTML document was used. Query

expansion by pseudo-relevance feedback was applied to all runs.
Three results {ricMS, ricMM, ricAP} are in the the third-ranked group among all of automatic title-only official runs. In our analysis of the results, however, we got the following findings.

- Skipping duplications in seed documents (section 2.3) produced a slightly positive effect.
- Skipping lengthy seed documents (section 2.3) clearly produced a negative effect.
- Eliminating an expansion term the document frequency of which is small (section 2.4) slightly hurt the performance.
- The loss expected from dropping the collection-wide weight (section 2.4) could not be compensated for by the rank merging.

We found that if the problematic procedures mentioned above had not been taken, the average precision in the almost same setting as ricMS would have been 0.2247.

Rank merging slightly improved the average precision in comparison with the baseline. Since the threshold *maxRank* is 20, the rank merging influences only the top part of the ranking. However it improved P@10 which is crucial to ad-hoc retrieval users.

The automatic estimation of parameter ρ exerted a little but positive influence on retrieval performance. We think this is due to the narrow range of the number of query terms in a title topic. RET100 of this run is the best one among all official runs. This partially comes from the effectiveness of our stemmer and the scalability of the system which enable us to index all documents in WT10G.

The score transformation clearly hurt the retrieval performance. Since parameters κ and *lambda* influence on retrieval performance more than the other parameters, automatic tuning of these parameters is very attractive. After the submission, we continued to develop a more effective and theoretically grounded method for the automatic tuning, taking account of Robertson's approximations to the two-Poisson model [9]. However, it has still been under way.

RUN	AveP	P@10	RET100	Experiment (what was different from baseline)
ricMS	0.2068	0.3360	16.84	baseline
ricMM	0.2084	0.3420	16.84	rank merging
ricAP	0.2077	0.3380	17.62	automatic estimation of parameter ρ
ricST	0.1933	0.3260	16.20	non-linear score transformation

Table 1: Results of Web track ad-hoc task official runs

References

[1] N. Belkin et al., "Combining the evidence of multiple query representations for information retrieval", Information Processing and Maragement, 31, 3, pp.431–448, 1995.

[2] C. Carpineto et. al., "An information-theoretic approach to automatic query Expansion", ACM Transactions on Information Systems, 19, 1, pp.1–27, 2001

[3] C. Fox, "A stop list for general text", *ACM SIGIR Forum*, 24, 2, pp. 19–35, 1991.

[4] Y. Ogawa, H. Mano, M. Narita, and S. Honma, "Structuring and expanding queries in the probabilistic model", In *The Eighth Text REtrieval Conference (TREC-9)*, pp.427–435, 2001.

[5] Y. Ogawa, H. Mano, M. Narita, and S. Honma, "Structuring and expanding queries in the probabilistic model", In *The Eighth Text REtrieval Conference (TREC-8)*, pp.541–548, 2000.

[6] Y. Ogawa, T. Matsuda, "An efficient document retrieval method using n-gram indexing" (in Japanese), *Transactions of IEICE*, J82-D-I, 1, pp. 121–129, 1999.

[7] S. E. Robertson, K. Spark-Jones, "Relevance weighting of search terms", Journal of ASIS, 27, pp.129-146, 1976

[8] S. E. Robertson, "On term selection for query Expansion", Journal of Documentation, 46, 4, pp.359-364, 1990

[9] S. E. Robertson, S. Walker, "Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval", In *Proc. of 17 th ACM SIGIR Conf.*, pp.232–241, 1994.

[10] S. E. Robertson, S. Walker, "On relevance weights with little relevance information", In *Proc. of 20th ACM SIGIR Conf.*, pp.16–24, 1997.

[11] H. Schutze, J. O. Pedersen, "A cooccurrence-based thesaurus and two applications to information retrieval", Information Processing and Management, 33, 3, pp. 307-318, 1997

[12] J. Xu, W. B. Croft, "Improving the effectiveness of information retrieval with local context analysis", ACM Transactions on Information Systems, 18, 1, pp.79–112, 2000

Rutgers' TREC 2001 Interactive Track Experience

N.J. Belkin, C. Cool*, J. Jeng, A. Keller, D. Kelly, J. Kim, H.-J. Lee, M.-C. Tang, X.-J. Yuan
School of Communication, Information & Library Studies
Rutgers University
New Brunswick, NJ 08901-1071
*GSLIS, Queens College, CUNY
nick@belkin.Rutgers.edu ccool@qc.edu [judyjeng | amkeller | diane | jaykim | hyukjinl | muhchyun | xjyuan]@scils.rutgers.edu

Abstract

Our focus this year was to investigate methods for increasing query length in interactive information searching in the Web context, and to see if these methods led to changes in task performance and/or interaction. Thirty-four subjects each searched on four of the Interactive Track topics, in one of two conditions: a "box" query input mode; and a "line" query input mode. One-half of the subjects were instructed to enter their queries as complete sentences or questions; the other half as lists of words or phrases. Results are that: queries entered as questions or statements were significantly longer than those entered as words or phrases (twice as long); that there was no difference in query length between the box and line modes (except for medical topics, where keyword mode led to significantly more unique terms per search); and, that longer queries led to better performance. Other results of note are that satisfaction with the search was negatively correlated with length of time searching and other measures of interaction effort, and that the "buying" topics were significantly more difficult than the other three types.

1 Introduction

The goal of the TREC 2001 Interactive Track was that the participants in the Track carry out exploratory studies which could lead to testable hypotheses (or firm research questions) to be investigated in the course of the TREC 2002 Interactive Track. These exploratory studies were to be carried out by having subjects search on a variety of predetermined topics on the "live" Web. At Rutgers, we decided to focus primarily on the issue of query length in interactive searching, with secondary interests in subject use of a feedback device, and in the effect of highlighting of query terms in search results.

We were interested in query length for three reasons. The first of these is the well-known finding that, for best-match retrieval engines, the longer the query, the better the retrieval results. Since it is also well-known that users of Web search engines typically enter rather short queries, we were interested in methods that might increase query length. The second reason is that our work was also connected with the NSF-funded MONGREL project in which we are collaborating with colleagues at the University of Massachusetts, Amherst (MONGREL). This project is concerned with using language-modeling methods (e.g. Ponte & Croft, 1998) for developing topic and user models; for this purpose, it is important to have fairly long queries. The third reason for considering query length was that the Interactive Track topics were designed to be of four different "types": medical; travel; buying; and project, and that the topics associated with these types were often couched (or could be couched) as questions. We hypothesized that differences between these types might show up in either length of query for each, or in framing of question for each. In either case, longer queries should enhance the chances of discovering any such differences.

We considered two different methods for increasing query length. The first was to vary the size and format of the query input mode. Karlgren & Franzén (1997) found that subjects who were asked to input queries in a box-like query input window (one in which the input query was wrapped for multiple lines) had significantly longer queries than subjects who entered queries in a standard Web-browser query input line. We decided to test this result in our study, which had more subjects than they did, and which also had a greater variety of search topic types. Our hypothesis was that the box mode would lead to longer queries than the line mode, for two reasons. The first is that the perceived space for query entry is larger in the box mode; the second is that the entire query, no matter how long (within some reasonable limits) would be visible in the box mode, and therefore people would be encouraged to continue query entry. The second method of increasing query length was to vary the form of query. We did this by instructing subjects either to enter their queries as complete questions or sentences, or to enter their queries as a list of words or phrases. Our hypothesis was that the former would lead to longer queries than the latter.

We were also secondarily interested in studying use of feedback facilities in Web searching, following up on our previous TREC Interactive Track studies (cf. Belkin, Cool, et al. 2001). This was implemented in our system this time as a "copy-and-paste" facility for moving text from displayed pages directly into the query. Finally, we decided to consider the perceived usefulness of highlighting of query terms.

2 System

Searching was conducted through a proxy server and our own interface, using the Netscape browser, to the Excite search engine. Our initial interface consisted of a query input window, which was either a standard 50-character line, or a scrollable 40-character by five line box, in which input text was automatically wrapped, and a "search" button. The query was displayed at the top and bottom of each retrieved Web page (result or linked), along with a query modification window, into which text from the page could be copied, and then copied into the query and run as a modified query. All query terms were highlighted in query result lists and in viewed pages. All displayed, visited and printed pages were logged, as were all queries and query modifications. Screen shots of the interface are available at http://www.scils.rutgers.edu/mongrel/trec.html.

3 Methods

3.1 Design

The Interactive Track specification provided sixteen search topics, four topics for each of four different topic "types": medical; travel; buying; project. Within each type, there were two "fully-specified" topics, and two "partially-specified" topics. Our study was designed with one within-subjects factor (line vs. box query input mode), and one between-subjects factor (complete question/sentence vs. list of words/phrases). In order to obtain adequate representation on all topic types and on the specified-partially specified dimension, we needed to have 32 subjects (in fact, we ran 34 subjects, duplicating the first two subject conditions), sixteen in the group instructed to search using a complete question/sentence; sixteen in the group instructed to search using a list of words or phrases. Each of the subjects searched on four topics, the first two fully-specified, the second two partially-specified. This order was determined on the basis that it would be easier for the subjects to do the fully-specified topics. Each subject performed one specified and one partially-specified search using the box input mode, and one specified and one partially-specified search using the line input mode. Search time was limited to a maximum of fifteen minutes. The query input modes were alternated, and the order in which they were performed was systematically varied for the entire group of subjects. The design of the study is shown in Table 1, where **Snn** is the subject number, column one defines the combination of type of query and order of input mode, and each cell represents the topics and the order in which they were searched by each subject. The order in which subjects were run is indicated by highlighting in Table 1, with the diagonal pattern continued, first recurrence beginning with S03.

Condition	**S01**	**S02**	**S03**	**S04**	**S05**	**S06**	**S07**	**S08**
Q	Medical 1	Medical 2	Buying 3	Buying 4	Travel 5	Travel 6	Project 7	Project 8
Order	Buying 3	Travel 6	Travel 5	Project 8	Project 7	Medical 2	Medical 1	Buying 4
LB	Project 15	Project 16	Medical 9	Travel 14	Buying 12	Buying 11	Travel 13	Medical 10
	Travel 14	Buying 11	Project 15	Medical 10	Medical 9	Project 16	Buying 12	Travel 13
Condition	**S09**	**S10**	**S11**	**S12**	**S13**	**S14**	**S15**	**S16**
Q	Medical 1	Medical 2	Buying 3	Buying 4	Travel 5	Travel 6	Project 7	Project 8
Order	Buying 3	Travel 6	Travel 5	Project 8	Project 7	Medical 2	Medical 1	Buying 4
BL	Project 15	Project 16	Medical 9	Travel 14	Buying 12	Buying 11	Travel 13	Medical 10
	Travel 14	Buying 11	Project 15	Medical 10	Medical 9	Project 16	Buying 12	Travel 13
Condition	**S17**	**S18**	**S19**	**S20**	**S21**	**S22**	**S23**	**S24**
T	Medical 1	Medical 2	Buying 3	Buying 4	Travel 5	Travel 6	Project 7	Project 8
Order	Buying 3	Travel 6	Travel 5	Project 8	Project 7	Medical 2	Medical 1	Buying 4
LB	Project 15	Project 16	Medical 9	Travel 14	Buying 12	Buying 11	Travel 13	Medical 10
	Travel 14	Buying 11	Project 15	Medical 10	Medical 9	Project 16	Buying 12	Travel 13
Condition	**S25**	**S26**	**S27**	**S28**	**S29**	**S30**	**S31**	**S32**
T	Medical 1	Medical 2	Buying 3	Buying 4	Travel 5	Travel 6	Project 7	Project 8
Order	Buying 3	Travel 6	Travel 5	Project 8	Project 7	Medical 2	Medical 1	Buying 4
BL	Project 15	Project 16	Medical 9	Travel 14	Buying 12	Buying 11	Travel 13	Medical 10
	Travel 14	Buying 11	Project 15	Medical 10	Medical 9	Project 16	Buying 12	Travel 13

Table 1. Subject assignment form. Q = question/sentence T = word/phrase L = line input B = box input. Specified topics are numbers 1-8; partially specified topics are numbers 9-16.

3.2 *Procedure*

Volunteer subjects were recruited primarily from the population of students at the School of Communication, Information and Library Studies (SCILS) at Rutgers University. The recruitment notice specified that the single session for which they were volunteering would last about two hours. The search sessions were held at the Information Interaction Laboratory at SCILS, which allows unobtrusive video and audio recording of searching behavior. Upon arrival, subjects completed first an Informed Consent form, and then a brief demographic questionnaire eliciting age, gender, educational background, and a variety of measures of previous searching experience and searching attitudes. They were then given a general description of the tasks that they would be asked to perform during the experimental session. Then they were handed a specification of the first search topic, on a form which asked them to indicate whether they knew the answer to the search topic, or where to find an answer, and their confidence in that judgment. Then they went to the search station, and began their search on the first topic. Subjects were instructed to "think aloud" as they searched, and their thinking aloud, as well as the monitor while searching were recorded on videotape. Subjects were instructed to print out all pages which helped them to answer the search topic. They were told that they could search for up to fifteen minutes, but could quit searching as soon as they felt they were done. On completion of the search, they answered a brief questionnaire about that search experience, and then explained to the experimenter present why they printed out each page that they did (i.e., what it was about that page that helped them to answer the search question/topic). This procedure was continued for all four search topics. After the fourth topic cycle, subjects were administered an exit interview, which was recorded on audio tape, eliciting their opinions about the different query input modes, about the query type that they were asked to use, about the query modification and highlighting features, and about the general characteristics of the systems that they used, as compared to those they ordinarily use. Examples of the data collection instruments are available at http://www.scils.rutgers.edu/mongrel/trec.html

3.3 *Subjects*

The subjects for this study were primarily students in the Masters of Library and Information Science program at SCILS, but also included some undergraduate students in communication courses. Of the 34 subjects, 5 were male, 29 female. The age distribution 44% between 20 and 29 years, 30% between 30 and 39 years, 12% between 40 and 49 years, and 14% over 50 years.

4 Results

4.1 *Query and interaction characteristics*

Queries were characterized according to the following measures: number of queries per search; average query length (in words) per search; number of unique query terms per search. Interaction was characterized according to the number of unique pages seen (i.e. urls displayed) and the number of unique pages viewed (i.e. opened by following a link). The data for these measures, for all searches, are displayed in Table 2.

Descriptive Statistics

	N	Minimum	Maximum	Mean	Std. Deviation
unique seen	133	10	83	23.28	16.21
unique viewed	133	0	16	3.65	2.61
number of queries	134	1	8	2.13	1.70
number of unique terms in search	133	1	28	7.23	4.72
AVLENGTH	134	1	17	5.54	3.29
Valid N (listwise)	132				

Table 2. Query and interaction measures for all searches

When the data were analyzed to see if the query type (i.e. question/sentence vs. list of words/phrases) affected query characteristics or interaction characteristics, we found that the two measures of query length, number of unique terms in the search, and average query length, were significantly greater for the question/sentence type, using the t test (unique terms in search, t (131) = 9.14, p<.01; average query length, t(132) = 11.94, p<.01). The data for all searches on all topics are displayed in Table 3. This relationship held for all different topic types, when analyzed separately, and also for both specified and unspecified queries (with the exception of the medical topics, see below).

Group Statistics

	condition	N	Mean	Std. Deviation	Std. Error Mean
unique seen	0	71	23.31	16.04	1.90
	1	62	23.24	16.53	2.10
unique viewed	0	71	3.70	2.28	.27
	1	62	3.58	2.97	.38
number of queries	0	71	2.30	1.84	.22
	1	63	1.94	1.51	.19
number of unique terms in search	0	71	9.97	4.48	.53
	1	62	4.10	2.53	.32
AVLENGTH	0	71	7.76	2.90	.34
	1	63	3.02	1.31	.17

Table 3. Query and interaction characteristics for question/sentence (0) and list of words/phrases (1) query types.

There was no significant difference in query length, or any other query or interaction characteristic between the box and line query input mode when all searches for all topics are considered. However, when each type of topic was considered separately, an interesting difference became apparent. For the medical topic type, the number of unique terms per search, and the number of queries per search, were significantly greater in the line mode than in the box mode (unique terms in search, t(31) = 2.41, p<.05; number of queries, t(31) = 2.82, p<.01). The data for the various measures for box vs. line mode are displayed in Table 4. It is interesting to note that the number of queries in the line condition, and the number of unique terms in the line condition, are both about double those in the box condition. Although the average query length for the line condition is somewhat longer than for the box condition, this difference is not statistically significant. But when considering specified vs. partially-specified medical topics, average query length is significantly longer (mean specified = 6.34; mean partially-specified = 4.32, t(31)=2.06, p<.05).

Group Statistics

	size of search box	N	Mean	Std. Deviation	Std. Error Mean
number of queries	Line	16	3.19	2.56	.64
	Box	17	1.53	1.18	.29
number of unique terms in search	Line	16	10.19	6.67	1.67
	Box	17	5.29	2.57	.62
AVLENGTH	Line	16	6.28	3.19	.80
	Box	17	4.50	2.49	.60
unique seen	Line	16	27.63	18.65	4.66
	Box	16	17.19	9.70	2.42
unique viewed	Line	16	4.31	2.70	.68
	Box	16	2.56	2.19	.55

Table 4. Query and interaction characteristics for medical topics in line and box modes.

4.2 Performance of task

Performance, that is, correct and complete answering of the topic, was measured by the experimenters on the basis of the pages printed out by each subject. We did this by deciding whether the pages which were printed out either: did not respond to the topic/question at all; partially answered the topic/question; or completely answered the topic/question. Although we also had a measure of performance based on the subjects' assessment after each search, we used our measure in preference because we sometimes found what appeared to be misunderstandings of the tasks on the part of the subjects. Using this measure, we found some interesting results. Figure 1 displays the average length of the query against performance on the task, for all searches. Here we see that as the query gets longer, performance regularly becomes better, although this is a descriptive finding, only.

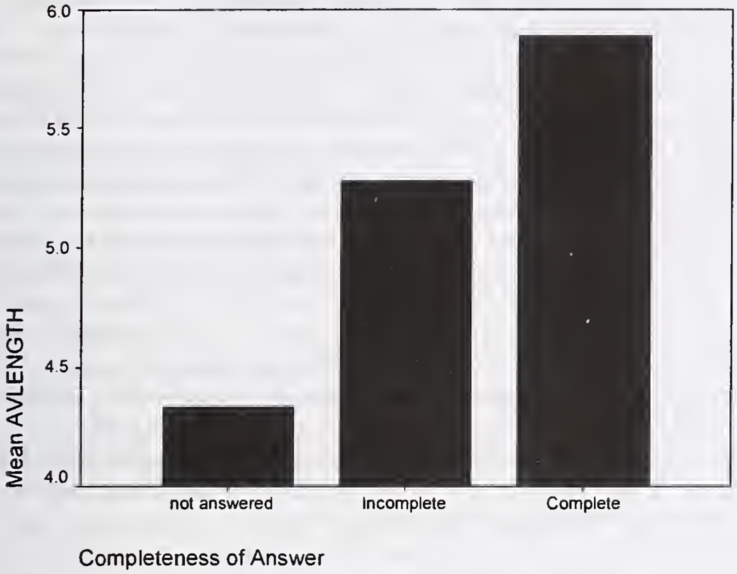


Figure 1. Task performance versus average query length.

Performance was not significantly related to either input mode or query type, nor, in general, to any other query or interaction characteristics. However, when analyzing performance by topic type, the buying topics turned out to be significantly more difficult than the others. These data are displayed in Table 5; the chi-squared test gives χ^2 (6) = 14.89, p<.05.

TASKTYPE * completeness of answer/objective Crosstabulation

Count

		completeness of answer/objective			
		not answered	partially answered	completely answered	Total
TASKTYPE	medical		15	16	31
	travel		13	17	30
	Project		12	19	31
	buying	3	20	9	32
Total		3	60	61	124

Table 5. Performance versus topic type.

4.3 Satisfaction with search

After each search, subjects were asked to indicate their degree of satisfaction with that search on a five-point Likert scale, with 1 indicating unsatisfied, and 5 completely satisfied. Using similar scales, they were also asked to indicate their familiarity with the task (i.e. topic) and whether previous knowledge of the topic had helped them in their search. These measures were correlated with the query and interaction measures indicated before, and with one another. The most interesting results from this analysis are that the major interaction measures, number of queries, unique seen and unique viewed are moderately, but significantly negatively correlated with satisfaction (using the Pearson correlation, these correlations are: number of queries -.350**; unique seen -.314**; unique viewed -.182*;

where **= significance at .01, *=significance at .05). A further interesting relationship is that previous knowledge of the topic is significantly correlated with satisfaction (.380*)

4.4 Query modification and highlighting

There was very little use of the query modification feature by our subjects. Overall, only 9 subjects (26%) used this feature at all. Five of them used it two or more searches, and the feature was used in only 16 (12%) of all of the searches. We asked about the usefulness of the query modification feature in the exit interview, and about the subjects' familiarity with this type of feature. The narrative data are still being analyzed, but we have some preliminary results. Almost all subjects were unfamiliar with this feature (rating of 1 or 2 on a 5-point Likert scale where 1 is completely unfamiliar and 5 is completely familiar, M = 1.72), and almost all rated it as not useful (1 or 2 on a 5-point Likert scale where 1 is useless and 5 very useful, M = 2.03). In general, those who who gave the feature low usefulness scores were also unfamiliar with the feature. The few subjects who gave a high usefulness rating also claimed to have high (4,5) familiarity with this type of feature. When asked to explain the reasons for their usefulness ratings, those who gave negative ratings had three types of reasons, as follows. (1) They didn't see a need for the feature, since it was just as easy to type directly; (2) they were used to doing things differently; and (3) they were unfamiliar with the feature.

We investigated the usefulness of highlighting also by asking about it in the exit interview. As with the query modification feature, the narrative data are still being analyzed, but we present preliminary results. The mean usefulness rating was 4.21, on the same 5-point scale as for usefulness of the query modification feature. This high rating was consistent across all topic categories and subject conditions. Reasons given for high ratings (4,5) were that highlighting made it easier to and quicker to tell when something was relevant, and also to tell when something was not relevant. Reasons give for low ratings (1,2) were that a lot of irrelevant words were highlighted, and that there was confusion between highlighted query terms and links to other pages.

5 Discussion

On the basis of our results, it appears that encouraging people to enter their queries as questions or complete statements will lead to longer queries, which in turn will lead to greater satisfaction with the search, and to better search performance. The strongest result that we obtained with respect to query length was that question/sentence queries were significantly longer than keyword queries. This is certainly at least in part due to the inclusion in the former of words and phrases which are traditionally found on stop lists, and are discouraged or not used in keyword queries (cf. Toms et al., this volume). However, given that the general task that was required of all of the topics was very close to a complex question-answering task, we believe that the inclusion of such words in a query will lead to better performance in systems which are explicitly designed to support this type of task. For instance, all of the systems in the TREC Question and Answer Track make some use of such words (e.g. when, where, what, how) in order to classify the type of question and thereby increase performance. Although the system used in our study, Excite, does not directly support this task, we find it noteworthy that performance did nevertheless increase with query length.

We were unable to replicate Karlgren and Franzen's (1997) results with respect to obtaining longer queries using the box input mode than the line input mode. Our initial hypothesis regarding query input mode, that the box input mode would lead to longer queries than the line input mode, was not directly supported by our results. This failure deserves further analysis of our qualitative data, where subjects were asked to explain their attitudes toward, and preferences for the different query types and input modes. However, we did notice that overall our queries, for both input modes, were slightly longer than what has been reported in the web searching literature. Jansen, Spink and Saracevic (2000) analyzed 51,473 Excite queries and found that on average, the queries contained 2.21 terms. Silverstein, Henzinger, Marais and Moricz (1999) studied over one billion queries from Altavista and found that 72.4% of the queries contained 2 or fewer terms. But our line mode queries were 3.02 words long. In looking for an explanation for these longer queries, we noticed that our line input mode was 50 characters long. This was 32 characters longer than the line input mode in Karlren and Franzen's (1997) study! We conducted a follow-up survey[1] of seven web search services' query input boxes, including Excite and Altavista, in order to better understand the relationship between the size of query input facilities and query length. The size of each search services' query box, measured in characters, was as follows: Altavista, 30; Excite, 37; Google, 35; InfoSeek, 30; NorthernLight, 30; WebCrawler, 33; Yahoo, 30. In only three of these query boxes (Altavista, Google and InfoSeek) could one enter a query that exceeded the number of displayed characters. Thus, we conclude that both our line input mode, and our box input mode encouraged longer queries. This may, perhaps, provide some support for our initial

[1] Survey conducted on 28 January 2002.

ideas about query length and query input size, and at least in part explain why our results did not replicate those of Karlgren and Franzén (1997).

The results for the medical topics with respect to numbers of queries and unique query terms per search in line versus box mode are quite strange, and we have not yet come up with a convincing explanation. Since the average length of each query is about the same in the two conditions, it appears that the increase in the unique number of search terms is due to the increase in the number of queries in the line condition. But we have no reasonable explanation of why this should increase for the line mode only in the medical topics. We considered the possible influence of long words being associated with medical topics, but found no difference in average word length between medical topics and the other topic types.

Performance needs to be better studied. The positive relationship between average query length and performance was encouraging, but how this happens, and in particular how it relates to measures of satisfaction and measures of interaction needs to be clarified. Also, it appears that the actual tasks that the subjects engage in need to be better defined, so that misunderstanding is less likely to happen. Examples of obvious misunderstanding of the task led us to determine performance not according to the subjects' own evaluation with respect to the task, but rather by our own understanding of the task, and of their answers. This is clearly not ideal, and needs to be addressed in future studies of this sort.

The negative correlations between interaction measures and satisfaction could be explained either because the subjects had to do a lot of interaction to get the answer, or because they did a lot of interaction, and still didn't get the answer. The relationship between familiarity (prior knowledge) of the topic and satisfaction is of some interest as well, since neither of these factors appears to have had an effect on performance. This, of course, may be due to our particular performance measure.

The lack of use of the query modification feature has several possible explanations. It was somewhat cumbersome to use, it was not very easy to find, and the idea was not very familiar to the subjects. It is also the case that it might not have been well-suited to the task at hand, especially given its time constraints. Despite these caveats, we think that these results confirm what others, ourselves included, have found; that explicit feedback mechanisms are in general too peripheral to searching tasks to be taken up in a major way by end users.

6 Conclusions

A goal of the TREC 2001 Interactive Track was that the studies conducted in it should lead to hypotheses which could be tested (or research questions which could be investigated) in the TREC 2002 Interactive Track. Our results in this year's Interactive Track raise a good number of issues which should be further investigated, in particular those having to do, on the one hand, with the negative relationship of our measures of degree of interaction with subject satisfaction, and on the other hand, with the relationship of query length to task performance. To address these two general problem areas, we will need to: develop new interaction measures and/or design our study to better measure interaction; develop techniques which we believe will lead to reduced interaction; investigate new methods for increasing query length; study the effect of "non-content-bearing" words in queries on task performance; and, develop some better measure(s) of performance, or at least design the new studies to better allow measurement of task performance.

In our TREC-9 study (Belkin, Keller et al., 2001), we developed and tested an interface which displayed, in two rows of six scrollable panes, the texts of the retrieved documents, beginning at the "best passage" in that document (our MDD interface). We believe that a large part of the interaction that we observed in TREC 2001 searches had to do going back and forth between search result lists and the actual pages to which the results pointed, and that searching through those pages also increased interaction effort. This leads us to our

> **Hypothesis 1**: Displaying search results as in our TREC-9 MDD system will increase user satisfaction over displaying search results as lists of references or links to the full texts of the retrieved documents.

Although we observed a consistent trend of increase in task performance with increased query length, this result is somewhat clouded by the measure of performance that was used, and by uncertainty regarding what aspects of query length led to this result. Thus, we will study methods of increasing query length, and design our TREC 2002 Interactive Track study to investigate the effects of different query types and features on performance, and will test our

> **Hypothesis 2**: Increased query length leads to better task performance.

7 Acknowledgements

This research was funded in part by Grant Number IIS 99-11942 from the National Science Foundation. We again gratefully thank our volunteer subjects, without whose collaboration none of our work could be done.

8 References

Belkin, N.J., Cool, C., Kelly, D., Lin, S.-J., Park, S.Y., Perez-Carballo, J., Sikora, C. (2001) Iterative exploration., design and evaluation of support for query reformulation in interactive information retrieval. *Information Processing and Management, 37*, 403-434.

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 & D. K. Harman (eds.) The Ninth Text REtrieval Conference (TREC-9) (pp. 463-475). Washington, DC: GPO.

Jansen, B. J., Spink, A., & Saracevic, T. (2000). Real life, real users, and real needs: A study and analysis of user queries on the Web. *Information Processing and Management, 36*(2), 207-277.

Karlgren, J. & Franzén, K. (1997) Verbosity and interface design. http://www.ling.su.se/staff/franzen/irinterface.html

MONGREL. Web page for the MONGREL Research Project. http://scils.rutgers.edu/mongrel

Ponte, J. M. & Croft, W.B. (1998) A language modeling approach to information retrieval. In: *SIGIR '98*. New York: ACM, 275-281.

Silverstein, C., Henzinger, M., Marais, J., & Moricz, M. (1999). Analysis of a very large web search engine query log. *SIGIR Forum, 33*(1), 6-12.

SERbrainware at TREC 2001

Pál Ruján
SER Technology Deutschland GmbH
Sandweg 236, D-26135 Oldenburg, Germany
e-mails:firstName.lastName@ser.de

Introduction

SER Technology Deutschland GmbH is the technological arm of SER Systems Inc, Herndon, Virginia. Our company focuses on knowledge-enabled software products, mostly related to document management. At the core of many of our products is a knowledge management engine called SERbrainware, which is being developed by our group in Oldenburg since 1999. This engine contains, among others, a text classifier and an associative access module. Both were used in preparing our entries.

This is our first TREC. In order to get acquainted with the usual procedures, evaluation criteria, etc., we decided to participate first in the filtering track. Due to the fact that we had a rather restricted amount of time - two weeks - at our disposition, we used the commercially available engine version 2.40 without any special add-ons.

Data analysis and system characteristics

As always when facing a new problem, one must first analyze the data. As noted also other participants, the 'new' REUTERS data - although rather well prepared in comparison with other data we dealt with in commercial applications - had some problems related to missing text and identical or almost identical texts being classified differently. In general, the documents belong to more than one class, with the classification scheme used for the filtering track being mostly flat with a few exceptions. Although permitted, we did not take advantage of the hierarchy. In our experience, news companies tend to define their own categories through a complete list of boolean expressions whose presence triggers then the class label. The problem with this approach is that it might be strongly biased and actually imply a 'higher order' clarification scheme, masked by the primary statistics of the text. As an example, consider an example set where in one class one has mostly English, on the other class mostly Berman newswire items. Any statistics based classifier will learn to distinguish the two classes based on this *primary* statistics about different languages. However, the task was actually to distinguish between America football vs. Europe soccer items. We are only beginning to work on text classifiers able to deal with such subtle situations.

Using the visualization tools provided by the engine, we could establish that in many cases the subclass structure did indeed correspond to the main clusters seen in three-dimensional projections. We did not use the 'cleaning' tools to improve the learning sets, hence we used all provided training examples. In addition, we did use exclusively the text, no titles or dateline entries of the news items were included. In retrospective, this was a mistake: some other groups at TREC 10 did very well using exactly those informations. Our software did not take advantage of any extra dictionaries, thesauri, or any external information other than the training sets. The runs were made in our usual work environment: PC's with Athlon 800 MHz processors running under Linux OS.

Understanding the tasks

We considered the routing/batch task to be basically a text classification task and used the standard tools of machine learning to do so. Our classifier was run on multiclass mode, not in the one class against the rest mode (which we call one-from-all → OFA mode). In multiclass mode our classifier constructs a maximal margin Voronoi tiling in the **space of classes**, which has class-error-correcting abilities. Although this approach works in general very well in text categorization tasks, it turned out to be difficult to tune for optimizing the linear and F-ratio utility functions. This fact will be taken into account for our next software generation. Another possibility - which in our interpretation was excluded by the TREC rules - is to use not only the given set of training documents but also statistics gathered from the full Reuters database. Such an optimization for a given data set is called **transduction** and profits from the extra information gathered about the statistics of unlabeled documents. A simple approximation of the transduction protocol implies that we add to the learning set documents classified with a high degree of precision to belong into one or more classes, followed by a relearning step. In our case this is mostly due to the extension of the document vector space by related class terms not contained in the original training set.

The adaptive filtering track seems particularly challenging also from a theoretical point of view. Consider a situation where there are K classes there and there are a possibly large number of items whose class is 'NOT in this set of classes'. Given some initial class information (or none!), a chronologically sorted stream of news items must be processed according to the following 'card playing' rules. Given an actual item, one can read the text but not its label(s). The actual item is like a card placed face-down.

1. Decide to 'turn-up' an item or not. If an item is missed, recall is diminished by d.

2. To turn a card face up, one must guess first one or more labels.
3. If a label is correct, one gets a reward r. If it is not, one gets a penalty p.
4. All information contained on turned-up cards can be used afterwards.
5. There is a limit of the maximal penalty P one can accumulate (per class). Below it this class is 'lost' completely.

What should be the optimal strategy to win at this game? Assume we have a trainable classifier algorithm, for simplicity let us assume that we use on for each class (OFA-structure). From previous experiments we might know the classifier's learning curve, that is how the average generalization error depends on the number of training examples.

This game is quite interesting, because it is similar to the typical problems a company would face in the market place. You must invest in research Let us now enumerate what kind of information would be desirable:

- The size-horizon: it is necessary to estimate how many items one will have to process. Let denote this number by S.
- The priors: from a total of S items, how many k items belong to class $k \in K$?
- The time-correlations, if any, between occurrences of documents belonging to class k, $k \in K$.

There are three aspects which has to be taken into account in order to make a rational (or almost Bayesian) decision about whether to turn up a new item or not. First, we must get some probability estimate from the classifiers concerning the probabilities that this item belongs to one of the classes of interest. A similar kind of information could be obtained from a prediction based on the time-series of successful hits for each class. There two distinct reasons to turn up a card: either the estimated risk $1 - p_c$ for the suggested label c is smaller than r/p, or $p_c = 0.5$ in which case we could get a very good candidate for a new training document - but must pay accordingly some extra price. The situation is made complex by the existence of a maximal penalty, the fact that the priors of different classes are unequal and because in the beginning there is an additional risk associated with a small amount of available data. It is hoped a full theory will emerge before the submission of TREC 2002 runs. It is perhaps interesting to remark that the filtering track could be seen also as a nice model for a commercial firm, where the costs of research and development are restricted to certain maximal losses and the profit generated by it should in the long run be optimized.

Methods

We used basically two methods: the classifier (serCLST10* runs) and an additional 'experimental' run using the associative access module in a nearest neighbor setting (serASSAT10* runs). The classifier requires disjoint classes. Therefore, some training examples are automatically discarded by the engine when the normalized overlap between two items belonging to different classes was bigger than 0.9. When an unknown document is processed, the classifier returns the confidences that the document belongs to every one of the learned classes. The confidences are real numbers between zero and one. With an appropriate normalization they can be transformed into estimated probabilities. The search engine also returns a score between 0 and 1 for ranking the documents by the similarity to a given query. Therefore, the handling of the two modules was quite similar but used different values for the thresholds.

The associative access module is basically tuned for precision: it has a more sophisticated and precise text representation than the classifier. For performance reasons we restricted the queries to be maximally 1K long (after filtering). This seems not to be a very important restriction for the REUTERS news, since by internal company rules the beginning of each item functions also as a kind of summary of the whole document. The actual learning items (their text content) were stored in the associative cache together with their class and ID information. Each unknown document was posed as a query to the search engine, which then returned a list of the best 100 hits. In general, the class of the best hit was accepted if the score was larger than a certain threshold. In special cases to be described later also the second and third class were accepted (the average number of classes per document is about 1.7). This method has absolutely no problems with overlapping classes: however, its classification capabilities are limited when compared to the classifier.

The batch and routing tracks

It is easy to show that the linear utility function whose optimization has been proposed does not scale correctly with size (or time). Therefore, we used default settings of our software, derived from many practical situations we solved before. This means that for the classifier the threshold was set at 0.8 confidence for the best class and at 0.9 for the second best class when the gap to the next (third best class) was greater than 0.05. Note that all learned documents have confidence 1.0. Classes R42 and R57 were ignored, because they overlapped with R40 and R65, respectively. For the search-engine/nearest neighbor classification we used a threshold of 0.4, which is the default score for our Internet application SERglobalBrain. Here we accepted the second and third best class only if they had practically the same score as the best hit. Note that in principle we could have many good hits belonging to the same class, because we compute the similarity to each training item and not to a class 'center'.

Obviously, the classifier performed much better here than the search-method. Since all our decisions were based on scores (confidences) the routing was just a special case of the batch track. We found these two tracks being a test on classification performance, which in turn, depends crucially on the document representation. We do not know how REUTERS defined its classes: it would be an interesting task to try to find this out from the provided data. Anyway - at least at the time of processing - we can be sure that REUTERS did not used advanced software letting employees immediately know how a certain actual news item was classified by another colleague. We guess that some kind of full text engine with boolean SQL expressions might have been used for guidance.

The adaptive track

The adaptive track seems much more interesting from both a theoretical and practical point of view. Due to the time constraints, however, we have directed our efforts to two experiments: we used the classifier for a very 'precise' run, not allowing for negative utility scores at all. In retrospect, this was a poor decision. The search engine run was directed towards recall. Here are the two algorithms we used:

Run serCLST10af

We start with two documents per class, number of classes is constant. At each iteration there are maximally 10 documents per class in the training-queues.

1. Initialize queues, learn training set (2 documents per class)
2. For each new document perform classification and get list of confidences
   - if best confidence < 0.8 ignore
   - else: ask for confirmation. If class is OK and confidence < 0.85 → add to *best* class. If class is false → add to *worst* class
3. Add to best class consists of the following procedure:
   Put the new document with label good at the bottom of the queue. If the queue is full, remove the top document. If a 'bad' marked document exist, remove this instead. Relearn the training set.
4. Add to worst class is similar, except that the new document is marked as bad.

Run serASSAT10ad

We start with two documents per class, number of classes is not constant. As explained below, for each class we might build an additional anti-class. The number of documents per class is not controlled.

1. Initialize and learn training set (2 documents per class)

2. For each new document: use document as query, get ranked list of training examples. Let *score* be one of the best three scores.

    - If score < 0.4 → ignore.
    - If the best score > 0.4 and the corresponding class is an anti-class, ignore.
    - If $0.4 \leq$ score ≤ 0.55 AND class is correct, add document to class if not already done so. If the class is not correct, count one error but do nothing.
    - If score > 0.55 AND class is incorrect, add document to anti-class set. If class is correct, count a good hit but otherwise do nothing.

3. After each **day** reload the actual set of training documents into associative cache.

Our results in both the routing and adaptive track were on the low-middle range of the spectrum. We did learn a lot from this competition and we are looking forward hope to perform much better next time!

Aggressive Morphology and Lexical Relations for Query Expansion

W. A. Woods　Stephen Green　Paul Martin
Ann Houston
Sun Microsystems Laboratories
1 Network Drive
Burlington, MA 01803
{William.Woods,Stephen.Green,Paul.Martin,Ann.Houston}@east.sun.com

1 Introduction

Our submission to TREC this year is based on a combination of systems. The first is the conceptual indexing and retrieval system that was developed at Sun Microsystems Laboratories (Woods et al., 2000a; Woods et al., 2000b). The second is the MultiText system developed at the University of Waterloo (Clarke et al., 2000; Cormack et al., 2000).

The conceptual indexing system was designed to help people find specific answers to specific questions in unrestricted text. It uses a combination of syntactic, semantic, and morphological knowledge, together with taxonomic subsumption techniques, to address differences in terminology between a user's queries and the material that may answer them. At indexing time, the system builds a conceptual taxonomy of all the words and phrases in the indexed material. This taxonomy is based on the morphological structure of words, the syntactic structure of phrases, and semantic relations between meanings of words that it knows in its lexicon.

It was not, however, designed as a question answering system. Our results from last year, while encouraging, showed that we needed more work in the area of question analysis (i.e., "What would constitute an answer to this question?") and answer determination (i.e., "Does this retrieved passage actually answer the question?") to support our relaxation ranking passage retrieval algorithm.

After conversations with the researchers at the University of Waterloo, we decided to submit a run where we would provide front-end processing consisting of query formulation and query expansion using our automatically derived taxonomy and Waterloo would provide the back-end processing via their MultiText passage retrieval system and their answer selection component. The result is a direct comparison of two question answering systems that differ only in the query formulation component.

2 The Conceptual Taxonomy

As we said earlier, Sun's conceptual indexing system builds a taxonomy of all the words (and possibly phrases) that are encountered during indexing. This taxonomy is built around the generality relationships between terms. More general terms are said to *subsume* more specific terms. There are three sources of knowledge that are used to build the taxonomy for a set of documents: syntactic structure of phrases, semantic subsumption relationships between words and word senses, and morphological structure and relationships between words. For this experiment, we used only the latter two sources of knowledge to expand terms in the input questions:

1. Semantic subsumption axioms. These are encoded in the lexicon used by the indexing system. The largest base lexicon currently used by the system system contains semantic subsumption information for something in excess of 15,000 words. This information consists of basic "kind of" and "instance of" information such as the fact that *book* is a kind of *document* and *washing* is a kind of *cleaning*.
2. Morphological rules. The current morphology component consists of approximately 1200 knowledge-based morphological rules. These rules cover prefixes, suffixes, lexical compounds, as well as some special cases (e.g., phone numbers).

The conceptual taxonomy for this experiment is constructed automatically as a byproduct of indexing the TREC material with the conceptual indexing system. As each word is encountered during the indexing process, it is looked up in the system's lexicon and if not found, it is given an entry whose content is determined by the morphological analysis component. This entry is used for any subsequent occurrences of the same term. The rules in the morphological analysis system can infer syntactic parts-of-speech for the word, morphological relationships to other words, and sometimes even semantic relationships to other words.

After the conceptual indexer has assured that the word has a lexical entry, the word is entered into the conceptual taxonomy if it is not already there. Then all of its root words and any more general concepts that are listed in this lexical entry are also entered into the conceptual taxonomy in the same way, and this word is recorded as being subsumed by those words. This process is carried on recursively so that a word's parents' parents are recorded and so on, until there are no more indirect parents that are not already in the taxonomy.

Thus the conceptual taxonomy includes all of the terms found in the indexed material, plus all of the more general terms that are known in its lexicon or inferred by morphological rules. Furthermore, the conceptual taxonomy contains only words that were induced by this process.

2.1 Aggressive Morphology

The morphological analysis component of this system makes use of a large set of morphological rules that can recognize and analyze words that are derived and inflected forms of known words, as well as words that appear to be derived or inflected forms of unknown words. They can also make plausible inferences about the syntactic categories of unknown words that do not appear to be derived from other words.

The morphological analysis system considers both prefixes and suffixes and their interaction, and it also recognizes and analyzes lexical compounds formed from concatenating known words. For example, in the TREC-10 collection, "pointy," "repoint," "repointing," and "standpoint" were analyzed as forms of point."

The morphological analysis system makes use of different kinds of morphological rules, applied in a preferred order to words that are not already in the lexicon. Generally, the rules are ordered in decreasing order of specificity, confidence and likelihood. Very specific tests are applied in Step 1 to identify and deal with "words" that are not ordinary sequences of alphabetic characters. These include numbers, alphanumeric sequences, and expressions involving special characters. Failing this, an ordered sequence of suffix rules is applied in Step 2 in a first pass that will allow a match only if the proposed root word is "known." The same list of rules will be applied later in a second pass without this known-root condition if an earlier analysis does not succeed.

If no phase-one suffix rules apply, prefix rules are tried in Step 3 to see if an interpretation of this word as a prefix combined with some other "known" word is possible. Failing this, a set of lexical compound rules is tried, in Step 4, to see if the word is interpretable as a compound of two or more words, and failing that, lists of first and last names of people and names of cities are checked in Step 5. All of steps 3–5 are considered more reliable if they succeed than a phase-two pass of the suffix rules without any restriction to known roots that comes in Step 6. This ordering allows prefixes and compounding to be tried before less confident suffix analyses are attempted, and avoids applying weak suffix analyses to known names.

Lexical compound rules are called by a specialized interpreter that looks for places to divide a word into two pieces of sufficient size. The points of potential decomposition are searched from right to left, and the first such point that has an interpretation is taken, with the following exception: The morph compound analyzer checks for special cases where, for example, the first word is plural and ends in an *s*, but there is an alternative segmentation in which the singular of the first word is followed by a word starting with the *s*. In such cases, the decomposition using the singular first word is preferred over the one using the plural. For example, the word *minesweeper* will be analyzed as *mine+sweeper* rather than *mines+weeper*.

2.1.1 Interaction of rules with semantic axioms

It is useful to do full morphology on unknown words and to know morphological relationships for words in the lexicon in order to make connections between derived forms of words and semantic subsumption facts that may be known about their roots. For example, destruction may link morphologically to destroy, which then links semantically to damage. A simple stemming technique would not be able to find such connections (unless all the semantic axioms were similarly stemmed, a process that would result in many false subsumption paths in the taxonomy, due to the kinds of noise and errors that result from stemming algorithms).

3 Query Formulation and Term Expansion

Because the taxonomies that the conceptual indexing system builds are specific to a particular document collection, the first step of our query processing was to re-index the TREC Question Answering collection using the latest revision of the conceptual indexer. We then ran the queries through a modified version of the query formulation component of the system we used for TREC-9. The reformulated queries were then passed to the term expansion system.

The query formulation component is based on the pilot version of our conceptual indexing system. The query formulator interprets the question words and the format of the question to determine the desired answer type. It also either replaces the question word in the query with the desired answer type or simply removes it from the request. In addition, the query formulation component will generalize some terms (e.g., *high* for *tall*), substitute base forms for inflected forms (e.g., *principle* for *principles*), and drop some "noise" terms.

The query formulation component that we used this year differs from the one used for TREC-9 in small ways. First, we did not limit the number of terms in the reformulated query as we did last year. This limitation was due to a limitation of our passage retrieval system which is not a problem for the MultiText system. Second, we fixed a limitation to allow a query term to be expanded from each of its roots when there was more than one root for that term (e.g., saw from root see as well as root saw). Finally, we fixed a bug that generated incorrect answer types for certain question forms.

During a typical query run with the relaxation ranking passage retrieval system, a query term is expanded to include all terms that are subsumed by that term. In some cases the expansion can be quite dramatic. For example, in the AP sub-collection of the TREC QA collection, the query term *person* expands to more than 17,000 other terms. These terms include morphological variations such as *people* and *persons* as well as semantic variations such as *blacksmith* and *lawyer*. This set of terms reaches 25 levels deep into the conceptual taxonomy.

Although such a wide-ranging expansion may seem counter-intuitive, there doesn't seem to be any *a priori* way to determine a reasonable cutoff level. We can decide on using a small integer, say 2, but this may preclude useful expansions such as *counterrevolutionary*, which may be crucial in finding just the right document.

In the end, we decided that we would cut the expansion off after the first level of expansion, since this would get most of the morphological expansions for the term and many of the semantic expansions. Even with this stringent criterion, the query term *person* still expands to more than 8,000 terms.

We originally intended to integrate the answer types from our query formulation stage with the answer types from the MultiText system, but that proved not to be possible in the time available, so we ended up providing only the selected query terms and their expansions to the MultiText back end.

The reformulated, expanded queries were passed to the MultiText system as a conjunction of disjunctions of each of the expanded terms.

4 Results

The results for our two submitted runs as well as the corresponding MultiText runs are shown in table 1. The MultiText system seems to have done better on its own than with the Sun query formulation and expansion engine as the front end. In part, this may be due to the lack of full integration between our front end and the MultiText back end, and in part is may be due to the selection of query terms from the original question that was done by the query formulation stage. It is interesting to note that there are a significant number of questions that each system answered that the other didn't. It is interesting to look at the cases where the combined Sun/MultiText system found answers to questions that were not found by the MultiText system alone, and vice versa.

	NIST Judgment	
Run	Strict	Lenient
Sun baseline (mtsuna1)	0.307	0.322
Sun with Web reinforcement (mtsuna0)	0.405	0.418
MultiText baseline (uwmta2)	0.346	0.365
MultiText with Web reinforcement (uwmta1)	0.434	0.457

Table 1: Results for the main QA task.

Run	Sun only	MultiText only	Sun and MultiText	Neither
Baseline	38	61	193	200
Web reinforcement	47	55	225	165
Intersection	26	32		

Table 2: Differences in answers found

Table 2 shows the number of questions for which answers were found by only one of the systems, the number of questions for which answers were found by both systems, and the number of questions for which neither system found an answer. The row labeled "Intersection" shows the number of questions that were found in both of the Sun runs and neither of the MultiText runs, and vice versa. For the rest of the discussion, we will focus on the 26 questions that were found in both of the Sun runs, and neither of the MultiText runs.

There are two ways that the query expansions can affect whether an answer is found or not. First, the expanded query may give the passage retrieval component enough information to retrieve a passage that would not be found with the unexpanded query. Second, the expanded query may give the answer selection component better information about what words would make up a useful answer to the query.

A variation of the second type appears to have occurred for question 910, "What metal has the highest melting point?" In both runs, the top passage retrieved by the MultiText system contained the correct answer, but it appears that the answer selection component determined only for the Sun runs that this document held an answer.

In this case, the Sun query formulation stage transformed the original question to the sequence of terms: (METAL HIGHEST MELT POINT), which expanded to:

```
(metal aluminium bimetal chrome copper coppers dimetal gunmetal
immetal intermetal lead leads lithium metal's metaled metaler metalic
metalist metalled metallic metallist metallize metallized metallizing
metallurgy metalor metals metalware nickel nickeled nickeling nickels
nonmetal nonmetals palladium polymetal rust rusts silver silvers
sliver slivered slivers tin tins)
(highest top)
(melt melted melter melting melts melty molten re-melt remelt smelt
smelted smelting smelts)
(point barb barbs breakpoint breakpoints checkpoint checkpoints
crosspoint cusp cusps depoint endpoint endpoints gunpoint interpoint
isopoint middle middles midpoint midpoints multipoint nib nibs
nonpoint outpoint outpointing outpoints point's pointed pointer
pointers pointeur pointful pointing pointless points pointy repoint
repointing standpoint standpoints subpoint tip tips viewpoint
viewpoints wellpoint)
```

This example also illustrates a potential negative interaction between our expansions and the MultiText answer selection strategy, since in this case the answer could well be one of the query expansion terms, and could then be rejected by the heuristic of looking for answers that are non-query terms.

Question 922, "Where is John Wayne airport", shows both effects. The sets of passages that were retrieved are completely disjoint. This query was only slightly expanded, and even though the answer was in passages in both sets, for the MultiText runs, the answer selection component seems to have focussed on other proper names than those in the query. We suspect that this is an effect of the removal of all query words before candidate selection. The query formulation stage transforms this to (JOHN WAYNE AIRPORT), which expands to:

```
(john dejohn demijohn john's johner johni johnie johny saint-john)

(wayne dewayne wayne's wayner waynes)

(airport airport's airporter airports multiairport)
```

With respect to questions that MultiText gets answers for that the joint Sun/MultiText system doesn't, a comparison of the two sets suggests that the joint system does better on questions with more complex descriptions (e.g., 1231 "What fruit is Melba sauce made from?") while the MultiText system does better alone on short direct questions (e.g., 1266 "What are pathogens?"). Another noticable pattern is that many of the questions for which MultiText does better alone contain plurals that the Sun question formulation stage generalizes to singulars (e.g., "Great Lakes" and "x-rays") where the plural is probably a better retrieval clue. These comparisons suggest that there is still a lot of tuning to be done to match the query formulation stage to the retrieval stage and answer selection stage, and that a switch between the two techniques based on the question type would do better than either one alone.

5 Conclusion and Future Work

We've only done a preliminary analysis of the results at this point. The results show that for some queries, the morphological and semantic expansions help, while for others (unfortunately somewhat more others) they degrade the results. This is typical of the impact of this kind of expansion on many retrieval techniques, with the notable exception of the penalty-based passage retrieval technique used in Sun's conceptual indexing system.

We haven't yet developed a full picture of how this aggressive expansion might be degrading the retrieved passages of the MultiText system. We have some hope that the MultiText system's passage retrieval method is fairly resistant to degradation, but we will have to further analyze the data to find out if that is the case, and if not, what can be done about it.

So far, the results above suggest a possible refinement to our front-end/back-end strategy. Since it seems clear that there are instances where the expanded queries aided more in the answer selection than in the passage selection, it would be interesting to try a run where the unexpanded queries are used for passage retrieval and the expanded queries are used during answer selection. Another obvious thing to try is a conditional system that uses the expansion technique for the longer more complex question types and avoids expansion for direct short questions, especially those that look like they are asking for the definition of a term.

References

(Clarke et al., 2000) C.L.A. Clarke, G.V. Cormack, D.I.E. Kisman, and T.R. Lynam. Question Answering by Passage Selection (MultiText Experiments for TREC-9). In *Proceedings of The Ninth Text REtrieval Conference (TREC 9)*. National Institute of Standards and Technology, 2000.

(Cormack et al., 2000) G. V. Cormack, C. L. A. Clarke, C. R. Palmer, and D. I. E. Kisman. Fast Automatic Passage Ranking (MultiText Experiments for TREC-8). In *Proceedings of The Eighth Text REtrieval Conference (TREC 8)*. National Institute of Standards and Technology, 2000.

(Woods et al., 2000a) W.A. Woods, Stephen Green, Paul Martin, and Ann Houston. Halfway to Question Answering. In *Proceedings of The Ninth Text REtrieval Conference (TREC 9)*. National Institute of Standards and Technology, 2000.

(Woods et al., 2000b) William A. Woods, Lawrence A. Bookman, Ann Houston, Robert J. Kuhns, Paul Martin, and Stephen Green. Linguistic Knowledge can Improve Information Retrieval. In *Sixth Annual Applied Natural Language Processing Conference*, pages 262–267. Association for Computational Linguistics, 2000.

Question Answering : CNLP at the TREC-10 Question Answering Track

Jiangping Chen, Anne R. Diekema, Mary D. Taffet, Nancy McCracken, Necati Ercan Ozgencil, Ozgur Yilmazel, and Elizabeth D. Liddy

Center for Natural Language Processing
Syracuse University, School of Information Studies
4-206 Center for Science and Technology
Syracuse, NY 1324-4100
www.cnlp.org
{jchen06,diekemar,mdtaffet,njm,neozgenc,oyilmaz,liddy}@syr.edu

Abstract
This paper describes the retrieval experiments for the main task and list task of the TREC-10 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, and keywords. 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

Question-answering systems retrieve answers rather than documents in response to a user's question. In the TREC question-answering track a number of question-answering systems attempt to answer a predefined list of questions by using a previously determined set of documents. The research is carried out in an unrestricted domain.

The CNLP question answering system uses a two-stage retrieval approach to answer-finding based on matching of named entities, linguistic patterns, and keywords. 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. Then the information is passed on to answer finding modules, which take the documents, retrieved in the first stage, for further processing and answer finding. The answer finding module uses three separate strategies to determine the correct answer. Two strategies are based on question focus, and the third strategy, based on keywords, is used when the question focus is not found or when the first two strategies fail to identify potential answers. A detailed system overview can be found in section 3.

2. Problem description

CNLP participated in two of the three QA track tasks: the main task and the list task. The main task is a continuation of last year's QA track in that systems are required to answer 500 short, fact-based questions. Two new aspects were introduced this year: unanswerable questions (with no answer present in the collection), and the notion of an answer confidence level. Systems needed to identify unanswerable questions as such, in order for them to be counted as correct. For the confidence level systems needed to state the rank of their final answer or state that they were unsure about their answer. For each question, up to five ranked answer responses were permitted, with the most likely answer ranked first. The maximum length of the answer string for a submitted run was 50 bytes. A response to a question consisted of the question number, the document ID of the document containing the answer, rank, run name, and the answer string itself.

The list task questions are similar to those of the main task but include an indication as to how many answer instances needed to be provided for an answer to be considered complete. A response to a list task question consisted of an unordered list with each line containing the question number, the document ID of the document containing an answer instance, and the answer string itself. The length of the list or the number of answer instances for each question is specified in the question. As in the main task, the maximum length of each answer string is 50 bytes. The different answer instances could be found within single documents or across multiple documents or a combination of both. There was no guarantee that all requested answer instances could indeed be found in the collection.

Answers to both main task and list task questions had to be retrieved automatically from approximately 3 gigabytes of data. Sources of the data were: AP newswire 1988-1990 (728 Mb), Wall Street Journal 1987-1992 (509 Mb), San Jose Mercury News 1991 (287 Mb), Financial Times 1991-1994 (564 Mb), Los Angeles Times 1989, 1990 (475 Mb), and Foreign Broadcast Information Service 1996 (470 Mb). The submitted answer strings for all tasks were evaluated by NIST's human assessors for correctness. [10] Examples of questions for both the main task and list task can be found in table 1.

TREC-10 QA questions
Main task questions: How much does the human adult female brain weigh? , Who was the first governor of Alaska? , When was Rosa Parks born? , Where is the Mason/Dixon line? , Why is a ladybug helpful? , Where is Milan? , In which state would you find the Catskill Mountains? , What are invertebrates?
List task questions: Name 2 U.S. dams that have fish ladders. , What are 6 names of navigational satellites? , Who are 6 actors who have played Tevye in "Fiddler on the Roof"? , Name 20 countries that produce coffee.

Table 1. Examples of TREC-10 questions.

3. System overview

The CNLP question-answering system consists of four different processes: question processing, document processing, paragraph finding, and answer finding. Each of the processes is described below.

3.1 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 (see section 3.3. and section 3.4). Last year we used our L2L module for first-stage retrieval but this year we relied solely on the ranked list of documents retrieved and provided by NIST. L2L was modified this year to also include query expansion for nouns and verbs found in WordNet 1.6 [8]. Based on the parts-of-speech of the question words, the system added all related synonyms of the first, most frequently used sense (see example at the end of this section) to the L2L representation.

Question focus recognition is performed in order to identify the type of answer expected. Expected answers fall into two broad groups – those based on lexical categories, and those based on answer patterns. Expected answers based on lexical categories can be identified from the terms used in the question. For example, in the question "What river flows between Fargo, North Dakota and Moorhead, Minnesota?", we identify that the questioner is looking for the name of a river. The expected answer type, and therefore the question focus, is *river*. Expected answers based on answer patterns are predicted by the recognition of certain question types. If the question is recognized as a definition question, then the answer sentence is likely to include one of several patterns, such as apposition, presence of a form of the verb *be*, etc.

The question focus recognition routine extracted four elements – the question focus, the lexical answer clue, the number of answers required (used for the list task only), and the confidence level (not fully implemented). In an effort to improve question focus recognition this year, we trained the Brill part-of-speech tagger [2] on questions from TREC 8, TREC 9 and HowStuffWorks. [7] The resulting rules were used to tag the TREC 10 questions. The tagged questions were then run through the Collins parser [3] [4] for a full parse.

There are three steps to question focus assignment. In the first step, the question type is determined using predefined search patterns based on regular expressions. There are 7 special question types (acronym, counterpart, definition, famous, standfor, synonym, why) and 7 standard question types (name-a, name-of, where, when, what/which, how). If a special question type is recognized, then the question type becomes the question focus. Second, the parsed question is examined to extract the lexical answer clue (word or phrase) using the predefined search patterns. In the third step, which applies only to standard question types, the lexical answer clue is used to assign the question focus based on lexical categories where possible. Table 2 is a review of the questions types. Predefined search patterns were developed for these question types:

	Question type	# of search patterns	Example question
Standard question types	Name-a	3	Name a food high in zinc. (TREC 10, question 1268)
	Name-of	2	What is the name of Neil Armstrong's wife? (TREC 10, question 1007)
	Where	10	Where is John Wayne airport? (TREC 10, question 922)
	When	9	When is the official first day of summer? (TREC 10, question 1331)
	What/Which	14	What is the capital of Mongolia? (TREC 10, question 1050)
	Who	9	Who lived in the Neuschwanstein castle? (TREC 10, question 1281)
	How	12	How tall is the Gateway Arch in St. Louis, MO? (TREC 10, question 971)
Special question types	Acronym	4	What is the abbreviation for Texas? (TREC 10, question 1172)
	Counterpart	2	What is the Islamic counterpart to the Red Cross? (TREC 9, question 454)
	Definition	7	What is autism? (TREC 10, question 903)
	Famous	5	Why is Jane Goodall famous? (TREC 9, question 748)
	Standfor	4	What does the technical term ISDN mean? (TREC 10, question 1219)
	Synonym	3	What is the colorful Korean traditional dress called? (TREC 10, question 1151)
	Why	1	Why does the moon turn orange? (TREC 10, question 902)

Table 2. Question types

Additional processing performed by the question focus assignment routine includes the extraction of the number of answers required (used for the list task only), and assignment of a confidence level. The number of answers required was extracted based on the predefined search patterns for each question type. The confidence level assigned ranged from 0 to 5, with 5 being the highest level of confidence in the question focus. If the question focus could not be determined, the confidence level was 0. Otherwise, the confidence level was set at a value ranging up to 5 depending on the certainty of the question focus. Due to the short time available for development, confidence level assignment was only partially implemented for TREC 10 and therefore not used in the experiments.

The output resulting from the L2L module and the Question Focus recognition module is passed on to the paragraph finding module, the answer candidate recognition module, and the answer formatting module. A standard question type (in this case what/which), will produce the following output for the question "What is the deepest lake in the US?":

```
Logical representation:   deep* lake* +US ( "United States" "United States of America" America
                          U.S. USA U.S.A.)
Query focus:              lake#deepest lake#2#5
Tagged:                   <sentence sid="s0"> what|WP be|VBZ the|DT <CN> deep|JJS lake|NN
                          </CN> in|IN the|DT <NP cat="cntry" id="0"> US|NP </NP> ?|.
                          </sentence>
```

A special question type (in this case definition), will produce the following output for the question "Who is Duke Ellington?":

Logical representation:	+Duke* +Ellington*
Query focus:	def#Duke Ellington#2#5
Tagged:	<sentence sid="s0"> who|WP be|VBZ <NP cat="per" id="0"> Duke|NP Ellington|NP </NP> ?|. </sentence>

As can be seen in the examples above, expansions from WordNet are enclosed in parentheses, and the four elements in the question focus are separated by '#'.

3.2 Document processing

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.

3.3 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. Although we used the same strategy as last year to identify the paragraphs, we decided to experiment with the selection process itself. For one set of runs we took the original document and divided it up into paragraphs, based on textual clues. After selecting the top 300 most relevant paragraphs we tag only those paragraphs. This approach is identical to our TREC9 approach and these runs are labeled "PAR" (paragraph tagging). For the other set of runs we tagged the original document first, then divided it up into paragraphs from which the top 300 paragraphs were selected. These runs are labeled "DOC" (document tagging). Paragraph detection is no longer based on orthographic clues (i.e. indentations) for the "DOC" runs because this information is removed during the tagging process. The tagged document is divided into several sentence groups based on a pre assigned value that specifies the approximate number of words in each sentence group.

We hypothesized that tagging the whole document versus isolated paragraphs should provide better named entity identification. Named entities are often referred to in their full form early in a document, only to be reduced to a shorter form later on. When an isolated paragraph is presented to our system for tagging, the context information of the preceding paragraphs is not available for entity categorization, thus hindering tagging performance. The complete documents as well as the individual paragraphs were part-of-speech tagged and categorized by <!**meta**Marker>^{TM} using CNLP's categorization rules.[1] The quality of selected paragraphs and the system's categorization capabilities directly impact later processing such as answer finding.

3.4 Answer finding

The answer finding process (see sections below) takes the tagged paragraphs from the paragraph finding stage (for "DOC" as well as "PAR" runs) and identifies different paragraph windows 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 or additional clue words. All answer candidates were weighted and the top 5 (main task) or top *n* (list task) were selected. The answer finding process expanded answer finding strategies without making major changes to the weighting strategy.

3.4.1 Paragraph-window identification and selection

Paragraph windows were selected by examining each occurrence of a question keyword in a paragraph. Each occurrence of a keyword in relation to the other question keywords was considered to be a paragraph window. A keyword that occurred multiple times thus resulted in multiple paragraph windows, one for each occurrence. A weight for each window was determined by the position of the keywords in the window and the distance between them. An alternative weighting formula was used for single-word questions. The window with the highest score was selected to represent that paragraph. The process was repeated for all 300 paragraphs resulting in an ordered list of paragraph windows - all potentially containing the answer to the question.

3.4.2 Answer candidate identification

This year we focused on expanding the answer candidate identification ability of the system by changing the answer finding strategies and adjusting our weighting schemes based on the TREC9 question set.

Answer candidate identification involves three separate strategies. Two strategies are based on question focus, and the third strategy, based on keywords, is used when the question focus is not found or when the first two strategies fail to identify potential answers. The two question focus strategies include search for a specific lexical category in the case of standard question types and search for a specific answer pattern in the case of special question types (see section 3.1). Which strategy is initially employed for a particular question is based on the value found in the question focus element in the question focus line. If the question focus value matches one of the special question types, then the specific answer pattern strategy is used. If the question focus has a value of "unknown", the third strategy involving keywords is invoked as a fallback. For all other values of the question focus element, the specific lexical category strategy is employed. For a discussion of the specific lexical category strategy and the keyword strategy see our TREC 9 paper. [5] For each special question type (acronym, counterpart, definition, famous, standfor, synonym, why), one or more answer patterns have been identified and defined in the answer candidate identification routine.

3.4.3 Answer-candidate scoring and answer selection

The system used a weighting scheme to assign a weight to each answer candidate. Although we intended to change the weighting scheme to accommodate the new answer finding strategies we ran out of time. The weight was based on the keywords (presence, order, and distance), whether the answer candidate matched the question focus, and punctuation near the answer candidate.

This resulted in a pool of at least 300 candidates for each question. A new unique-answer-identifier module removed duplicate answers from the answer-candidate list. The top 5 highest scoring answer candidates were selected as the final answers for each question for the main task. The required number of answers, identified during question processing, determined the number of answers for the list task questions. The answer strings were formatted according to NIST specifications.

4. Results

We submitted four runs for the TREC10 QA track: two runs for the main task and two runs for the list task. Each run name can be parsed into four components: 1) organization name, 2) trec, 3) tagging approach (see section 3.3), and 4) task.[1]

4.1 Main task results

Averages over 492 questions[2] (strict evaluation):	SUT10DOCMT	SUT10PARMT
Mean reciprocal rank	0.148	0.218
Questions with no correct answer found	381 (77.4 %)	332 (67.5 %)
Questions with rank above the median	80 (16.3 %)	117 (23.8 %)
Questions with rank on the median	345 (70.1 %)	329 (66.9 %)
Questions with rank below the median.	67 (13.6 %)	46 (9.3 %)
Correctly Answered NIL questions	0 (out of 3)	0 (out of 3)

Table 3. Question answering results for the main task.

The evaluation measure for the main task (see Table 3) is the mean reciprocal answer rank. For each question, a reciprocal answer rank is determined by evaluating the top five ranked answers starting with one. The reciprocal answer rank is the reciprocal of the rank of the first correct answer. If there is no

[1] SU = Syracuse University, T10 = TREC10, DOC = tag entire document / PAR = tag individual paragraphs, MT = main task / LT = list task

[2] The initial question set of 500 questions was reduced to 492 questions after 8 questions were discarded by the National Institute for Standards and Technology.

correct answer among the top five, the reciprocal rank is zero. Since there are only five possible ranks, the mean reciprocal answer ranks can be 1, 0.5, 0.33, 0.25, 0.2, or 0. The mean reciprocal answer ranks for all the questions are summed together and divided by the total number of questions to get the mean reciprocal rank for each system run.

4.2 List task results

Averages over 25 questions	SUT10DOCLT	SUT10PARLT
Average Accuracy	0.25	0.33
Questions with no correct answer found	4 (16 %)	5 (20 %)
Questions above the median	13 (52 %)	15 (60 %)
Questions on the median	9 (36 %	7 (28 %)
Questions below the median	3 (12 %)	3 (12 %)

Table 4. Question answering results for the list task.

The evaluation measure for the list task (see Table 4) is average accuracy. For each question accuracy is determined by the number of distinct correct answers over the target number of instances to retrieve. Accuracy for all the questions is summed together and divided by the total number of questions to get the average accuracy.

5. Analysis

The main task analysis examines: (5.1) retrieval performance of first stage retrieval based on the ranked list provided by NIST, (5.2) the Language-to-Logic module, (5.3) question focus assignment, (5.4) query expansion, and (5.5) the difference between the tagged document and tagged paragraph run performance. The list task analysis (5.6) examines list task performance, instance assignment, and the difference between the tagged document and tagged paragraph run performance.

5.1 First stage retrieval

As mentioned previously, we used the ranked document list as provided by NIST for first stage retrieval. The retrieved lists were created using the PRISE system [6]. For TREC9 NIST used the SMART [9] information retrieval system (see Table 5).

Top 200 results	TREC9	TREC10
Questions without any retrieved documents	0	0
Questions without any relevant retrieved documents	48	32
Questions for which there are no relevance judgments	20	48
Questions with relevant retrieved documents	625	420
Total number of questions	693	500
Total number of documents retrieved	134,600	90,400
Number of known relevant documents	7,963	4,465
Total number of relevant documents retrieved	6,014	2,966
Average precision	0.29	0.23

Table 5. First stage retrieval performance.

Compared to last year's retrieval results, both the number of known relevant documents as well as the average number of retrieved relevant documents for each question decreased. The TREC10 retrieval results might have increased the difficulty of finding correct answers.

5.2 Question representation

A logical representation of the question is created in the question processing stage (see section 3.1). The question representation analysis of this year is based on the main task tagged "PAR" run (SUT10PARMT). We noticed that there were much more short questions this year than the previous two years. Even after query expansion, our system still produced 45 (9 %) single word queries and 64 (12.8 %) two-word queries. Many of these questions are "What/Who is/are/was/were" questions which asked for a definition of a person or a thing. Short queries, although represented correctly, may lead to failure in answer finding because the current weighting strategy has not been adapted to them. After excluding short queries, 73 (14.6 %) questions had various representation problems. The major query representation problems include keyword selection problems; part-of-speech errors; and misplaced wildcards (see Table 6).

Problem count	Problems with description
30	*Keyword selection problems* Content words such as numbers were erroneously filtered out or truncated, or inappropriate words were selected
16	*Part-of-speech tagging errors* Wrong tags led to incorrect morphological processing and query expansion error
13	*Misplaced wildcards* Wildcards placed in the wrong place of single words created bad stems

Table 6. Question representation problems.

Compared with the query representation of last year, the system has improved the part-of-speech tagging, but did worse on keyword selection. Some important numbers, such as the number in question "What city has the zip code of 35824? " were filtered out by the system, which had a negative impact on answer finding.

Our conclusion of last year held true - query representation problems only accounted for part of the failure of answer finding. The "PAR" run contained 160 questions that did find the correct answer: 53 (33 %) were short queries, and 19 (12 %) had various query representation problems. The procedure we developed for answer candidate identification helped finding answers for short queries. However, the system did not find the correct answers for most of the questions even when the query representations were correct. Further analysis is needed to identify why this is the case.

5.3 Question focus

As described in section 3.1, we determined the question focus based on special question patterns and lexical answer clues. The question focus analysis is based on the main task "PAR" run (SUT10PARMT). Out of 492 answerable questions, our system determined a question focus for 365 (74.17%) of the questions, more than 10 percent better than TREC-9 (see Table 7). [5] Our efforts to improve focus recognition aided in this increase. Out of these 365 questions, 322 questions (88.2 %) had a correct focus, and 43 questions (11.8 %) had an incorrect focus. Not only did we find a question focus for a greater percentage of questions this year, we also found the *correct* focus for a greater percentage of questions as well. For 127 (25.8 %) questions, our system could not determine a focus.

	Correct question focus	Incorrect question focus	No determinable question Focus
Rank 1	68 (21.1 %)	7 (16.3 %)	3 (2.4 %)
Rank 2	27 (8.4 %)	2 (4.7 %)	2 (1.6 %)
Rank 3	15 (4.7 %)	1 (2.3 %)	2 (1.6 %)
Rank 4	12 (3.7 %)	2 (4.7 %)	5 (3.9 %)
Rank 5	12 (3.7 %)	1 (2.3 %)	1 (0.8 %)
Rank 0	188 (58.4 %)	30 (69.8 %)	114 (89.8 %)
Total	322	43	127

Table 7. Answer rank distribution of question focus status.

An analysis of the special question types (see Table 8) shows that some of the special question routines (definition, standfor) aided in finding the answer. Our ability to find the answers for definition type questions in particular is improved over last year. But since the majority of special question types still failed to find a correct answer, more work is needed.

	Acronym	Definition	Standfor	Synonym	Why
Rank 1		19 (19.4 %)	4 (40.0 %)		
Rank 2		7 (7.1 %)			
Rank 3		9 (9.2 %)			
Rank 4		3 (3.1 %)			
Rank 5		5 (5.1 %)			
Rank 0	1 (100 %)	55 (56.1 %)	6 (60.0 %)	5 (100 %)	4 (100.0 %)
Total	1	98	10	5	4

Table 8. Analysis of Special question types[3]

An analysis of lexical answer clues (see Table 9) shows that having the correct lexical answer clue aids in finding the correct question focus.

	Correct question focus	Incorrect question focus	No determinable question focus
Correct Lexical Answer Clue	295 (91.6 %)	70 (64.8 %)	55 (88.7 %)
Incorrect Lexical Answer Clue	27 (8.4 %)	38 (35.2 %)	7 (11.3 %)
Total = 492	322	108	62

Table 9. Lexical Answer Clue vs. Question Focus

In summary, our efforts to improve focus recognition led to a greater percentage of both identified question focus and correctly identified question focus. Having a question focus is clearly important for finding the answer, as 89.8 % (114/127) of the questions with no determinable focus failed to find an answer. Finding the correct lexical answer clues aids in finding the correct question focus. Special question processing helps, but needs improvement.

Since the majority of the questions with a correct focus (188/322 = 58.4 %) did not retrieve an answer, we need to examine this finding in more detail.

5.4 Effects of query expansion

As discussed in section 3.1, we used WordNet 1.6 to expand nouns and verbs in the questions this year. Experiments using the TREC9 questions showed that the expansion helped find more relevant paragraphs, but whether it helped in locating the final answer within those paragraphs was not investigated. Query terms added from WordNet were found in 109 out of 160 (68 %) questions with correct answers in our paragraph run SUT10PARMT.

Query expansion had an additional, positive, impact. It actually provided correct answers for some short queries. For the question "*What does the acronym NASA stand for?* ", the phrase "National Aeronautics and Space Administration", was added to the L2L representation. This feature has been used in our procedure for identifying answer candidates for some question types.

5.5 Document tagging versus paragraph tagging

Contrary to our expectation, the "DOC" run (see section 3.3) did not achieve better performance, but did worse than the "PAR" run (see Table 3). This held true for both the main task and the list task. Following is the comparison of the two runs for main task (see Table 10).

[3] This analysis is based solely on the special question types identified as such; there were a total of 151 special questions.

RunID	# of correct answer	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
Document tagging (SUT10DOCMT)	111	51	22	20	8	10
Paragraph tagging (SUT10PARMT)	160	78	31	18	19	14

Table 10. Comparison of correct answers found by document run and paragraph run.

We hypothesized that the low performance of the document run might be caused by a lack of system testing due to time constraints. The analysis provided a good opportunity to find system bugs as well as evaluate our approach. We noticed that in most cases where the two runs got the right answers at the same rank, they found the answers in different documents or different paragraphs. A careful examination of the results for the first 103 questions (questions 894 to 996) demonstrated that the following sources contributed to the poor performance of the "DOC" run and the difference between the two runs:

1. A bug in the paragraph finding program truncated some documents when they were split into paragraphs (see section 3.3). The impact of this bug was minor.
2. The keyword weighting strategy used for paragraph finding inadvertently differed slightly between runs, which led to different scores even when the same answer strings were found. The influence of this difference was minor because it did not cause a change in rank.
3. The sentence alignment procedure truncated part of the texts for some documents. The word alignment procedure occasionally failed to record some of the keywords in the paragraph window, which threw out some paragraphs with the correct answers and dramatically changed the weighting score for some answer candidates. The alignment problems were the major cause of the low performance of the document run and the difference between the two runs.
4. The size of the paragraphs also played a role in making the two runs different. In the "PAR" run, we identified paragraphs according to text indentation while the "DOC" run uses a predefined value (400 bytes) to group sentences into paragraphs. Normally the sentence groups are longer than the natural paragraphs. The difference in length changed the position of paragraph windows and led to different scores for the same candidates.

After fixing the bugs and adjusting the alignment procedures (sources 1 and 3), we ran the "DOC" run again and achieved comparable results between the two runs. For the first 103 questions, both runs found correct answers for 30 questions out of which 23 were identical.

We also compared the tagging and categorization between complete documents and individual paragraphs. No difference between the two was found in this analysis. It might be that the TREC10 questions did not bring out the need for context information in tagging. This issue will need further investigation. Ultimately we need to decide between these two approaches.

5.6 List task evaluation

Both list task runs (SUT10PARLT and SUT10DOCLT) are based on the same question processing output. The list task analysis examines the performance of the answer instance identification as well as the reasons for the large performance difference between main and list tasks.

A special feature was added to our question processing module this year to handle the extraction of the number of desired instances from the list questions. Analysis revealed that for 22 (88%) of the questions, the number of instances was determined correctly. For three questions the program could not determine the correct number of instances so it defaulted to 2 instances (enough instances to make up a list). Out of the 22 questions that provided the right number of instances none of the questions managed to get all of the desired answers correct.

The system seemed to perform better on the list task than on the main task (see Tables 3 and 4). For the SUT10PARLT run only 20% of the questions could not be answered versus 67.5% in the main task counterpart run (SUT10PARMT). In observing the questions themselves it appears that the list task questions are more straightforward compared to the more complicated main task questions where 151 questions required more advanced linguistic pattern searches. The fact that the questions seem to be easier is reflected in the performance of the focus assignment module for the list task. Out of 25 list questions, 13 questions had a correct focus assignment, 3 questions had a wrong focus assignment, and for 9 questions the system correctly indicated that the focus was unknown. For the list task 88 % of the questions had a correct focus assignment versus 78% questions in the main task. Two out of the three questions with the wrong focus assignment were of identical form (*Name n people who/from …*) and both indicated the answer should be a number instead of a person. The error is due to a clue in the focus program dealing with *how many people* questions.

6. Conclusions and future research

The expansion of our question processing module clearly improved the accuracy of our focus assignment although there are still a large number of questions for which the system did not provide the correct answer. It appears that tagging the entire document before splitting it into paragraphs versus splitting it into paragraphs before tagging does not make a lot of difference. The decision on what tagging approach to take will depend on processing speed.

After the TREC10 experiments it is clear that a lot of work remains to be done. Our analysis shows that a one-size-fits-all approach to answer-finding does not work well. The system needs alternative answer-finding strategies for different question types and the means to differentiate between these question types. These different strategies also imply more advanced weighting schemes than are currently implemented. Our work on answer confidence level assignment needs to be completed and refined. The confidence level work will also include the ability to decide whether an answer can indeed by provided. In addition the system also needs to be adapted to deal with the context specific task (the third TREC Q&A track task) where each answer provides contextual information to help answering the next (related) question.

References

[1] <!metaMarker>^{TM} . http:// http://www.solutions-united.com/products_information.html

[2] Brill, Eric. (1995). Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging. Computational Linguistics. Available at: <http://www.cs.jhu.edu/~brill/CompLing95.ps>.

[3] Collins, Michael. (1996). A New Statistical Parser based on Bigram Lexical Dependencies. Proceedings of the 34th Annual Meeting of the ACL, Santa Cruz. Available at <http://www.research.att.com/~mcollins/papers/acl9629.ps>.

[4] Collins, Michael. (1997). Three Generative, Lexicalised Models for Statistical Parsing. Proceedings of the 35th Annual Meeting of the ACL (jointly with the 8th Conference of the EACL), Madrid. Available at <http://www.research.att.com/~mcollins/papers/paper14.short.ps>.

[5] Diekema, Anne; Liu, Xiaoyong; Chen, Jiangping; Wang, Hudong; McCracken, Nancy; Yilmazel, Ozgur; and Liddy, Elizabeth D. (2000). Question Answering: CNLP at the TREC-9 Question Answering Track. Available at: <http://trec.nist.gov/pubs/trec9/papers/cnlptrec9.pdf>.

[6] Dimmick, D., O'Brien, G., Over, P., and Rodgers, W., Guide to z39.50/prise 2.0: Its installation, use & modification, 1998. http://www-nlpir.nist.gov/works/papers/zp2/zp2.html

[7] HowStuffWorks. http://www.howstuffworks.com/.

[8] Miller, G. (1990). WordNet: An On-line Lexical Database. *International Journal of Lexicography*,.Vol. 3, no. 4. Special Issue.

[9] Salton, G. Ed. (1971) *The SMART Retrieval System: Experiments in Automatic Document Processing.* Prentice-Hall, Inc. Englewood Cliffs, NJ. 556p.

[10] Voorhees, Ellen M.; Tice, Dawn M. (1999). The TREC-8 Question Answering Track Evaluation. In: E.M. Voorhees and D.K. Harman (Eds.) *The Eighth Text REtrieval Conference (TREC-8).* 1999, November 17-19; National Institute of Standards and Technology (NIST), Gaithersburg, MD.

Tampere University of Technology at TREC 2001

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

Abstract

In this paper we present the prototype based text matching methodology used in the Routing Sub-Task of TREC 2001 Filtering Track. The methodology examines texts on word and sentence levels. On the word level the methodology is based on word coding and transforming the codes into histograms by the means of Weibull distribution. On the sentence level the word coding is done in a similar manner as on the word level. But instead of making histograms we use a more simple method. After the word coding, we transform the sentence vectors to sentence feature vectors using Slant transform. The paper includes also description of the TREC runs and some discussion about the results.

1 Introduction

A common approach to topic detection and tracking is the usage of keywords, especially, in context of Dewey Decimal Classification [3, 2] 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 [6]. Traditionally, information retrieval has roughly been based on a fixed list of index terms [6, 5], or vector space models [10, 9]. 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 [7] but they are computationally heavy.

In this paper, we present our methodology and concentrate on tests of content based topic classification, which is highly attractive in text mining. The evolution of the methodology has been earlier

This research is supported by TEKES, the National Technology Agency of Finland (grant number 40943/99). The support is gratefully acknowledged.

discussed in several publications [11, 12, 13]. In the second chapter the applied methodology is described. In the third chapter the experiments with the Reuters database are described. The execution times are presented in chapter four. Finally, the methodology and the results are discussed.

2 Methodology

The methodology we applied to the TREC 2001 Routing runs is a multilevel one. It examines the contents of text documents on word and sentence levels.

The process starts with preprocessing of the training set text. This includes omitting extra spaces and carriage returns, and separating single words with single spaces. With the Reuters database, the preprocessing also includes the removal of the XML tags. The filtered text is next 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. Or, 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.

We selected the last alternative, because it is accurate and suitable for statistical analysis. A word w is 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: if the word is "**c a t**", then

$$y = k^2 * ascii(c) + k * ascii(a) + ascii(t) \tag{2}$$

The encoding algorithm produces a unique code number for each different word. After each word has been converted to a code number, we consider the distribution of these numbers and try to estimate their statistical distribution. Many distributions, e.g. Gamma distribution, are suitable for this purpose. However, it would be advantageous, if the selected distribution had only few parameters and it matched the observed distribution as well as possible. Based on tests with different types of text databases, we selected the Weibull distribution to estimate the distribution of the code numbers.

In the training phase the range from the logarithm of the minimum value to the logarithm of the maximum value of code numbers is examined. This range is divided to N_w equal bins. Next, the frequency count of words belonging to each bin is calculated. The bins' counts are normalized with the number of all words. Then the best Weibull distribution corresponding to the data is determined. Weibull distribution is compared with empirical distribution by examining both distributions' cumulative distributions. Weibull's Cumulative Distribution Function is calculated by:

$$CDF = 1 - e^{(((-2.6*log(y/y_{max}))^b)*a)} \tag{3}$$

There are two parameters that can be varied in Weibull's CDF formula: a and b. A set of Weibull distributions are calculated with all the possible combinations of a's and b's using a selected precision.

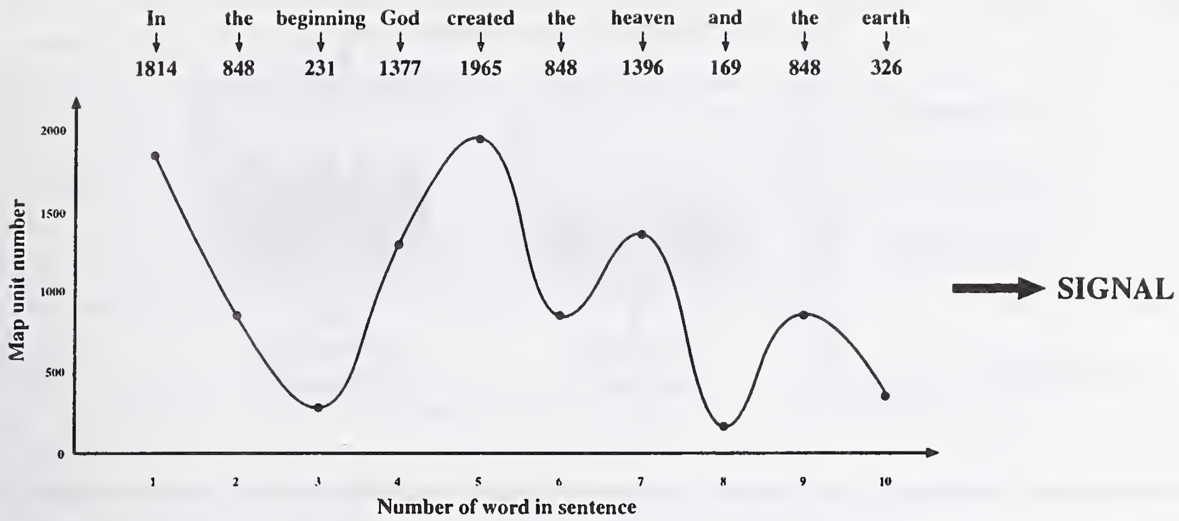


Figure 1. The transform from a sentence to a signal.

The possible values for the coefficients are restricted between realistic minimum and maximum values. The empirical cumulative distribution and Weibull's cumulative distribution are compared in the least square sum sense.

The best Weibull distribution is divided into N_w equal probable bins. Every word belongs now to a bin that can be found using the word code number and the best fitting Weibull distribution. Using this type of quantization the word can now be presented as the bin number, i.e. the number of the bin that it belongs to. Due to the selected coding method the resolution will be the best where the words are most typical to text (usually words with 2-5 characters). Rare words (usually long words) are not so accurately separated from each other.

Similarly at the sentence level every sentence has to be converted to numbers. Every word in a sentence is now replaced with a bin number. The bin number is generated with the method described earlier. Example of encoding a sentence :

I	**have**	**a**	**cat**	**.**
bn_0	bn_1	bn_2	bn_3	bn_4

where bn_i = bin number of the word i. Note, that in the encoding also the punctuation marks are encoded. When the sentence is encoded, it can be considered as a sampled signal. To illustrate this way of thinking, an example sentence and it's encoded form are presented in figure 1.

We use Slant transform matrix for transforming the sentence signals. Slant transform coding is commonly used in image processing. In detail Slant transform is explained e.g. in publications [1, 4, 8]. The size of the Slant matrix is, based on experiments, selected to be 32*32. If the sentences length is over 32 words, the rest of the sentence after 32 words is not considered. If the sentence is shorter than 32 words, then the missing words get zero as value.

First row of Slant matrix is multiplied with sentence vector and the result is stored to S_0. Second row

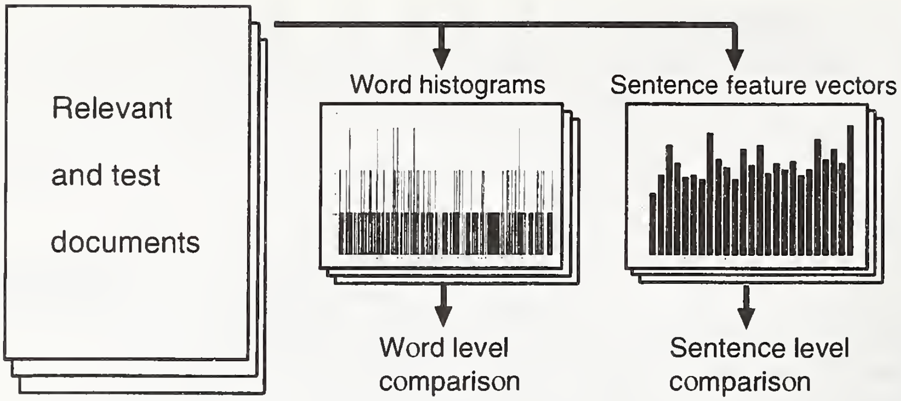


Figure 2. Process of comparing and analyzing documents based on extracted histograms and feature vectors.

multiplication with sentence vector is stored to S_1. I.e.

$$[Slant\ matrix] * [Sentence\ vector]^T \tag{4}$$

creates the result vector of the sentence $[S_0, S_1, S_2 ... S_{31}]$. Now there are 32 real numbers in a vector that describe the sentence. The numbers can be negative or positive.

This method have to go through all the sentence vectors of the text and the result is summed up with previous result. After every sentence of the text is transformed with Slant transform and then normalized, then we have 32-dimensional feature vector that describes the text.

When examining the test set documents on the word level, we create histograms of the relevant and test set documents' word code numbers. The filtered text from a single document is encoded word by word. Each word number is quantized using word quantization created with all the words of the training set. 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 consisting of N_w bins is finally normalized by the length of the histogram vector. On the sentence level the relevant and test documents are encoded to straight to feature vectors as described earlier.

With the histograms and feature vectors derived from all the relevant and test documents in the database it is possible to compare and analyze the test documents' text on the word and sentence levels against the relevant texts. The histogram and vector creation and comparison processes are illustrated in Figure 2. Note, that it is not necessary to have any prior knowledge of the actual text documents to use this methodology. The training set words define the distribution formula that is used with the quantization of words. This information is transferred to the sentence level, where the relations and order of the words are taken into account. No linguistic methods are used in the process.

3 Runs with Reuters database

For TREC 2001 we did two runs, one on the word level (VisaWordT10) and one on the sentence level (VisaSentT10). This is our first time in the TREC conference, so we were left with very little time for doing the actual runs. This led to simplifying the training and testing processes.

In the runs for TREC, 2080 was selected for bin number N_w. The amount of bins was selected on the grounds of earlier experiments with English text databases. On the word level run, every test word histogram is compared with randomly chosen 969 relevant document histograms. The number of relevant documents is reduced from 15261 to 969 because it speeds up the test comparison time. Otherwise, the computing time would have risen unreasonable high. To all 84 topics 13 relevant documents have been chosen by random. If a topic has less than 13 relevant documents the number of chosen relevant documents is the amount that exist in that topic. Euclidean distance metric is used in comparing word histograms. The topic of the test document becomes the topic of the relevant document which have the smallest Euclidean distance with the test document. Only the best match relevant document is considered.

On the sentence level run, feature vector of every sentence is compared with all 15261 feature vectors of relevant documents. Since the sentence feature vector is only 32-dimensional, the comparisons are very fast. Comparison was done using Euclidean distance between the sentence vectors. Distances to every topic are calculated for each test document. On the sentence level, all the distances between the test document and the relevant documents are taken into consideration.

On both runs, top thousand test documents are chosen from every topic as the final result. The actual value of the distances are not take into consideration, there are no thresholds for belonging to a topic.

4 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 (word level) execution time include finding the best Weibull distribution and creating the word histograms for test documents. Making feature vectors (sentence level) execution time consists of the encoding of the sentences and creation of the 32-dimensional feature vectors. The comparing execution times are the times that it took to compare the test histograms and vectors with the relevant documents' histograms and vectors. The computer used in the runs was a PC with a Intel® 550 MHz Pentium® III processor and 128 Mb of memory. The operating system was Linux.

Table 1. Execution times rounded up to the nearest hour.

	Making histograms/ feature vectors	Comparing	Altogether
Word level	3 h	30 h	33 h
Sentence level	4 h	26 h	30 h

5 Conclusions

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 few days period of a single month, the vocabulary in the relevant documents seemed not to vary enough. The type of methodology we used requires a good set of representative word and sentence samples from the whole database. The training set vocabulary was restricted in the sense of yearly cycle, to one month in autumn of 1996. This type of difficulties are, on the other hand, very common in real life tasks.

More precise problematic issues using the methodology include the selection of quantization method and distance metrics. Using the word quantization method the way presented here has also some disadvantages. Some accuracy is lost when the word codes are quantized with the Weibull distribution. Words that are quite different may get the same value in the quantization, since strict division of the distribution is used to quantize the codes. Perhaps this could be prevented by using a quantization method that would use the code number values to create a more natural classification. Such a method would be very simple and would perhaps create a more precise and truthful quantization of words. Selection of distance metrics is perhaps even more problematic area. Euclidean distance, which was used here, seems not to be taking into account the shape of the histograms and feature vectors. Euclidean distance compares just the values that are in the same place in two vectors. A distance metric that would allow more variation in the position of values in the histograms would perhaps be more suitable. This kind of metrics are for example Levenshtein metrics or some metrics utilizing the angle between vectors.

Our methodology seemed to work well only in few topics. However, the methodology was competitive with other competitors on some topics. The runs were designed so that only a very basic form of the methodology was used. The methods used are very fast and it seems that the speed in these runs was gained at the cost of accuracy. It should be noted, that the methodology uses no external aids in the process. Dictionaries, stemming algorithms, transformation to base form, or linguistic information were not used. The methodology does not depend on the language. The comparisons can be performed as long as the training data and the test data are written in the same language.

Future improvements in the methodology include finding a way to balance the effects of different amounts of relevant documents in topics. One of our future aims is to more efficiently take into account the advantage from the given relevancy information. This would hopefully help to create a specific view of the topic, i.e. the knowledge of the words and sentences that separate the topics. Also the size of the word histograms and the sentence feature vectors should be more properly optimized for the amount and type of the text in the database.

References

[1] N. Ahmed and K. Rao. *Orthogonal Transforms for Digital Signal Processing.* Springer Verlag, 1975.

[2] 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.

[3] 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.

[4] A. Jain. *Fundamentals of Digital Image Processing.* Prentice Hall, 1989.

[5] 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.

[6] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing.* The MIT Press, Cambridge, Massachusetts, 1999.

[7] D. W. Oard and G. Marchionini. A conceptual framework for text filtering. Technical Report CS-TR3643, University of Maryland, May 1996.

[8] W. K. Pratt, L. R. Welch, and W. H. Chen. Slant transform image coding. *IEEE Trans. on Communications*, 22:1075–1093, August 1974.

[9] G. Salton. *Automatic Text Processing.* Addison-Wesley, 1989.

[10] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[11] A. Visa, J. Toivonen, S. Autio, J. Mäkinen, H. Vanharanta, and B. Back. Data Mining of Text as a Tool in Authorship Attribution. In B. V. Dasarathy, editor, *Proceedings of AeroSense 2001, SPIE 15th Annual International Symposium on Aerospace/Defense Sensing, Simulation and Controls. Data Mining and Knowledge Discovery: Theory, Tools, and Technology III*, volume 4384, Orlando, Florida, USA, April 16–20 2001.

[12] A. Visa, J. Toivonen, B. Back, and H. Vanharanta. Improvements on a Knowledge Discovery Methodology for Text Documents. In *Proceedings of SSGRR 2000 – International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet*, L'Aquila, Rome, Italy, July 31–August 6 2000. (CD–ROM).

[13] A. Visa, J. Toivonen, H. Vanharanta, and B. Back. Prototype Matching – Finding Meaning in the Books of the Bible. In J. Ralph H. Sprague, editor, *Proceedings of the Thirty-Fourth Annual Hawaii International Conference on System Sciences (HICSS-34)*, Maui, Hawaii, USA, January 3–6 2001. (CD–ROM).

Using Grammatical Relations, Answer Frequencies and the World Wide Web for TREC Question Answering

Sabine Buchholz*
ILK, Tilburg University
P.O. Box 90153, 5000 LE Tilburg
The Netherlands
s.buchholz@kub.nl

Abstract

This year, we participated for the first time in TREC, and entered two runs for the main task of the TREC 2001 question answering track. Both runs use a simple baseline component implemented especially for TREC, and a high-level NLP component (called Shapaqa) that uses various NLP tools developed earlier by our group. Shapaqa imposes many linguistic constraints on potential answers strings which results in not so many answers being found but those that are found have a reasonably high precision. The difference between the two runs is that the first applies Shapaqa to the TREC document collection only, whereas the second one also uses it on the World Wide Web (WWW). Answers found there are then mapped back to the TREC collection. The first run achieved a MRR of 0.122 under the strict evaluation (and 0.128 lenient), the second one 0.210 (0.234). We argue that the better performance is due to the much larger number of documents that Shapaqa-WWW's answers are based on.

1 Introduction

For the TREC-2001 question answering (QA) main track, systems automatically had to answer 500[1] open-domain fact-based questions on the basis of nearly one million news articles from the TREC collection. Up to five ranked 50-byte answer strings could be returned for each question. Unlike in previous years, there was no guarantee that an answer was contained in the collection and indeed, 49 questions have no known correct answer in this collection. Systems could return "NIL" as one of the five answer strings if they "thought" that there was no answer. Human assessors judged each answer string. For each question, a system received a score that is reciprocal to the rank at which the first correct answer was found. The final score is then the mean of all question scores (mean reciprocal rank, MRR).

This paper presents the system that we used for the main task of the TREC-2001 question answering (QA) track. It differs from most of the other systems used in the QA track in the past years in three aspects. Firstly, it does not use a Named Entity recognition component. For example, it does not make a difference between persons, organizations and other Named Entities. Instead, the answer finding approach is centered on the main verb of the question. It relies on analyzing grammatical relations between this verb and the other parts of the question and answer. Secondly, it does not use complicated document, paragraph, sentence or string weights. The baseline component only counts the number of keywords in a sentence, and the Shapaqa component counts the frequency with which a given answer occurs. Thirdly and most importantly, it uses the World Wide Web as an additional resource.

*This research was done in the context of the "Induction of Linguistic Knowledge" research programme, which is funded by the Dutch Organization for Scientific Research (NWO).

[1] Eight questions were later removed from the evaluation.

The next section describes the baseline and the TREC and WWW versions of the Shapaqa component in detail. Section 3 shows how these components were combined in the two submitted runs. Section 4 analyses the system's errors and Section 5 gives a short summary.

2 System description

Our system does not have its own Information Retrieval engine. It uses the top ranked 1000 documents per question from the list provided by NIST.

There are two components: One high-level NLP component called Shapaqa[2] that has reasonable precision on the answer strings it returns but that often does not return any answer string, and one simple keyword matching component that has low precision but nearly always returns some (possibly incorrect) answer string. The latter provides us with a baseline performance. The Shapaqa component and the baseline component are integrated as follows: If Shapaqa returns at least one answer string, its top answer string is taken as the top ranked answer string of the combined system. All the remaining ranks are filled by the top answer strings returned by the baseline system. We describe the two components separately in the following two sections.

2.1 Baseline

For the baseline component, the question is tokenized and then part-of-speech tagged (Daelemans et al., 1996). Then all those words are extracted as keywords that the POS tagger did not know (regardless of what tag it assigned) or that it tagged as noun, non-modal verb, adjective, particle number or foreign word except the words "much", "many", "name" and forms of "be", "have" and "do". A special case are non-subject questions with a form of the auxiliary *do* and following infinitive, e.g. "When *did* Elvis Presley *die*?". English grammar tells us that in a declarative sentence (i.e. the answer) it will be the main verb that carries the inflection: "Elvis Presley *died* in 1977". Therefore if the question contains "did" or "does", all following infinitival verb keywords are replaced by their past tense or third person singular present tense form, respectively. This rule applied to 22 and 14 questions, respectively, and uses the CELEX lexical database (Baayen, Piepenbrock, and van Rijn, 1993).

After keywords have been extracted, all top 1000 documents for each question (according to the list provided by NIST) are tokenized, and all sentences are extracted which contain at least one keyword for that question. Multiple occurrences of the same keyword are only counted once. The document ID and the number and position of the keywords are stored together with each extracted sentence. This yields 1,233,692 sentences. For two questions no sentences could be extracted.[3] The sentences for each question are then sorted according to the number of keywords in them.

The top sentences of the sorted list are taken as the baseline component's answers. To trim them to 50 bytes we compute that byte position in the sentence that is at the center of all the keyword positions, and then take 50 bytes around it (possibly shifting the 50 byte window as far as necessary to the right or left for it not to extend beyond the sentence boudary).

As an example, consider the question "When was President Kennedy shot?". Keywords are *President Kennedy* and *shot*. 5928 sentences are extracted. The topmost chosen answer sentence and its 50-byte string are "In 1963, he was riding in the motorcade wi*th President Kennedy when Kennedy was fatally shot* by Lee Harvey Oswald in Dallas.", in which the answer falls outside the chosen 50 bytes.

On the non-variant questions of the TREC-9 data, taking the full top five sentences of the sorted list as answers yielded an MRR of 0.321 using the automatic evaluation. After the sentences were trimmed to 50 bytes with the above method, MRR dropped to 0.125. This is hardly surprising, given that the method is completely insensitive to the type of answer the question is asking for (it

[2] see also (Buchholz and Daelemans, 2001a), (Buchholz and Daelemans, 2001b)

[3] "What is pilates?" and "What is dianetics?". Keyword matching is sensitive to capitalization and both terms appear in the documents only with capital letters.

returns the same 50 byte window, whether the question word is “who”, “where”, or “when” etc.) and that the complete answer sentences were on average 293 bytes long.

2.2 Shapaqa

2.2.1 Question Analysis

For the Shapaqa component, the POS tagged question is further chunked and grammatical relations between verb chunks and other chunks are determined (Buchholz, Veenstra, and Daelemans, 1999). Possible relations are subject (SBJ), object (OBJ), logical subject of passive verbs (LGS), locative adjunct (LOC), temporal adjunct (TMP), adjunct of purpose and reason (PRP), manner adjunct (MNR), and an unspecified relation between a verb and a prepositional chunk (OTH).[4] In the latter case, the preposition is stored as additional information. The relation finder does not work well on questions, especially in the first part, which shows the characteristic question syntax. This is probably due to the small number of direct questions in the training material. Therefore a set of hand-made regular expressions and substitutions (developed on the TREC-9 data) are applied to the question after parsing to fix the most common errors. For our previous example, the result would be “[$_{ADVP-TMP-1}$ When] was [$_{NP-SBJ-1}$ President Kennedy] [$_{VP-1}$ shot] ?”.

Next, the parsed question is transformed into Shapaqa’s internal format. Prepositions stranded at the end of the question are rejoined with their NP. Passive is converted to active. The head (i.e. the last word) of the first verb chunk is the central verb (auxiliaries in inverted questions should not be a verb chunk of their own). All the chunks that are directly dependent on it are stored together with their relation. Chunks that are not directly dependent on the central verb are concatenated to the next chunk to the left that is, because they are probably dependent on that chunk and thus, as relations seldom cross, belong to the same phrase. In this way the question is split up into phrases that all have some relation to the central verb. The phrase that contains the wh-word is replaced by a special marker (“?”). Our example would then look like: VERB="shot" OBJ="President Kennedy" TMP="?"

Questions asking for “Which/What X” are mapped as if they were simple “Who/What” questions. The phrase “In which state” was mapped as a simple “Where” because the relation finder assigned it a locative relation. In total, 60 “Which/What X” questions could be mapped. Questions starting with “Name a X” where mapped as if they read “What is a X?” (which is an ad-hoc solution which did not work: both questions were incorrectly answered). Some other minor simplifications are also performed during mapping.

In total, 416 questions could be converted to Shapaqa’s internal format.

2.2.2 Answer extraction

Once the central verb and all its related phrases are in Shapaqa’s internal format, they can be matched against parsed sentences extracted from the documents. As parsing is time-consuming, and Shapaqa phrases normally include all keywords (see Section 2.1), only those sentences are POS tagged, chunked and assigned relations that contain all the question’s keywords (in total 44,753 sentences).

A parsed sentence matches the information in Shapaqa’s internal format if the central verb matches with a verb chunk’s head in the sentence, if all the phrases match literally somewhere in the sentence (the special marker “?” matches any chunk) and if each phrase’s relation in the internal format matches the relation assigned to the chunk that contains the beginning of this phrase in the sentence (recall that a phrase’s relation in the question is also determined by its first chunk’s relation). Two relations match if they are identical, or if they form a special pair with special conditions . Special pairs are SBJ/OBJ where the former occurs in a passive question or sentence and the latter in an active one, LGS/SBJ under the same condition, and SBJ/OBJ if the verb is a form of “to be” (to match: “Who is X?” with “X is the president of Y”)

[4] The relations are based on the functional tags in the Wall Street Journal corpus of the Penn Treebank II (Marcus et al., 1994) on which the tagger, chunker and relation finder are trained.

If a sentence matches the internal format, we extract the chunk that matched the special marker "?" as an answer unless it is any of a fixed number of semantically empty answers like "he/who/somebody" etc.[5]

We isolate the head word of the answer chunk, i.e. its last word, and update a frequency counter for it. After all answer chunks have been extracted, they are sorted according to the frequency with which their head word was found as head of an answer .

2.3 Shapaqa-TREC versus Shapaqa-WWW

We used two versions of Shapaqa in our system. Shapaqa-TREC extracts answers directly from the TREC document collection in the way described above. One of the answer chunks whose head has the highest frequency is taken as the answer of the Shapaqa component. If necessary the answer is trimmed to 50 bytes by cutting off the end. For our example, the answer was extracted from the sentence: "President Kennedy was shot *on Nov. 22, 1963.*" The head word was found three times as head of an answer.

Whereas Shapaqa-TREC extracts answers directly from the TREC document collection, Shapaqa-WWW first extracts answers from the World Wide Web. The central verb and the phrases of the question in Shapaqa's internal format are used as search terms for Google.[6] Google returned results for 380 of the 416 questions that could be converted to Shapaqa's internal format. Shapaqa-WWW then searches for answers in Google's top 1000 text snippets. There, it found answers for 283 questions (7936 answers in total). For some questions, the only answers found are semantically empty ones like "he/who/somebody" etc., which are discarded. This leaves us with answers for 265 questions, which is slighty more than half of all TREC-10 questions. As explained above, answers are sorted according to the frequency of their head word. The most frequent head word is taken to be the preliminary answer. For our example, the head word *1963* was found twelve times.

To turn a preliminary answer from the WWW into a valid TREC answer string, we have to find a document in the TREC collection that contains the head word. To increase the chance that the document actually supports the answer (as is necessary to be judged correct under the strict evaluation), we look for the head word in the sorted sentences extracted for the question, starting with the ones that contain most of the question keywords. Head words could be found in these sentences for 226 questions. After a sentence is found, a 50 byte piece of it centered around the head word (unless shifted to meet sentence boundaries) is extracted as Shapaqa-WWW's answer string. For our example, answer sentence and string are: "Ruby shot Oswald to death with the .38-caliber Colt Cobra revolver in the basement of Dallas *City Jail on Nov. 24, 1963, two days after Presi*dent Kennedy was assassinated." which unfortunately makes the answer invalid.

3 Runs: TilburgILKs and TilburgILK

We submitted two runs for the TREC QA track main task. Run TilburgILKs uses Shapaqa-WWW's answer (if present) on the first rank, Shapaqa-TREC's answer (if present) on the next highest rank, and the baseline component's answer on all other ranks. If not enough answers can be found to fill the five ranks, NIL (meaning that no answer exists in the document collection) is added.[7] We did not use the option to specify a "final answer" other than rank one.

TilburgILKs received a MRR of 0.210 (0.234 lenient). Run TilburgILK uses only Shapaqa-TREC (for the first rank) and the baseline component. Its MRR is 0.122 (0.128 lenient).

[5] Obviously this does not work for questions whose answer is not a chunk but a clause (in that case only the complementizer chunk would be extracted).

[6] http://www.google.com. A prototype of Shapaqa-WWW without the question parser functionality is online at http://ilk.kub.nl/shapaqa/. Unfortunately, the version that was used for the experiments described in this paper, which was based on the search engine Google, is not operable anymore. Instead a version is implemented that uses the search engine AltaVista (http://www.altavista.com/). This variant is much slower because AltaVista does not return text snippets containing the keywords, so Shapaqa has to retrieve and search the full documents.

[7] Due to a bug, the NIL answer was not added at the lowest rank, as intended, but at the highest.

rank of first correct	1	2	3	4	5	none
TilburgILK	43	15	15	7	14	398
TilburgILKs	87	19	9	7	11	359

Table 1: Distribution of rank of first correct answer in both submitted runs (strict evaluation)

components used	TilburgILKs	after bug fix
none (only answer NIL given)	3	3
only baseline	267	263
Shapaqa-TREC and baseline	4	8
Shapaqa-WWW and baseline	195	159
Shapaqa-WWW, Shapaqa-TREC and baseline	31	67

Table 2: Number of questions for which different components contributed to the five answer strings.

When writing this paper, we unfortunately noticed a serious bug in Shapaqa-TREC. The SBJ/OBJ special pair (see Section 2.2.2) was omitted in this implementation.[8] This means that for example for the frequent question type "What is a Y?" which asks for a definition or description, only sentences matched which read "X is a Y" and not those that read "A Y is X" (which is the way to formulate a definition). This led to many erroneous answers like "What is a prism?" – "Serbian agression", extracted from a sentence that reads "Serbian aggression is a prism through which we can see all sides of Europe."

As 254 of the questions that could be mapped to Shapaqa's internal format have a form where the special pair could have been applied, the omission might have influenced Shapaqa-TREC's performance significantly. To find out how serious this effect is, we reran the TilburgILK run after fixing the bug and studied the differences. It turned out that only 50 questions were affected by the bug fix, so we compared these manually. In most of the cases, the new answer was as bad as the old one, so the score did not change. We estimate that the new version's MRR would be about .004 higher than the original one's, which seems neglectable.

Table 1 displays the distribution of the rank of the first correct answer in both runs and shows that TilburgILKs has twice as many correct answers at the first rank than TilburgILK.

Table 2 shows how often each of the three components contributed to the five answers for the TilburgILKs run before and after the bug fix. We see that even after the bug fix, Shapaqa-WWW was applied nearly three times as often as Shapaqa-TREC. This is probably due to the much larger document collection that Shapaqa-WWW works on.

Table 3 shows how the assessors judged each component's answer strings. We compute the *precision* of a component as the percentage of correct answers among all the answers it contributed. We see that Shapaqa-WWW has a higher precision than Shapaqa-TREC, especially after the bug fix. Both versions of Shapaqa have a much higher precision than the simple baseline component. Precision of the baseline component's answer is slightly higher for its top ranked answer than for the lower ones, but in general there does not seem to be a clear correlation between its answers' ranks and their reliability. Shapaqa-WWW has more unsupported answers than the other components. This is clearly due to the mapping from WWW-answers to TREC documents.

Table 4 gives a breakdown of Shapaqa-WWW's precision on different types of questions. The first column shows the question type according to the module that maps questions to Shapaqa's internal format. The second column indicates how many questions of this type could successfully be converted to Shapaqa's internal format. The third columns shows for how many questions Shapaqa-WWW returned an answer string. The fourth and fifth columns gives the precision on these strings (strict and lenient). "When"-questions do best, followed by those with the wh-word inside a prepositional phrase.

[8] Shapaqa-WWW is implented in PHP and runs on our webserver, whereas Shapaqa-TREC is in Perl and runs on a machine whose hard disk can hold all the TREC documents for the QA track. However, both systems access the same tagger, chunker and relation finder through socket connections.

judgement	baseline 1st	2nd	3rd	S-TREC	S-TREC*	S-WWW
incorrect	462	473	469	27	63	144
correct	34	22	26	8	12	71
unsupported	1	1	1	0	0	11
prec. strict	6.8	4.4	5.2	22.9	16.0	31.4
prec. lenient	7.0	4.6	5.4	22.9	16.0	36.3

Table 3: Judgements (under strict evaluation) and precision of answers (strict and lenient) of each component. Baseline 1st, 2nd and 3rd means the baseline's first, second and third answer. S-TREC is Shapaqa-TREC, S-TREC* is Shapaqa-TREC after the bug fix, S-WWW is Shapaqa-WWW.

Wh-type	# of questions	# of q. answered	prec. strict	prec. lenient
who	44	21	57.1	57.1
what	237	136	25	30.1
which X	5	2	0	0
what X	55	27	18.5	25.9
where	24	12	25	25
when	24	18	66.7	72.2
why	2	0	0	0
how	7	1	0	0
PP	16	9	55.6	66.7
Name a	2	0	0	0

Table 4: Precision of Shapaqa-WWW on different wh-types.

Shapaqa relies on high-level NLP (chunking, grammatical relations) for finding answers. If it finds more than one answer however, it uses the frequencies of the answers' head words to chose among the answers. The idea is that frequency correlates roughly with reliability of the answer. However, this can only work if the document collection from which answers are extracted is large enough. On average, Shapaqa-TREC (after the bug fix) finds 2.55 different answers (where "different" means having a different head word) for the questions it finds answers for at all. The average frequency of the most frequent answer for each question is 1.44, but the distribution is highly skewed in that most top answers have only frequency one, and only one top answer has a frequency higher than ten. By contrast, Shapaqa-WWW finds 17.0 different preliminary answers per question, and the average frequency of the most frequent answers is 26.7.

To further study the correlation between frequency and reliability, we divided the 265 questions for which Shapaqa-WWW found an preliminary answer into a high-frequency and a low-frequency group according to the frequency of the top answer (greater, or less or equal than 7). The precision of Shapaqa-WWW's answers for questions in the high-frequency group is 36.1% (39.3%) whereas it is only 26.0% (32.7%) for the low-frequency ones.

This shows that answers which we find often are more reliable than those with little evidence. As Shapaqa-WWW searches on a much larger document collection than Shapaqa-TREC, it can take advantage of this fact. This effect even holds despite the very simplistic way of mapping the WWW answers back to the TREC collection in order to comply with the TREC guidelines.

4 Error analysis

In this section, we study the effect of several design decisions we made about the document selection, the transformation from natural language questions to Shapaqa's internal format and the mapping from WWW preliminary answers back to the TREC collection.

Our system does not search the whole document collection for answers but only the top 1000 documents per question (as provided by NIST). For 14 questions, some other systems found an answer in documents not in the top 1000, so some minor improvement should be possible through a better IR component.

Several things can go wrong during the transformation from natural language questions to Shapaqa's internal format. First, if no central verb can be found, the transformation is not possible. This happened with 14 questions. In three of these cases there really was no verb (e.g. "How many liters in a gallon?"), so the system would have needed a special rule to insert the verb "are". In one case the main verb "was" was not analyzed as a verb chunk. In the other ten cases, the main verb is analyzed as a noun. This problem is probably due to too few questions in the training material of the tagger.

Second, two chunks are assigned the same relation and there is no coordinating conjunction between them. This happened with 22 questions. Sometimes the chunks really have the same relation (e.g. "*In Poland*, *where* do most people live?"), sometimes the analysis is due to the relation finder's failure to distinguish between different object relations ("*What* do you call *a newborn kangaroo*?"), but most of the time the analysis is just plain wrong. Again, more questions in the training data could improve performance.

Third, some chunk has a direct relation to the central verb that does not fit any of the predefined categories (28 cases). These are mostly nouns mistagged as adverbs which then give rise to adverbial chunks being neither locative nor temporal, manner or purpose/reason. In some cases they are adjectival complements of "to be". The system needs to be extended to deal with these categories.

Fourth, questions with "How many/How much" or "How X" where X is some adjective cannot be converted (12 cases). Finally, there are some rare cases, like no relation between the *wh*-phrase and the central verb or failure to recognize the question phrase as such.

Shapaqa treats "which/what X" questions like simple "who/what" questions. To see in how far this influences performance, we manually checked the top frequency answers found on the WWW for 10 of these questions. In three cases the topmost answer looks okay. In two other cases at least the second answer is correct (e.g. "What river in the US is known as the Big Muddy?" – "The river"; "The Missouri"). For the remaining five questions, the missing constraint leads to wrong results ("Which president was unmarried?" – "The mother"). So ideally, we would want to use the extra information. However, we would then need a component that can e.g. decide whether something is a "mountain range in North America". Until that time, our solution is an approximation.

We conducted a similar survey to find out how harmful our approach of ignoring the difference between *who* and *what* is. A manual check on Shapaqa-WWW's answers for five questions of each type suggests that this simplification does not introduce many errors. This makes sense, given that questions like "Who/What discovered radium?" are very unlikely to have any "what" answers, and questions like "Who/What is Australia's national flower?" are unlikely to have any "who" answers. The only counterexample is "Who developed the Macintosh computer?", for which the assessors did not accept "Apple Computer Inc." or similar.

Clearly, the forced mapping from preliminary WWW answers to TREC documents is far from ideal. Sometimes Shapaqa-WWW finds the correct answer on the WWW but cannot map it. One reason is that no answer exists in the collection. This happened with "What is Australia's national flower?" – "The Golden Wattle". Alternatively, the answer that came up highest from the WWW does not exist in the collection but others do. This happened with "What is a shaman?" and the answer "a healer" (other systems found answers like "tribal magician" or "a kind of priest").

A problem for both versions of Shapaqa are question-answer-pairs like "What is mold?" – "Mold is a problem". This answers the question in letter but not in spirit. There is probably a limited number of abstract nouns that can occur in this contruction (another one is "solution") and explicitly excluding the most common ones might be an opportunistic solution. A more principled approach would probably need semantic knowledge.

"What is a panic disorder?" – "A panic disorder is a type of generalized anxiety disorder." Although this answer as a whole is okay, Shapaqa identifies "type" as the head of the (predicative) object, so it is this word that gets looked for in the TREC document sentences, which might or might not work. As in the previous case, there are probably only a limited number of nouns that cause this problem but a general solution needs semantic knowledge.

"What is epilepsy?" – "Epilepsy is a common neurological disorder." This answer is correct and the head is correctly identified, too. However, as the head is rather unspecific, the answer string that is finally extracted from the TREC documents ("from an inner-ear disorder that causes vertigo") makes the answer invalid. In contrast to the previous cases, this problem is entirely due to the forced mapping from WWW answers to TREC documents, so its solution is not of general interest. One might try to find a match not only for the head word but also for the other words in the chunk.

5 Summary

We described our approach to QA which combines a basic keyword matching component with a high-level NLP component that uses chunking and grammtical relations to impose many linguistically motivated constraints on what it extracts as an answer. This results in a much higher precision but less answers. We tackle the problem by searching for answers not only in the TREC document collection but also on the WWW. This results in answers to more questions and more reliability of the answers through the use of answer frequencies. Many aspects of the system can still be improved, but the achieved MRR of 0.210 (0.234) is certainly encouraging.

References

Baayen, R. H., R. Piepenbrock, and H. van Rijn. 1993. *The CELEX lexical data base on CD-ROM.* Linguistic Data Consortium, Philadelphia, PA.

Buchholz, Sabine and Walter Daelemans. 2001a. Complex answers: A case study using a www question answering system. *Journal of Natural Language Engineering*, Special issue on question answering. To appear.

Buchholz, Sabine and Walter Daelemans. 2001b. SHAPAQA: Shallow parsing for question answering on the world wide web. In *Proceedings of RANLP - 2001.*

Buchholz, Sabine, Jorn Veenstra, and Walter Daelemans. 1999. Cascaded grammatical relation assignment. In Pascale Fung and Joe Zhou, editors, *Proceedings of EMNLP/VLC-99*, pages 239–246. ACL.

Daelemans, W., J. Zavrel, P. Berck, and S. Gillis. 1996. MBT: A memory-based part of speech tagger generator. In E. Ejerhed and I. Dagan, editors, *Proc. of Fourth Workshop on Very Large Corpora*, pages 14–27. ACL SIGDAT.

Marcus, Mitchell, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: Annotating predicate argument structure. In *Proceedings of ARPA Human Technology Workshop*, pages 110–115.

University of Alicante at TREC-10

Vicedo, Jose Luis & Ferrández, Antonio & Llopis, Fernando.

{vicedo,antonio,llopis}@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. Our system is based on our TREC-9 approach where different improvements have been introduced. Essentially these modifications are twofold: the introduction of a passage retrieval module at first stage retrieval and the redefinition of our semantic approach for paragraph selection and answer extraction.

1. Introduction

Open domain QA systems are defined as tools capable of extracting the answer to user queries directly from unrestricted domain documents. Question answering systems performance is continuously increasing since recent Text REtrieval Conferences [9] [10] included a special task for evaluating and comparing this kind of systems. The analysis of current best systems [1] [3] [4] [7] allows identifying main QA sub-components:

- Question analysis
- Document / passage retrieval
- Paragraph selection
- Answer extraction

The system presented to TREC-10 QA task is based on the described structure. It departs from the system presented in last TREC conference [11] where new tools have been added and existing ones have been updated. Modifications introduced rely on several aspects. First, document retrieval stage has been changed. Instead of using first fifty documents supplied by TREC organisation, we have implemented a passage retrieval module that allows a more successful retrieval. Second, our semantic-based paragraph selection approach has been redefined in order to increase selection process performance. Finally, question analysis and answer extraction modules have been updated by including special modules for managing with definition questions.

This year, question answering task has been significantly modified. The organisation has designed three different tasks: *main task, list task* and *context task.* Main task is similar to previous years' tasks but only permitting a maximum of 50 bytes as answer length. Besides, there is no guarantee that an answer will actually occur in the document collection and participants have to measure the degree of correctness of its answers. 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. Finally, the context task consist of answering a set of related questions in such a way that the interpretation of a question will depend on the meaning of and answers to one or more earlier questions in a series.

Our participation has been restricted to the main task although we did not face up all the restrictions. In fact, no effort was accomplished to measure which of the returned answers is more likely to be the correct one or to detect questions without correct answers in the document collection.

This paper is structured as follows: Section 2 describes the structure and operation of our system. Afterwards, we present and analyse the results obtained for TREC-10 task we participated in. Finally, initial conclusions are extracted and directions for future work are discussed.

2. System Overview

Our QA system is structured into the four main modules outlined before: *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 new passage retrieval approach. Afterwards, the paragraph selection module analyses these passages in order to select smaller text fragments 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.

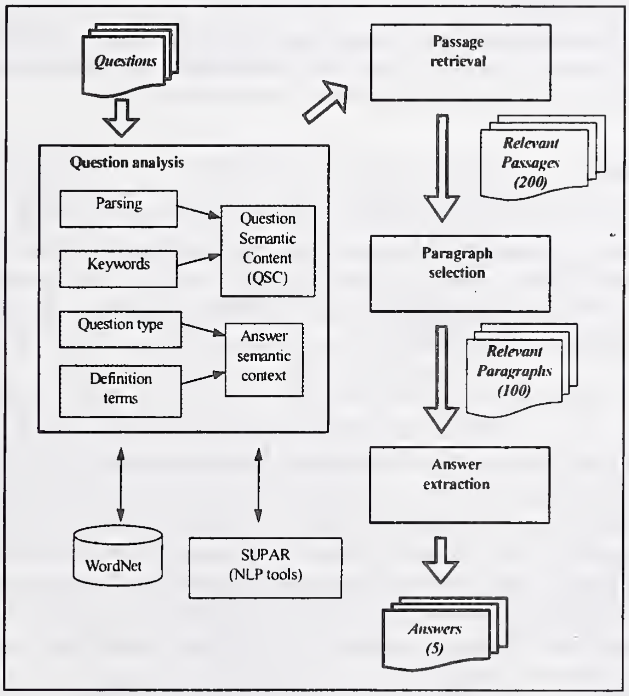


Figure 1. System architecture

Several standard natural language processing techniques have been applied to both questions and documents. These tools compose the Slot Unification Parser for Anaphora Resolution (SUPAR).

2.1. SUPAR NLP tools

In this section, the NLP Slot Unification Parser for Anaphora Resolution (SUPAR) is briefly described [2] [12]. SUPAR's architecture consists of three independent modules that interact with one other. These modules are lexical analysis, syntactic analysis, and a resolution module for Natural Language Processing problems.

Lexical analysis module. This module each document sentence or question to parse as input, along with a tool that provides the system with all the lexical information for each word of the sentence. This tool may be either a dictionary or a part-of-speech tagger. In addition, this module returns a list with all the necessary information for the remaining modules as output. SUPAR works sentence by sentence from the input text, but stores information from previous sentences, which it uses in other modules, (e.g. the list of antecedents of previous sentences for anaphora resolution).

Syntactic analysis module. This module takes as input the output of lexical analysis module and the syntactic information represented by means of grammatical formalism Slot Unification Grammar (SUG). It returns what is called slot structure, which stores all necessary information for following modules. One of the main advantages of this system is that it allows carrying out either partial or full parsing of the text.

NLP problems resolution module. In this module, NLP problems (e.g. anaphora, extraposition, ellipsis or PP-attachment) are dealt with. It takes the slot structure (SS) that corresponds to the parsed sentence as input. The output is an SS in which all the anaphors have been solved. In this paper, only the resolution of third person pronouns has been applied.

2.2. Question Analysis

Question processing module accomplishes several tasks. First, SUPAR system accomplishes part-of-speech tagging and parsing of the question. Afterwards, this module determines *question type*, classifies non-Wh terms into two categories (*keywords* or *definition terms)* and finally, concepts referred into the question are detected and processed to obtain the semantic representation of the concepts appearing in the question.

Question type is detected by analysing Wh-terms (e.g. What, Which, How, etc). This process maps Wh-terms into one or several of the categories listed in figure 2. Each of these categories is related to WordNet top concepts [6]. This module has been updated by including the definition questions as new question type. When no category can be detected by Wh-term analysis, NONE is used (e.g. "What" questions). This analysis gives the system the following information: (1) lexical restrictions that expected answer should validate (e.g. proper noun), (2) how to detect definition terms (if they exist), and (3) top WordNet concepts and related synsets that are compatible with the expected answer. Definition questions are detected by applying a pattern matching process. As example, questions such as "Who was Galileo?", "What are amphibians?" or "What does USPS

PERSON	GROUP	LOCATION	TIME	
QUANTITY	DEFINITION	REASON	MANNER	NONE

Figure 2. Question type categories

stands for?" are correctly analysed.

Once question type has been obtained, the system selects the *definition terms.* A term in a query is considered a definition term if it expresses semantic characteristics of the expected answer. Definition terms do not help the system to locate the correct answer into the document collection but they usually describe the kind of information requested by a query. Depending on question type, different patterns are used to detect definition terms. For "What", "Which", "How" and similar questions, this terms are detected by selecting noun phrases located next to the Wh-term. When questions such as "Find the number of whales..." or "Name a flying mammal ..." are analysed, noun phrases following the verb are considered definition terms.

Question type and definition terms are used to generate the *expected answer semantic context* (EASC). This context defines the lexical characteristics that the expected answer should validate to be considered a probable answer (e.g. proper noun) and the semantic context that the expected answer has to be compatible with. This context is made up by the set of synsets that are semantically related to definition terms and question type. These synsets are obtained by extracting from WordNet all hyperonyms of each definition term (its path to top concepts). These synsets are weighted depending on its level into the WordNet hierarchy and the frequency of its appearance into the path towards top concepts. Intuitively, this set of synsets defines the semantic context that has to be compatible with the expected answer semantic context. Finally, remaining question terms are classified as *keywords.*

Last question processing stage builds the semantic representation of the 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 questions and its main aim is to achieve concept representation in such a way that make possible to overcome term-based approach limits into the paragraph selection stage. To obtain this representation we have to deal with two basic requirements:

a) Concepts appearing in questions need to be correctly detected and extracted.
b) The different ways of expressing a concept have to be obtained and represented.

First requirement is accomplished by parsing questions. This process obtains all the syntactic structures that made up each question. Structures containing definition terms are discarded. Then, each syntactic structure (noun and verbal phrases) that contains one or more keywords defines a concept. The head of each 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.

Accomplishing the second requirement involves obtaining and representing the different ways of expressing each of the concepts detected in a query. 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* [8] 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. This process is widely explained in [13].

Figure 3 shows the semantic content of an example question First, the system identifies the concepts "*manufactures*" and "*American Girl doll collection*" by detecting syntactic structures that contain keywords. Afterwards, the semantic content of each concept is generated.

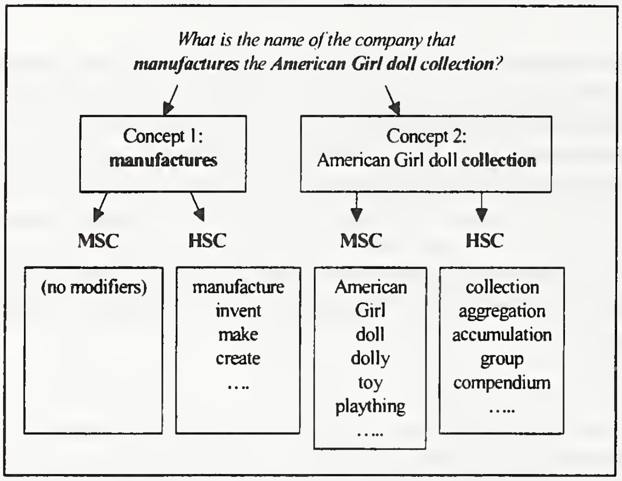


Figure 3. Example of QSC

Question keywords are used for first stage passage retrieval while QSC information will help paragraph selection module to detect the paragraphs that are more likely to contain the answer.

2.3. Passage retrieval module

First stage retrieval applies the passage retrieval approach described in [5]. This passage retrieval can be applied over all the document collection, but it has only been applied for the 1000 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 costly NLP modules since these passages are made up by text snippets of 15 sentences length.

2.4. Paragraph selection

This module processes 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 [13] we extract here the basic algorithm:

a) Documents are split into sentences.

b) Overlapping paragraphs of three sentences length are obtained.

c) Each paragraph is scored. This value measures the similarity between each paragraph and the question.

d) 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).

At this stage, only best 100 ranked paragraphs are selected to continue with the remaining processes.

2.5. Answer extraction

This process consists on analysing selected paragraphs in order to extract and rank the text snippets of the desired length that are considered to contain the correct answer. For this purpose, the system selects a window for each probable answer by taking as centre the term considered a probable answer. Each window is assigned a score (*window-score*) that is computed as follows:

$$\textit{Window-score} = \textit{paragraph-score}*(1+\cos(EASC,PASC))$$

where *EASC* is the vector representing the semantic context of the expected answer and *PASC* stands for the vector representing the semantic context of the possible answer. PASC is computed as done for EASC but using the terms contained into the syntactic structures the probable answer appear into, as well as surrounding syntactic structures.

Intuitively, the window-score combines (1) the semantic compatibility between the probable answer and the expected answer (cos(EASC,PASC)) and (2) the degree of similarity between question and paragraphs (paragraph-score).

Finally, windows are ranked on window-score and the system returns the first five as answer.

Answer extraction manages differently with definition questions. This questions look for answers that define or explain the concept expressed in the question. From the analysis of definition questions in TREC-9 question set we derived a set of heuristics for detecting answers to definition questions. Each of these heuristics refers to a different way of expressing definition answers. The following list shows the main ways in which answers to definition questions are more probably expressed and several examples (the answer is italicised):

- Noun phrases including the answer ("*Italian archbishop* Filippo Cune ...").
- Explanatory appositions (" Filippo Cune, the *Italian archbishop*, …").
- Explanatory conjunctions ("*Italian archbishops* Federico Pane, Filippo Cune and ...").
- Definition phrases (" Filippo Cune was the *Italian archbishop* …").
- Coreference resolution ("Filippo Cune travelled to Pisa. The *Italian archbishop* desired to renew the …").
- …

These heuristics were ordered depending on the probability of obtaining a correct answer to a question (*heuristic probability*) by applying each of them on TREC-9 definition question set. This order determines the sequence of application of each heuristic over relevant paragraphs. The following algorithm shows how these heuristics are applied:

a) Heuristics are applied over each relevant paragraph in an ordered way until one of them (or none) succeeds.
b) Answers detected by successful heuristics are extracted.
c) These answers are scored (*answer-score*) as follows:

*Answer-score = paragraph-score * heuristic probability*

d) For duplicated answers, only the highest ranked is maintained,
e) First five ranked answers are returned as final answers.

3. Results

This year we submitted two runs for main task. This task allowed five answers for each question and a maximum answer string length of 50 bytes. Figure 4 shows the results obtained. Applying the whole system described above has produced ALIC01M2 run. ALIC01M1 files contain results obtained applying the same strategy but without solving pronominal anaphora in relevant passages. These results were computed after the organisation decided to get rid of eight questions. Therefore, 492 questions were evaluated.

Although a detailed results analysis is a very complex task, several conclusions can be extracted.

Run	Mean reciprocal rank		% Answers found	
	strict	lenient	strict	lenient
ALIC01M1	0,296	0,302	39,2%	40,0%
ALIC01M2	0,300	0,306	39,6%	40,4%

Figure 4. TREC-10 main task results

Comparison with TREC-9 results.
Our system has achieved a significant improvement since TREC-9 participation. Comparison between strict best results for 50 bytes answer length at TREC-9 (see figure 5) and TREC-10 (figure

4) shows that the mean reciprocal rank has increased 0.7 points (from 0.23 to 0.30) and besides, the percentage of correct answers found has increased 5.7 points (from 33.9% to 39.6%).

Run	Mean reciprocal rank		% Answers found	
	strict	lenient	strict	lenient
ALI9C50	23,0%	24,5%	33,9%	36,1%
ALI9A50	22,7%	24,0%	33,9%	35,8%

Figure 5. TREC-9 50 bytes answer length results

Retrieving relevant documents.
Correct answer was not included into the top ranked documents supplied by TREC for 61 questions. If we discard the 49 questions with no correct answer in the collection this number falls to 12 questions. Figure 6 compares the percentage of questions that could be correctly answered between the two possible approaches: (1) processing a number of top documents and (2) selecting a number of passages.

	Top Passages		Top Documents						
500 questions	100	200	50	100	200	350	500	750	1.000
Answer included	200	424	393	407	420	430	432	435	439
Answer Not included	300	76	107	93	80	70	68	65	61
% Answer Included	40,0%	84,8%	78,6%	81,4%	84,0%	86,0%	86,4%	87,0%	87,8%

Figure 6. Passage and document retrieval comparison

As we can notice processing 200 passages produces best results than processing 200 complete documents and besides it dramatically reduces later NLP processing costs.

Paragraph selection.
Our main objective was to inspect if our new paragraph selection method was more effective than last year proposal. As we expected, this model has achieved a better performance. Strict MRR increased 0.7 points from past results, which corroborates that precision achieved at this process has improved significantly.

Pronominal anaphora resolution
The small benefit obtained last year from applying pronominal anaphora resolution has been corroborated with TREC-10 results. This fact is mainly due to the same reasons described last year [11]. Nevertheless, although we have not participated into the context task thise kind of questions will surely take more profit form coreference resolution techniques.

4. Future Work

Several areas of future work have appeared while analysing results. First, passage retrieval has to be tested over the whole collection to investigate the level of benefit it can produce over current results. Besides, although our paragraph selection module has revealed to be very efficient, several aspects can be improved, especially by incorporating a validation module that could measure the inexistence of the answer. Third, it seems essential to incorporate a Name-Entity tagger to our answer extraction module since we missed several answers that could have easily been detected. And fourth, the system needs to be adapted to manage with list and context questions.

5. References

1. Clarke C.L., Cormack G.V., Kisman D. and Lynam T. R. *Question Answering by Passage Selection (MultiText Experiments for TREC-9).* In Proceedings of the Nineth Text Retrieval Conference. November 2000. Gaithersburg (US).
2. Ferrández A., Palomar M. and Lidia Moreno. *An empirical approach to Spanish anaphora resolution.* Machine TranslationSpecial Issue on Anaphora Resolution in Machine Translation. Kluwer Academic publishers. ISSN 0922-6567 .1999.vol 14(3/4) pages 191-216.
3. Harabagiu S., Moldovan D., Pasca M., Mihalcea R., Surdeanu M., Bunescu R., Gîrju R., Rus V, and Morarescu, P. *FALCON: Boosting Knowledge for Answer Engines.* In Proceedings of the Nineth Text Retrieval Conference. November 2000. Gaithersburg (US).
4. Ittycheriah A., Franz M., Zu W. and Adwait Ratnaparkhi. *IBM's Statistical Question Answering System.* In Proceedings of the Nineth Text Retrieval Conference. November 2000. Gaithersburg (US).
5. 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. (To appear). September 2001. Darmstadt (Germany).
6. Miller G.(1995), *"Wordnet: A Lexical Database for English"*, Communications of the ACM 38(11) pp 39-41.
7. Prager J., Brown E., Radev D. and Krzysztof Czuba. *One Search Engine or Two for Question-Answering.* In Proceedings of the Nineth Text Retrieval Conference. November 2000. Gaithersburg (US).
8. Salton G.(1989). *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison Wesley Publishing, New York.
9. TREC-8, 1999. Call for participation Text Retrieval Conference 1999 (TREC-8).
10. TREC-9, 2000. Call for participation Text Retrieval Conference 2000 (TREC-9).
11. Vicedo J.L. and Ferrandez A. *A semantic approach to Question Answering systems.* In Proceedings of the Nineth Text Retrieval Conference. November 2000. Gaithersburg (US).
12. Vicedo J.L. and Ferrández A. *Importance of Pronominal·Anaphora resolution in Question Answering systems.* In proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL2000). October 2000. Hong Kong (China).
13. 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).

Tequesta: The University of Amsterdam's Textual Question Answering System

Christof Monz **Maarten de Rijke**
Language & Inference Technology, University of Amsterdam
Nieuwe Achtergracht 166, 1018 WV Amsterdam
The Netherlands
E-mail: {christof,mdr}@science.uva.nl
URL: www.science.uva.nl/~ {christof,mdr}

Abstract: We describe our participation in the TREC-10 Question Answering track. All our runs used the Tequesta system; we provide a detailed account of the natural language processing and inferencing techniques that are part of Tequesta. We also summarize and discuss our results, which concern both the main task and the list task.

1 Introduction

Current information retrieval systems allow us to locate documents that might contain the pertinent information, but most of them leave it to the user to extract the useful information from a ranked list. However, users want not whole documents but brief answers to specific questions. Question answering is meant to be a step closer to real information retrieval in that it attempts to facilitate just that.

For researchers (such as ourselves) who are interested in bringing natural language processing (NLP) and inferencing to bear on real-world tasks, question answering (QA) provides an ideal setting. Many years of experimental research have shown that advanced NLP techniques hurt more than they help for traditional document retrieval [1]. For any system that aims to address the QA task, however, issues such as question classification, partial parsing, and named entity recognition appear to be essential components. In addition, the best performing systems at the TREC-8 and TREC-9 QA tracks have demonstrated that various forms of inferencing (ranging from the use of semantic relations in WordNet to actually abducing answers from questions) make a significant positive contribution towards the effectiveness of QA systems [20, 19]. The recently released (and deliberately ambitious) vision statement that aims to guide future research in QA calls for approaches that are even more knowledge intensive than the current ones [8].

This paper describes our submissions for the question answering track at TREC-10; we submitted runs for the main task and for the list task. This is the first time that we participated in the QA track (and in TREC, for that matter), and our main focus was on evaluting a basic question answering system that exploits shallow NLP techiques in combination with standard retrieval techniques.

The remainder of the paper is organized as follows. Section 2 describes the Tequesta system that we developed for the QA track. We outline the underlying retrieval engine, the kind of document analysis that we perform (partial parsing and named entity recognition), as well as our question analysis and answer selection modules. Then, in Section 3 we describe the runs that we submitted to the main QA task, and discuss the outcomes. In Section 4 we do the same for the runs submitted to the list task. Section 5 contains our conclusions and plans for future work.

2 System Description

2.1 System Architecture

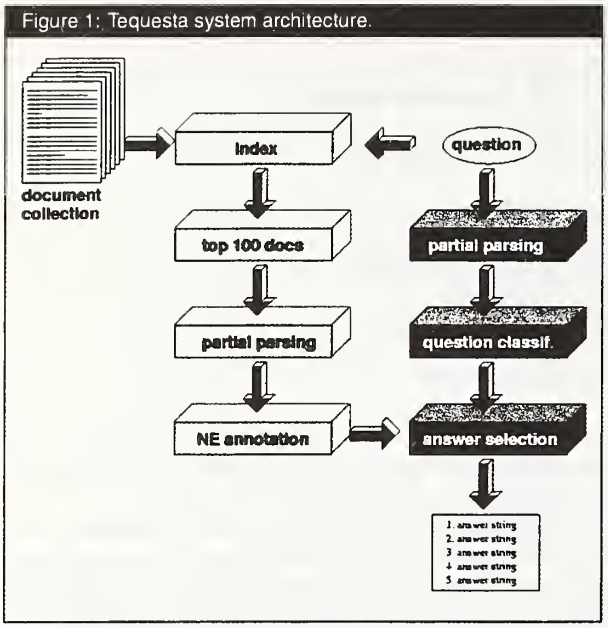
The system architecture of Tequesta is fairly standard; its overall architecture is displayed in Figure 1. Like most current QA systems, Tequesta is built on top of a retrieval system. The first step is to build an index for the document collection, in this case the TREC-10 collection. Then the question is translated into a retrieval query which is posed to the retrieval system. For retrieval we use FlexIR [13], a vector-space based retrieval system, described in Section 2.1.

The retrieval system is used to identify a set of documents that are likely to contain the answer to a question posed to the system. The top 100 documents returned by FlexIR are processed by a partial parser described in Section 2.2. Then, named entities are annotated with the appropriate type. Named entity recognition is discussed in Section 2.3.

Just like the top 100 documents, the question is also parsed. The parsed output is used to determine the focus of the question, i.e., what it is looking for. Question analysis is explained in Section 2.4.

The document analysis and question analysis are mostly done independently from each other, but in order to generate a top 5 list of answers, document information and question information are combined in the answer selection process, described in Section 2.5.

Figure 1: Tequesta system architecture.



2.2 Document Retrieval

For pre-fetching relevant documents that are likely to contain the answer, Tequesta uses FlexIR, an information retrieval system developed at the University of Amsterdam. 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; it is built around the standard UNIX pipeline architecture, and supports many types of pre-processing, scoring, indexing, and retrieval methods.

The retrieval model underlying FlexIR is the standard vector space model. All our official runs for TREC-10 used the Lnu.ltc weighting scheme [4] 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 lexical-based stemmer, or lemmatizer, because it tends to be less aggressive than rule-based stemmers such as Porter's [14] or Lovins' [9] stemmer. The lemmatizer is part of the TreeTagger part-of-speech tagger [17]. 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.

2.3 Document Analysis

2.3.1 Partial Parsing

At present, full parsing is still computationally rather expensive, and building a grammar that is able to cope with a large number of phenomena is very laborious. For these reasons we decided to use a partial parser which can at least identify simple phrases of various kinds. Our partial parser is based on finite-state technology and is therefore able to process large amounts of data efficiently.

We focused on identifying noun phrases (NPs), prepositional phrases (PPs) and verb groups (VGs). A verb group is the verbal complex containing the semantic head of a verb phrase (VP) and its auxiliaries (*have*, *be*) and modal modifiers (*can*, *would*, *should*, etc.).

For each noun phrase, its semantic head is marked. If the noun phrase is complex, the right most noun is identified as the head [21], which holds for almost all noun phrases in English.[1] Similarly, the semantic head of a prepositional phrase is the head of its noun phrase and the syntactic head is the preposition.

NPs, PPs, and VGs form the basic constituents of a dependency structure. A dependency structure is headed by a VG and the NPs and PPs in its vicinity are arguments or modifiers of the verb. We did not exploit subcategorization information derived from the verb in order to deal with ambiguities arising from more complex verb-argument relations such as controlling verbs (e.g., *promise*, *persuade*), and anaphoric relations. A disadvantage of this approach is that it is harder to distinguish between arguments and modifiers of a verb. On the other hand, using a flat and underspecified representation, one does not have to cope with these ambiguities. Since we did not try to resolve anaphoric relations, this approach has the additional advantage that noun phrases serving as antecedents to intra-sentential pronouns are considered to be part of the dependency structure. Consider, for instance, the sentence in (1), taken from document `AP900416-0132`.

(1) Teachers in Oklahoma City and some other districts said they feared reprisals if they took part in the strike.

Neglecting any context possibly preceding (1), there are four potential antecedents of the plural pronoun *they*:

(2) a. Teachers
b. Teachers in Oklahoma City
c. some other districts
d. Teachers in Oklahoma City and some other districts

The correct antecedent of *they*, and therefore the subject of *fear* and *take part* is (2.d). Since resolving anaphora, and plural anaphora in particular, can be rather complex, we refrained from this task and relaxed our notion of dependency structure instead.

There are three dependency structures that can be identified in (1). An abstract representation is given in (3):

(3) a. `head`: say
`arg(1,1)`: teacher
`arg(1,2)`: in Oklahoma City
`arg(1,3)`: some other district

[1] Exceptions of the Right-hand Head Rule (RHR) include some hyphenated noun phrases, such as *passer-by* and *mother-in-law*.

b. head: fear
   `arg(l,1)`: teacher
   `arg(l,2)`: in Oklahoma City
   `arg(l,3)`: some other district
   `arg(r,4)`: reprisal

c. head: take-part
   `arg(l,1)`: teacher
   `arg(l,2)`: in Oklahoma City
   `arg(l,3)`: some other district
   `arg(r,4)`: reprisal
   `arg(r,5)`: in the strike

The two parameters of `arg` indicate the direction of the argument with respect to the heading verb and the distance.

In the actual system, syntactic annotation is done in XML format. Below we show the dependency structure for (3.b).

```
<C CAT=NP SEMHEAD=teacher TYPE=SMTH ID=217-217>
   <C CAT=NNS ID=217 LEM=teacher>Teachers</C>
</C>
<C CAT=PP SEMHEAD=City SYNHEAD=in ID=218-220>
   <C CAT=IN ID=218 LEM=in>in</C>
   <C CAT=NP SEMHEAD=City TYPE=CITY_3 ID=>
      <C CAT=NNP ID=219 LEM=Oklahoma>Oklahoma</C>
      <C CAT=NNP ID=220 LEM=City>City</C>
   </C>
</C>
<C CAT=CC ID=221 LEM=and>and</C>
<C CAT=NP SEMHEAD=district TYPE=SMTH ID=222-224>
   <C CAT=DT ID=222 LEM=some>some</C>
   <C CAT=JJ ID=223 LEM=other>other</C>
   <C CAT=NNS ID=224 LEM=district>districts</C>
</C>
<C CAT=NP SEMHEAD=they TYPE=SMTH ID=226-226>
   <C CAT=PRP ID=226 LEM=they>they</C>
</C>
<C CAT=VG SEMHEAD=fear PART= VC=act
 DEP=l_217-217(teacher),l_218-220(City),
 l_222-224(district),r_228-228(reprisal)
 ID=227-227>
   <C CAT=VBD ID=227 LEM=fear>feared</C>
</C>
<C CAT=NP SEMHEAD=reprisal TYPE=SMTH ID=228-228>
   <C CAT=NNS ID=228 LEM=reprisal>reprisals</C>
</C>
```

A number of things require further explanation, and we will briefly discuss most of the features present in the XML structure above. The CAT feature represents the syntactic category of a word or a phrase. The word categories are based on the Penn Treebank tag set, cf. [16]. The morphologically normalized form of a word, its lemma, is given by the LEM feature. The SEMHEAD feature marks the semantic head of a phrase, and in case of a PP the SYNHEAD feature marks the preposition as the syntactic head. Each occurrcncc of a word in a document has a unique identifier, indicated by the ID feature. Similarly, each top-level phrase has a unique identifier indicating its scope. The TYPE feature assigns a semantic type to named entities, SMTH (something) being the default value. Named entity annotation is discussed in more detail in the next subsection. Whether a verb is in active or passive voice is marked by the VC feature.

Returning to the representation of dependency structures, this information is contained in the annotation of the VG phrase. The feature DEP has as its value a list of strings, separated by a comma. For instance, `l_222-224(district)` says that the phrase `222-224` is within the scope to the left of the verb group and that its semantic head is `district`. Anaphoric phrases, such as `226-226`, are not mentioned in the dependency list.

In addition to VG phrases some noun phrases can also have dependency relations. Nominalizations, such as (4), behave very much like the verbs from which they are derived.

(4) Mr Paul Volcker, the former chairman of the US Federal Reserve Board, is considering an offer to serve as an *adviser* to the Russian government on economic and banking reform.

In (4), taken from document `FT921-10181`, *adviser*, or its underlying verb *advise*, takes *Mr Paul Volcker* as subject and *Russian government* as object. To identify nominalizations, we used CELEX [2] and NOMLEX [10] as lexical resources.

As we will see in Section 2.5, dependency structures are the basic constituents in the answer selection process for several types of questions. Especially questions of the form *Who VP?* make use of dependency structures to match the question with dependency structures within the document.

2.3.2 Named Entity Annotation

In addition to the syntactic annotation described in the previous subsection, we also annotate some named entities with their semantic types. The set of semantic types that we have used is shown in Table 1. Some of the semantic types, such as PERS and LOC, are further divided into subtypes.

Table 1: Types for named entity annotation.

Type	Subtypes	Description
COMP		companies and organizations
NUMERIC	MONEY	monetary expressions
	NUM-RATIO	percentages
DATE		explicit dates
TIME		time periods
LOC	COUNTRY	countries
	STATES	U.S. states
	PROVINCE	provinces
	CITY	cities
	PORT	harbors
	ISLAND	islands
PERS	MALE	male persons
	FEMALE	female persons
SMTH		other NPs

Type recognition is accomplished by fairly simple techniques such as pattern matching, gazetteer look-up, or a combination of both.

To identify companies, organizations, associations, etc. we compiled a list of names and extracted 20 features that occur frequently. For instance, *&*, *Inc.*, and *International* are likely to indicate the name of a company. If a company or organization name was followed by an expression between parentheses, adhering to some pattern, we took it to be the company's abbreviation and added this information to the annotation in order to allow for aliases. For instance, the annotation for the *Asia Pacific Economic Co-operation Group (APEC)* is as follows:

```
<C CAT=NP SEMHEAD= TYPE=COMP ABBR=APEC ID=356-364>
 <C CAT=DT ID=356 LEM=the>the</C>
 <C CAT=NNP ID=357 LEM=Asia>Asia</C>
 <C CAT=NNP ID=358 LEM=Pacific>Pacific</C>
 <C CAT=NNP ID=359 LEM=Economic>Economic</C>
 <C CAT=NN ID=360 LEM=cooperation>Co-operation</C>
 <C CAT=NNP ID=361 LEM=Group>Group</C>
 <C CAT=( ID=362 LEM=(>(</C>
 <C CAT=NNP ID=363  LEM=APEC>APEC</C>
 <C CAT=) ID=364 LEM=)>)</C>
</C>
```

Keeping track of abbreviations does not only allow one to match a name with its abbreviation when a question is matched with a dependency structure, it can also be used for questions concerning abbreviations directly; e.g., questions of the form *What does X stand for?*.

Phrases of type NUMERIC, DATE, and TIME, are recognized by pattern matching. The TIPSTER gazetteer, containing a list of more than 240,000 locations, is used to find names of cities, provinces, etc.

The identification of person names uses the U.S. census list of the 80,000 most frequent last names, 4275 most frequent female first names, and 1219 most frequent male first names in the U.S. as a gazetteer. In addition we look for particular indicators for a person name, including titles, such as *Mrs.*, *President*, *Dr.*, and relative clauses following an NP with capitalized nouns. If a name was identified by pattern matching, it was dynamically added to the list of known names. Whenever it was possible to identify the gender of a person by looking at the first name or title, the more specific subtype information was recorded. Although we did not yet exploit this distinction in the current version of our system, we plan to do so in the future in order to facilitate anaphora resolution.

If an NP cannot not be recognized by the techniques above, it receives the default semantic type SMTH.

Obviously, these techniques are rather simple and error prone. In particular, the use of gazetteers has the disadvantages of being inherently incomplete and causing false alarms; see e.g., [11] for a discussion of the use of gazetteers in the area of Information Extraction. More sophisticated systems such as IdentiFinder$^{\text{TM}}$ [3] therefore use feature learning techniques for named entity annotation. On the other hand, the use of gazetteers has the advantage of being rather simple to implement, which was the main reason we opted for this solution.

In the current version of system, false alarms account for the majority of errors made by the name entity recognizer. This is caused mainly by the interference of location names and person names. As we do not allow for multiple typing, this has the effect that once a named entity is falsely recognized as being of type A, it cannot be identified as being of type B. Since it is rather unlikely that we will replace the gazetteer look-up by a feature-learning component in the near future — for the aforementioned reasons — we at least intend to allow for multiple typing. As a consequence, false alarms will continue to have a negative impact on precision, but recall should increase.

Our final remark on the named entity annotation component concerns the interaction between annotation and document retrieval. Currently, the named entity recognizer is applied to the top 100 documents returned by our retrieval system FlexIR. We did not apply the recognizer to the collection as a whole. Pre-processing the whole collection would have two advantages: First, it results in a more efficient system (although efficiency was not one of our major concerns at the current stage), and second, it is possible to index the collection with respect to the semantic types attached to named entities and exploit this additional information during retrieval, cf., e.g., [15]. The main reason for not doing so was that we developed the named entity recognizer in tandem with the other components. Since applying it to the whole collection is rather time consuming, it would have increased the duration of each development cycle in a significant way. We are hopeful that once we have enabled multiple typing, we will have a stable and reliable version of the recognizer which can used to assist the retrieval process.

2.4 Question Analysis

Just like the top 100 documents, the questions themselves were also part-of-speech tagged, morphologically normalized, and partially 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 18 categories to classify the focus or target of a question; the first 16 of these are listed in Figure 2. The two missing categories (what:X and unknown) will shortly be discussed.

To identify the target of a question, pattern matching is applied to assign one of the 18 categories to the question. In total, a set of 67 patterns is used to accomplish this. Some of the patterns used are shown in Table 2.

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:X form a special category. Here

Table 2: 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>also-known-as</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:X</td><td>-</td></tr>
<tr><td>unknown</td><td>-</td></tr>
</table>

we use partial parsing to identify the appropriate target, symbolized by X in the type. Usually, what:X questions are of the form *What NP VP?* or *What NP PP VP?*. After parsing the question, we use the head of the NP following *what* as the target, potentially modified by further constituents from the NP or PP modifying the head. For instance, question 934 from the TREC-10 question set, shown in (5), is assigned what:plant, and question 1339, shown in (6), is assigned what:breed:of dog as question target.

(5) Material called linen is made from what plant?

(6) What breed of hunting dog did the Beverly Hillbillies own?

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.

2.5 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. Although the top 100 documents are analyzed, earlier experiments on TREC-9 questions have shown that in some cases focusing on the top 25 or top 50 documents in the answer selection process results in a better performance. Therefore, we restricted ourselves to analyzing the top 100 documents and varied the parameter of documents analyzed during selection over the submitted runs.

While answer selection strongly depends on the question target, a basic strategy common to all question types is to match the dependency structure(s) present in the question with dependency structures in the top documents. More precisely, we try to find a *maximally* matching segment; in our implementation such a segment can be a sentence or a pair of adjacent sentences. Once such a segment has been found, we check whether it contains constituents that fit the appropriate question target. If this is the case, these constituents are marked as potential answers, and the next best matching segment is analyzed, etc.

For this strategy to work, it is important to have a proper matching algorithm that allows for partial matching and also assigns a weight or score to a match that allows to compare and rank different matches.

Matching dependency structures involves three steps: First it has to be checked whether the two heads, i.e., verbs, match, and then the overlap between the arguments of the two structures has to be determined. Since the arguments themselves can be complex phrases, it is necessary to also apply phrase matching on this lower level so as to determine to which extent two arguments match.

There is a number of ways to devise a phrase matching algorithm, although the literature on phrase matching is rather sparse. To our knowledge, there is only one algorithm described in the literature, viz. [7]. Note that phrase matching is different from phrase weighting, cf., e.g., [5, 18], which assigns a weight to a whole phrase but does not deal with partial matches between phrases, which is essential in this context.

Here, we will briefly describe one of our implementations of a phrase matching algorithm which was used for all submitted runs. Given two phrases p1 and p2, the function phrase_match returns a real between 0 and 1 as the matching score. Stop words, such as *a*, *the*, *some*, *all*, etc., are removed before the phrases are passed as arguments to phrase_match. A pseudo algorithm for phrase_match is given in Figure 3.

First, the if-than-else statement in lines 2–6 checks

Figure 2: Question targets, plus examples from the TREC-9 and TREC-10 questions.

name the name of a person or an entity in general.
(Q-1094): *What is the name of the satellite that the Soviet Union sent into space in 1957?*

pers-def the function or role of a person
(Q-959): *Who was Abraham Lincoln?*

thing-def further explanation or definition of some entity
(Q-903): *What is autism?*

pers-ident a person fitting some description expressed in the question
(Q-973): *Who was the first governor of Alaska?*

thing-ident thing fitting some description expressed in the question
(Q-988): *What is the oldest university in the US?*

number some kind of numerical expression. Actually, the `number` target is subdivided into different subtypes such as `number-money`, `number-height`, `number-distance`, etc.
(Q-1156): *How many Admirals are there in the U.S. Navy?*

expand-abbr the full meaning of an abbreviation
(Q-1176): *What does I.V. stand for?*

find-abbr the abbreviation for some name
(Q-540): *What's the abbreviation for limited partnership?*

agent name or description of an animate entity
(Q-1239): *Who painted the ceiling of the Sistine Chapel?*

object `object` questions are near-reverses of the `agent` questions. Here, the object of an action described in the question is sought.
(Q-1354): *What did Jesse Jackson organize?*

known-for distinguishing feature of some entity
(Q-207): *What is Francis Scott Key best known for?*

also-known-as alternative name for some entity
(Q-1044): *What is another name for vitamin B1?*

name-instance an instance of some description expressed in the question
(Q-1268): *Name a food high in zinc.*

location location of some entity
(Q-1351): *Where was the first golf course in the United States?*

date date of an event
(Q-1302): *When was the Boston tea party?*

reason reason for an event or fact
(Q-1220): *Why is the sun yellow?*

Figure 3: Phrase matching algorithm.

```
 1 float phrase_match(phrase p1, phrase p2) {
 2   if(head(p1) = head(p2)) {
 3       score = 0.5;
 4   } else {
 5       return 0;
 6   };
 7
 8   if(length(p1) > length(p2)) {
 9       max_length = length(p1) - 1;
10   } else {
11       max_length = length(p2) - 1;
12   };
13
14   if(max_length = 0) {
15     return score;
16   };
17
18   foreach const ∈ (p1∪p2)\head(p1) {
19     if(const ∈ (p1∩p2)\head(p1)) {
20         score += 0.5/max_length;
21     };
22   };
23   return score;
24 }
```

whether the semantic heads of the two phrases are identical. If this is the case, the initial score is set to 0.5, otherwise, `phrase_match` returns with a matching score of 0. This reflects our strong emphasis on the head of a phrase. Of course this leaves room for other options, such as choosing a different value or not returning immediately if the heads do not match.

Lines 8–12 compare the lengths of the two phrases, initializing `max_length`. Since the heads were already compared, they can be neglected and `max_length` is decremented by 1 in line 9 and 11. `max-length` is the maximal number of constituents that the two phrases can have in common. Later on it is used for normalization. If `max_length` equals 0, this means that no constituents other than the heads are to be compared and `phrase_match` returns with the value 0.5, see lines 14–16.

Then, for each constituent occurring in either one of the phrases we check whether it occurs in both phrases (lines 18–22). If this is the case, `score` is incremented by `0.5/max_length`. Finally, line 23 returns the final matching score.

A couple of remarks are in order. First, except for the identification of the head, we do not consider word order; i.e., matching phrases of the form ABC and BAC get a score of 1 although they differ in word order. A side effect is that the distance of a constituent to the head of its phrase is not considered, although one might argue that the closer a constituent is to the head, the more important it is.

Another simplification is the fact that we neglect term importance such as *tf.idf* weighting. Each constituent or term occurring in both phrases contributes equally to the computation of the matching score, even though some terms are obviously more content bearing than others.

Finally, in the algorithm as it was described above, two constituents are compared with respect to identity. This is a very strict constraint which was softened in the actual implementation of Tequesta. We used WordNet [6] relations, such as synonomy and hyponomy, thus allowing for a match between two constituents if they are in linked by chain of these WordNet relations.

Phrase matching is used in the process of matching dependency structures, which, in turn, helps us to rank matching text segments taken from the top documents. Starting with the highest ranked segment, we apply strategies that depend on the question target to extract the answer string from these segments. In the remainder of the subsection we briefly discuss some of our strategies.

When selecting the answer to a question, we distinguish between the *focus*, or target, of a question and its *topic*. The focus is the element the question is asking for, or put differently, the element lacking. The focus, on the other hand, is the information providing some description or context, the answer should fit into.

Questions of type `pers-def` or `thing-def` ask for the function or role of person and some further explanation or definition of a thing, respectively. Often, this kind of information is contained in an apposition (as illustrated by (8.a)) or a relative clause following the occurrence of this person's name or thing's name (as illustrated by (8.b)).

(7) Who is Desmond Tutu?

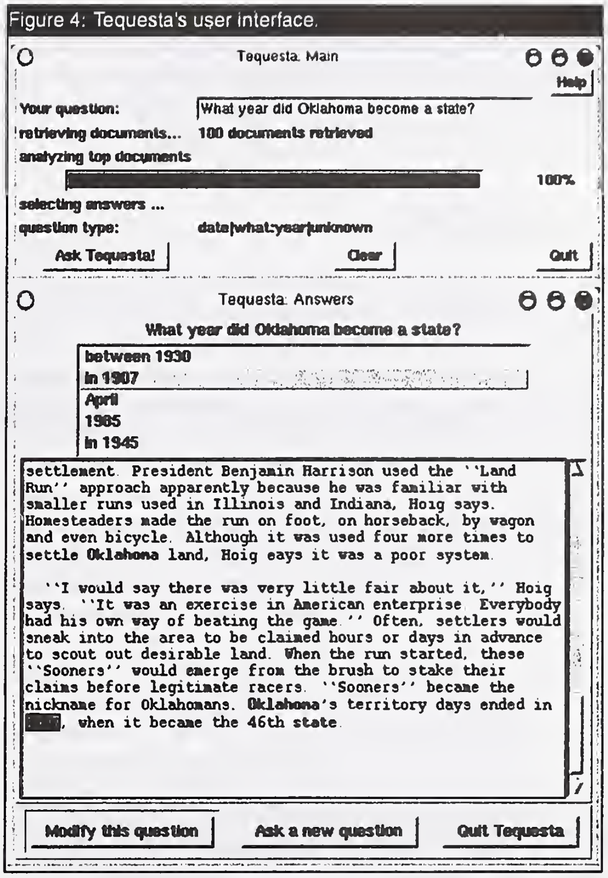
(8) a. Tutu, winner of the 1984 Nobel Peace Prize
b. Desmond Tutu, who is a member of Harvard University's governing board

In order to make sure that the apposition or relative clause forming the potential answer contains descriptive information rather than some other information we apply further heuristics. For instance, a potential answer is preferred if it contains superlative adjectives, such as *first*, *highest*, *most*, etc., or nouns ending in *-er* which are likely to describe some role, e.g., *winner*, *member*, etc.

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 (positive sentence) or 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:X` are particularly interesting because they are very frequent (at least in the TREC data) and explicitly require some lexical knowledge base. Questions of type `what:X` ask for something that is a kind of `X` and that fits the further description expressed in the remainder of the question. For example, question 429, given in (9), asks for something which is a university.

Figure 4: Tequesta's user interface.



Your question: What year did Oklahoma become a state?
retrieving documents... 100 documents retrieved
analyzing top documents
100%
selecting answers ...
question type: date|what:year|unknown

What year did Oklahoma become a state?

between 1930
in 1907
April
1985
in 1945

```
settlement. President Benjamin Harrison used the ``Land
Run'' approach apparently because he was familiar with
smaller runs used in Illinois and Indiana, Hoig says.
Homesteaders made the run on foot, on horseback, by wagon
and even bicycle. Although it was used four more times to
settle Oklahoma land, Hoig eays it was a poor system.

  ``I would say there was very little fair about it,'' Hoig
says. ``It was an exercise in American enterprise. Everybody
had his own way of beating the game.'' Often, settlers would
sneak into the area to be claimed hours or days in advance
to scout out desirable land. When the run started, these
``Sooners'' would emerge from the brush to stake their
claims before legitimate racers. ``Sooners'' became the
nickname for Oklahomans. Oklahoma's territory days ended in
[illegible], when it became the 46th state.
```

(9) What university was Woodrow Wilson President of?

In (9) *university* is the focus of the question and the further constraint *was Woodrow Wilson President of?* 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.

While extracting potential answers, we also keep track of the number of steps that had to been taken while traversing WordNet, and the matching scores that were involved. The higher the matching scores and the smaller the number of lexical relations that had to be used from WordNet, the higher the overall answer score of a potential answer. Finally, the extracted answer strings are ordered and the top five are selected as the final set of answers.

Tequesta also provides a graphical user interface which we use for evaluation and demonstration purposes. Figure 4 shows the two windows that are used to interact with the user. The top window in Figure 4 is the main window; it allows the user to enter a question and provides information on the status of the subtasks involved in answering the question. The bottom window presents the results; in the upper part the extracted answer strings (at most 50 bytes long) are listed and by clicking on them the answer document is displayed. Words occurring in the answer are high-lightened by reversing foreground and background color, and words occurring in the question are displayed in bold face; this is done to facilitate the search for justifications of the extracted answer.

3 Main Task

The main QA task in TREC-10 is similar to the main tasks in TREC-8 and TREC-9. The document set consists of data sets taken from Disks 1–5 of the TIPSTER/TREC document CDs. 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. At least one and no more than five ranked responses per question ranked were to be returned for each question, where the first response is to be preferred over the other responses. A response is either a [answer-string, docid] pair or the string "NIL," where the answer-string may contain no more than 50 bytes and the docid must be the id of a document in the collection that supports the answer-string as an answer.

An [answer-string, docid] pair is judged *correct* if the answer-string contains an answer to the question, the answer-string is responsive to the question, and the document supports the answer. 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" and the pair will only contribute towards the "lenient" score, not to the "strict" score. Otherwise, the pair is judged incorrect.

As with TREC-8 and TREC-9, the score assigned to each question is the reciprocal of the rank for the first response to be judged correct, or 0 if no response is judged correct. The total score for a run is the mean reciprocal rank (MRR) over all questions.

3.1 Submitted Runs

We submitted three runs for the main task (UAmsT10qaM1, M2, and M3). Each of our runs employed the Tequesta system, which was given a total of 979,678 documents to index. The runs differed along 2 dimensions: the number of documents used as input for the answer selection process (either 25 or 50 documents), and the size of the text segments that were used to match the question during the answer selection process (either a single sentence or 2 consecutive sentences); see Table 3.

3.2 Results and Discussion

Of the 500 questions that were originally released, eight questions were removed from the evaluation due to various problems with those questions. Table 3 summarizes the statistics for each of our three submitted runs (UAmsT10qaM1, M2, and M3) over the (remaining) 492 questions.

Table 3: Summary of the results for the main task.

UAmsT10qa	M1	M2	M3
Top documents used	25	50	25
# Sentences in segments	1	1	2
MRR strict	0.185	0.183	0.190
MRR lenient	0.197	0.196	0.203

As Table 3 indicates, it is unlikely that there are significant differences between the MRRs for the three runs that we submitted for the main task. Despite this, we took a closer look at the difference between UAmsT10qaM2 and UAmsT10qaM3. We first ordered the questions with respect to the individual reciprocal ranks from run UAmsT10qaM2 and, in case they were identical, with respect to the question's id. Then, we marked the extent to which run UAmsT10qaM3 differs from run UAmsT10qaM2 for each question. Figure 5 shows the differences for the first 164 ordered questions.

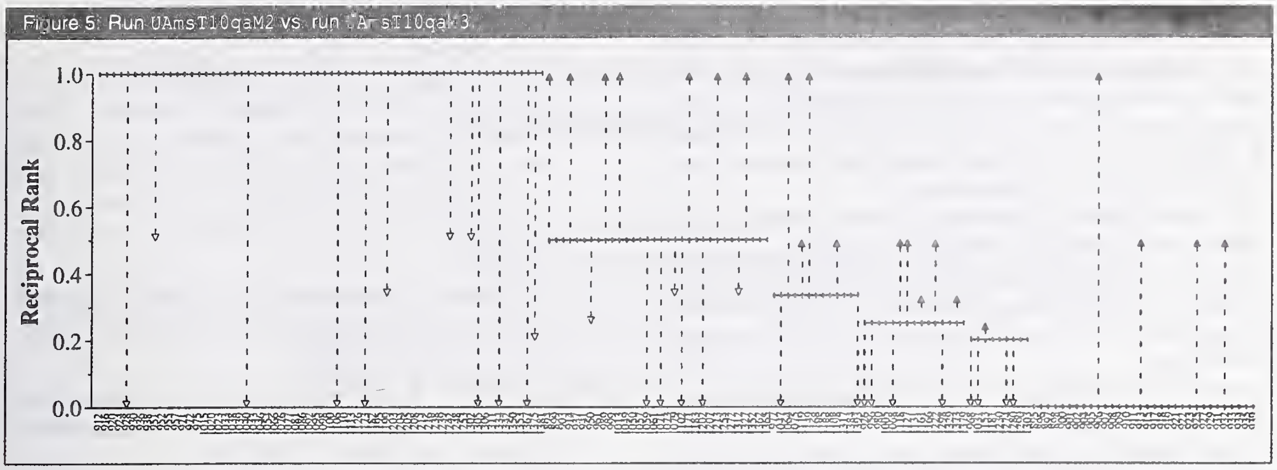
Although the overall effectiveness of run UAmsT10qaM3 increased by only 3.86% in comparison to run UAmsT10qaM2, it is by no means consistently spread over the questions. For many questions there is a severe decrease in effectiveness. What causes this decrease for some questions is not clear to us at the moment, but we hope to gain further insights by analyzing the results more carefully.

Table 4: Analysis of the scores for UAmsT10qaM3.

Question class	#	MRR	Diff.	Rel. Cont.
name	9	0.111	−41.5%	0.002
pers-def	3	0	−100%	0
thing-def	110	0.254	+33.8%	0.057
pers-ident	22	0.167	−12.3%	0.007
thing-ident	107	0.196	+3.30%	0.043
number	35	0.267	+40.4%	0.019
expand-abbr	4	0.125	−34.2%	0.001
find-abbr	0	N/A	N/A	N/A
agent	21	0.071	−62.4%	0.003
object	18	0.069	−63.5%	0.003
known-for	0	N/A	N/A	N/A
also-known-as	11	0.273	+43.5%	0.006
name-instance	2	0	−100%	0
location	27	0.272	+43.0%	0.015
date	41	0.250	+31.6%	0.021
reason	4	0	−100%	0
what:X	71	0.093	−50.8%	0.013
unknown	7	0	−100%	0
Total	492	0.190		

Table 4 provides a closer look at our best run for the main task, UAmsT10qaM3, and a breakdown in terms of the individual question types. Column 1 lists the question classes as discussed in Section 2.4; column 2 lists how many of the 492 questions belonged to a particular class. According to our question classifier two classes did not have any questions in this year's set of questions: find-abbr and known-for. Column 3 lists the mean reciprocal rank for each class of ques-

Figure 5: Run UAmsT10qaM2 vs. run UAmsT10qaM3.



tions. Column 4 ("*Diff.*") records the relative difference between the MRR for the class and the overall MRR for the run, while column 5 ("*Rel. Con.*") indicates the relative contribution of the question class.

The relative contribution of a question class is the MRR for the class multiplied by the proportion of the questions in that class. For example, if a class has an MRR of 0.25, and 10% of all the questions were in that class, the relative contribution would be $0.25 \times 0.10 = 0.025$. For development purposes it can be especially helpful to record differences in MRR and/or relative contribution. Differences in MRR give an indication of how well a question class was handled. Changes in relative contribution give an indication of how much this matters, and therefore where efforts should be focussed to alter the system's performance.

It is clear from Table 4 that our overall score for UAmsT10qaM3 is strongly positively influenced by our scores on the following classes: thing-def, thing-ident, number, location, and date, while our performance on pers-ident, agent, object, and, especially, what:X, contributed negatively towards our overall score.

4 List Task

TREC-10 featured a new task, the QA list task, where answers are to be collected from multiple documents. The list task consisted of 25 questions in the same format as the main task. Each list question specifies a number of instances to be retrieved; e.g., 10 flavors of ice cream in question 11, shown in (10).

(10) Name 10 different flavors of Ben and Jerry's ice cream.

Participants were not allowed to return more instances than specified in the question.

We modified Tequesta only minimally for this task. Since questions in the list task are typically looking for instances of some description, all questions were classified as what:X type questions. The major difference with the main task is that answers are collected from several documents. When compiling the list of answers we checked for duplicates and near duplicates by using simple techniques such as word overlap while ignoring stop words.

In the list task, the answers returned are not ranked. Performance is measured in terms of accuracy, which is computed as the number of distinct correct instances divided by the number of instances requested in the question. Table 5 summarizes the results for the two submitted runs.

Table 5: Summary of the results for the list task.

Run	Avg. Accuracy
UAmsT10qaL1	0.12
UAmsT10qaL2	0.13

The strategies for run UAmsT10qaL1 and UAmsT10qaL2 only differ minimally from each other. Run UAmsT10qaL1 uses the top 50 documents to compile the answer list whereas run UAmsT10qaL2 uses the top 25 documents. This similarity between the runs probably also explains the small difference in performance (+8.33%).

5 Conclusions

In this paper we presented our question answering system Tequesta and evaluated its performance in the TREC-10 QA task. Clearly, Tequesta is still in its early stages and our participation in the TREC-10 QA task was very helpful in revealing aspects that need additional attention in future developments of the system. Most of the shortcomings were already discussed in more detail throughout the paper and we will just summarize some of them here.

First, the underlying information retrieval system FlexIR that was used for pre-fetching is not tuned for the overall task of question answering. Integrating further constraints into the retrieval process, such as phrase-indexing, locality, and Boolean operators, might help in formulating more structured queries that will increase the density of documents containing an answer in the set of top documents.

One of the main problems of the named entity recognizer was that it does not allow for multiple semantic types, which results in a high error rate when using gazetteers to assign certain semantic types, such as locations and person names. In addition, we plan to include the annotated semantic types into the index which is used for retrieval.

Of course, improving the answer selection component remains the main challenge. Table 4 shows that there are significant differences in performance between the question types. Especially the performance for questions of type `agent`, `object`, and `what:X` is far below the average performance of the system.

In this year's participation, we did not spend much time or effort on customizing Tequesta for the list task, but we plan to further develop this aspect of our question answering system, as the problem of fusing information from different sources — in QA as well as in related areas such as multi-document fusion [12] — strikes us as an interesting challenge.

Acknowledgments

Christof Monz was supported by the Physical Sciences Council with financial support from the Netherlands Organization for Scientific Research (NWO), project 612-13-001. Maarten de Rijke was supported by the Spinoza project 'Logic in Action' and by grants from NWO under project numbers 612-13-001, 365-20-005, 612.069.006, 612.000.106, and 220-80-001.

References

[1] J. Allan. NLP for IR. Tutorial presented at NAACL/ANLP language technology joint conference, April 29, 2000.

[2] R.H. Baayen, R. Piepenbrock, and L. Gulikers. The CELEX lexical database (release 2). Distributed by the Linguistic Data Consortium, University of Pennsylvania, 1995.

[3] D. Bikel, R. Schwartz, and R. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 1(3):211–231, 1999.

[4] C. Buckley, A. Singhal, and M. Mitra. New retrieval approaches using SMART: TREC 4. In *Proceedings TREC-4*, pages 25–48, 1995.

[5] J. Fagan. *Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and Non-Syntactic Methods*. PhD thesis, Department of Computer Science, Cornell University, 1987.

[6] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[7] G. Galbiati. A phrase-based matching function. *Journal of the American Society for Information Science (JASIS)*,, 42(1):36–48, 1991.

[8] S. Harabagiu, J. Burger, C. Gardie, V. Chaudri, R. Gaizauskas, D. Israel, C. Jacquemin, C.-Y. Lin, S. Maiorano, G. Miller, D. Moldovan, B. Ogden, J. Prager, E. Riloff, A. Singhal, R. Shrihari, T. Strzalkowski, E. Voorhees, and R. Weishedel. Issues, tasks, and program structures to roadmap research in question & answering (Q&A). URL: `http://www-nlpir.nist.gov/projects/duc/roadmapping.html`, October 2000.

[9] J.B. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11(1–2):22–31, 1968.

[10] C. Macleod, R. Grishman, A. Meyers, L. Barrett, and R. Reeves. NOMLEX: A lexicon of nominalizations. In *Proceedings EURALEX'98*, 1998.

[11] C. Mikheev, M. Moens, and A. Grover. Named entity recognition without gazetteers. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL'99)*, pages 1–8, 1999.

[12] C. Monz. Document fusion for comprehensive event description. In M. Maybury, editor, *Proceedings of the ACL 2001 Workshop on Human Language Technology and Knowledge Management*, 2001.

[13] C. Monz and M. de Rijke. University of Amsterdam at CLEF 2001. In *Proceedings of the Cross Language Evaluation Forum Workshop (CLEF 2001)*, pages 165–169, 2001.

[14] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[15] J. Prager, E. Brown, A. Coden, and D. Radev. Question-answering by predictive annotation. In *Proceedings ACM SIGIR 2000*, pages 184–191, 2000.

[16] B. Santorini. *Part-of-speech tagging guidelines for the Penn Treebank*. Department of Computer Science, University of Pennsylvania, 3rd revision, 2nd printing edition, 1990.

[17] H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, 1994.

[18] T. Strzalkowski. Natural language information retrieval. *Information Processing & Management*, 31(3):397–417, 1995.

[19] E.M. Voorhees. Overview of the TREC-9 question answering track. In *Proceedings TREC-9*, 2001.

[20] E.M. Voorhees and D.M. Tice. The TREC-8 question answering track evaluation. In *Proceedings TREC-8*, pages 83–105, 2000.

[21] E. Williams. On the notions 'lexically related' and 'head of a word'. *Linguistic Inquiry*, 12:245–274, 1981.

Translation Term Weighting and Combining Translation Resources in Cross-Language Retrieval

Aitao Chen[*] and Fredric Gey[†]
[*]School of Information Management and Systems
[†]UC Data Archive & Technical Assistance (UC DATA)
University of California at Berkeley, CA 94720, USA
aitao@sims.berkeley.edu, gey@ucdata.berkeley.edu

1 Introduction

In TREC-10 the Berkeley group participated only in the English-Arabic cross-language retrieval (CLIR) track. One Arabic monolingual run and four 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 bilingual dictionaries and machine translation software. The five official runs are named as BKYAAA1, BKYEAA1, BKYEAA2, BKYEAA3, and BKYEAA4. The BKYAAA1 is the Arabic monolingual run, and the rest are English-to-Arabic cross-language runs. The same logistic regression based document ranking algorithm without pseudo relevance feedback was applied in all five runs. We refer the readers to the paper in [1] for details.

2 Test Collection

The document collection used in TREC-10 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 25 English topics with Arabic and French translations. A topic has three tagged fields, *title*, *description*, and *narrative*. The newswire articles are encoded in UTF-8 format, while the topics are encoded in ASMO 708. The cross-language retrieval task is to search the English topics against the Arabic documents and present the retrieved documents in ranked order.

3 Preprocessing

Because the texts in the documents and topics are encoded in different schemes, we converted the documents and topics to Windows 1256 code. We created a stoplist of 1,131 words using two sources. First, we translated our English stopword list to Arabic using the *Ajeeb* online English-Arabic dictionary. Second, we garnered some of the stopwords from the Arabic-English glossary published in *Elementary Modern Standard Arabic*.

A consecutive sequence of Arabic letters, except for the punctuation marks, was recognized as a word. The words that are stopwords were removed when the documents and topics were indexed. The tokens were normalized by removing the initial letter و, the final letter ة, and the initial letters ال. In addition, the letters أ and إ were changed to the letter ا. The marks above or underneath the letter ا in أ, أّ, إٍ, اَ, أَ, اِ, ■, آ, if present, were also removed.

Arabic has a definite article, but no indefinite articles. The definite article 'al-' is sometimes attached to a word as a prefix. In addition to the singular and plural forms, Arabic also has a form called *dual* which is formed by adding the suffix -$\bar{a}n$. The plurals have regular (also called *sound*) and irregular (also called *broken*) forms. However, the irregular forms are very common, and it is difficult to predict execept that there exist several commonly occurring patterns. The regular plural is formed by adding the suffix $-\bar{u}n$ for the masculine and $-\bar{a}t$ for the feminine form. In Arabic, the adjectives modifying plural nouns also have to be in plural form. Arabic has only two genders, masculine and feminine. The feminine is formed from masculine nouns and adjectives by adding the suffix $-a$.

Since neither of the authors really knows Arabic, it is difficult to write a linguistically motivated Arabic stemmer. One of us learned a little Arabic during the course of participating in this English-Arabic cross-language track and wrote a simple stemmer to remove the definite article $al-$ from the definite nouns, the suffix $\bar{a}n$ from nouns in *dual* form, $-\bar{u}n$ from masculine plural nouns, $-\bar{a}t$ from feminine plural nouns, and suffix $-a$ from feminine noun. Here we assumed that the categories (i.e. part of speech) of words are known. Unfortunately we do not have the part of speech for each word in the collection, nor do we have a part of speech tagger to tag the words. So we cannot simply apply the rules described here. We took a data-driven (i.e, corpus-based) approach to stemming. First we collected all the words in their original form from the document collection. Then we applied each of the rules to the list of Arabic words. For example, to remove the suffix $-\bar{u}n$ from masculine plural nouns, we remove the suffix $-\bar{u}n$ from a word if both the word with the suffix $-\bar{u}n$ and the word without the suffix $-\bar{u}n$ occur in the document collection. Because a word ends with the letters $-\bar{u}n$ is not necessary a masculine plural noun, it is possible to remove the suffix $-\bar{u}n$ from a word incorrectly. The same mistake may also be committed in applying other stemming rules. Our stemming, despite being simple and imperfect, brought an improvement of 9.4% in overall precision for the Arabic monolingual retrieval over the baseline run without stemming.

4 Query Translation

Our approach to cross-language retrieval is to translate the English topics into Arabic, and then search the translated Arabic topics against the Arabic documents.

4.1 Translation Resources

Two online English-Arabic bilingual dictionaries and one online machine translation system were utilized in translating the English topics into Arabic in our cross-language retrieval experiments. The first online English-Arabic dictionary is publicly accessible at http://dictionary.ajeeb.com/en.htm. We will refer to this dictionary as the *Ajeeb* dictionary. The English-Arabic machine translation system is also available from http://dictionary.ajeeb.com/en.htm. The second one is the *Ectaco* dictionary publicly available at http://www.get-together.net/.

4.2 Translation Term Selection

Each word in the English topics was submitted to both English-Arabic online dictionaries. The translations from both dictionaries were merged to form the translation for the English word. To use the Ectaco Arabic-English dictionary, one has to enter nouns in the singular form, verbs in the infinitive form, and adjectives in their positive form. Before we submitted each word as a query to the Ectaco online dictionary, we normalized the English words using an English morphological analyzer [2]. The Ajeeb Arabic-English dictionary can take un-normalized words as input. All the Arabic translations for an English word were sorted and ranked by their occurrence frequency in the Arabic document collection. The top-ranked Arabic translations, but not more than five, an English word were retained as the translation of the English word.

4.3 Translation Term Weighting

After term selection, the term frequency of a source English word in the original query was distributed among the Arabic translations of the English word according to their occurrence frequency in the Arabic collection. The weight assigned to an Arabic translation is proportional to its occurrence frequency in the document collection. That is,

$$qtf_{ai} = qtf_e * \frac{ctf_i}{\sum_{j=1}^{n} ctf_j} \tag{1}$$

where qtf_e is the within-query term frequency of the English word e, ctf_i is the within-collection term frequency of the ith Arabic translation, qtf_{ai} is the weight assigned to the ith Arabic translation, and n is the number of translations retained for the source English word. For the word *education*, the five translations that occur most frequently in the document collection are shown in the second column in table 1. Column 3 in the table shows the number of times each Arabic translation is found in the Arabic collection, and the last column the weight assigned to each of the Arabic translations of *education*, assuming *education* occurs only once in the original English topics. Otherwise, the translation weight is multiplied by the term frequency of *education* in the original query.

	Arabic Translation	Frequency in Collection	Translation Weight
1	بحث	15,183	0.35
2	دراس	11,185	0.25
3	ثقاف	6,484	0.15
4	خبر	5,527	0.13
5	تعليم	5,500	0.13

Table 1. The top-ranked five Arabic translations for *education*.

5 Experimental Results

The official runs we submitted are summarized in table 2. The BKYAAA1 is our only Arabic monolingual run in which all three topic fields were indexed, stopwords removed from both topics and documents, and remaining words stemmed. The BKYEAA2 run used only the machine translation to translate the English topics to Arabic, while the BKYEAA3 used the online dictionaries only to translate the English topics into Arabic. For the other two runs, BKYEAA1 and BKYEAA4, the English topics were separately translated into Arabic using the machine translation system and the bilingual dictionaries first, then their translations were merged before being searched against the Arabic document collection. The only difference between BKYEAA4 and BKYEAA1 is that the former indexed only the title and description fields, where as the latter indexed all three topic fields.

Table 3 shows the overall precision for the five runs. There are a total of 4,122 relevant documents for all 25 topics. As mentioned above, all five runs were performed without pseudo relevance feedback. Our best cross-language performance is 85.68% of the monolingual performance. The queries translated from the combined online dictionaries substantially outperformed those translated from the machine translation system. We believe that the superior performance of the combined dictionaries could be attributed in part to the fact that up to five translation terms from the online dictionaries were retained for the source words while the machine translation system retained only one translation for each source word.

Run ID	Type	Topic Fields	Translation Resources
BKYAAA1	Arabic Monolingual	Title,Description,Narrative	
BKYEAA1	English-to-Arabic	Title,Description,Narrative	Dictionaries and MT
BKYEAA2	English-to-Arabic	Title,Description,Narrative	MT
BKYEAA3	English-to-Arabic	Title,Description,Narrative	Dictionaries
BKYEAA4	English-to-Arabic	Title,Description	Dictionaries and MT

Table 2. Summary of official runs.

recall level	BRKAAA1 (MONO)	BRKEAA1 (CLIR)	BRKEAA2 (CLIR)	BRKEAA3 (CLIR)	BKYEAA4 (CLIR)
at 0.0	0.8432	0.7803	0.7133	0.7052	0.7372
at 0.1	0.6174	0.5250	0.4374	0.5119	0.4901
at 0.2	0.4582	0.3970	0.3229	0.4418	0.3807
at 0.3	0.3716	0.3241	0.2752	0.3463	0.2967
at 0.4	0.3021	0.2627	0.2265	0.2870	0.2493
at 0.5	0.2487	0.1967	0.1780	0.2257	0.2026
at 0.6	0.1959	0.1309	0.1290	0.1490	0.1437
at 0.7	0.1604	0.0945	0.0861	0.1206	0.1134
at 0.8	0.1200	0.0620	0.0588	0.0915	0.0874
at 0.9	0.0701	0.0121	0.0170	0.0240	0.0200
at 1.0	0.0141	0.0014	0.0015	0.0141	0.0200
average precision	0.2877	0.2337	0.2006	0.2465	0.2316
relevant retrieved	2,393	2,579	2,485	2,490	2,300
% of mono		81.23%	69.73%	85.68%	80.50%

Table 3. Evaluation results for one Arabic monolingual run and three English to Arabic cross-language retrieval runs.

A number of additional experimental runs were performed and evaluated locally to show the effect of various aspect of preprocessing on the retrieval performance. We broke down the preprocessing of the texts into three steps: stopwords removal, word normalization, and word stemming. Table 4 presents the overall precision and recall by incrementally adding more features into the preprocessing of the Arabic texts. The overall precision was .1581 when no preprocessing was performed at all. That is, no words were removed from indexing, words were not normalized and stemmed. When stopwords were removed from indexing, the overall precision increased to .2046, and when words were normalized as described above the overall precision was substantially improved. Further improvement was shown by stemming the words even though our stemmer was rather simple. Many more possible word form changes were not considered at all in our stemmer. The very simple normalization of words brought 28.54% improvement in overall precision over the run without word normalization. The results presented in table 4 leads us to believe that further gain in overall precision could be achieved by using a more sophisticated Arabic stemmer or morphological analyzer. All three topic fields were indexed in the runs shown in table 4. Our official monolingual run, BKYAAA1, included all three steps in preprocessing. The overall recall for our official monolingual run was only 58.05%. Besides applying a more sophisticated Arabic stemmer, we believe that pseudo relevance feedback should also improve both overall recall and overall precision.

recall	stoplist	normalization	stemming	precision	recall
baseline	-	-	-	0.1581	1594/4122
mono1	+	-	-	0.2046	1930/4122
mono2	+	+	-	0.2630	2333/4122
BKYAAA1	+	+	+	0.2877	2393/4122

Table 4. Arabic monolingual retrieval performance.

For the runs, BKYEAA1 and BKYEAA4, the separately translated topics using online dictionaries and online machine translation system were merged before being searched against the Arabic collection. We also experimented with linearly combining the ranked lists produced in searching the translated topics separately against the Arabic documents. That is, we first ran the dictionary-translated topics against the Arabic documents, and the machine translation system-translated topics against the Arabic documents. Then we merged the two ranked lists by averaging the probabilities of relevance. The overall precision for the long queries increased from .2337 of BKYEAA1 to .2552, a 9.20% improvement.

6 Conclusions

In summary, we performed four English-Arabic cross-language retrieval runs and one Arabic monolingual run, all being automatic. We took the approach of translating queries into document language using two online dictionaries and one machine translation system. Our best cross-language retrieval run achieved 85.68% of the monolingual run. Furthermore, our cross-language run using online bilingual dictionaries substantially outperformed the run using an online machine translation system. All of our runs had low overall recall, which we believe could be in part attributed to our failure to conflate the various forms of the words to their stems. Even though the preprocessing was quite simple, it substantially improved the overall precision and recall over the baseline run without any preprocessing at all. We believe that further improvement could be achieved by applying a more sophisticated Arabic stemmer and pseudo relevance feedback.

7 Acknowledgements

This work was supported by DARPA (Department of Defense Advanced Research Projects Agency) under research contract N66001-97-C-8541, AO-F477.

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.

Comparing explicit and implicit feedback techniques for web retrieval: TREC-10 interactive track report

Ryen W. White [1], Joemon M . Jose [1] and I. Ruthven[2]

[1]Department of Computing Science
University of Glasgow, Glasgow, G12 8QQ. Scotland
whiter, jj@dcs.gla.ac.uk

[2]Department of Computer and Information Sciences
University of Strathclyde, Glasgow G1 1XH. Scotland
Ian.Ruthven@cis.strath.ac.uk

1. Introduction

In this paper we examine the extent to which implicit feedback (where the system attempts to estimate what the user may be interested in) can act as a substitute for *explicit* feedback (where searchers explicitly mark documents relevant). Therefore, we attempt to side-step the problem of getting users to explicitly mark documents relevant by making predictions on relevance through analysing the user's interaction with the system. Specifically, we hypothesised that implicit and explicit feedback were interchangeable as sources of relevance information for relevance feedback. Through developing a system that utilised each type of feedback we were able to compare the two approaches in terms of search effectiveness.

2. Systems

Our basic experimental system is a generic interface that can connect to any web search engine. In our experiments we use the interface to connect to the Google search engine. The interface is based on a summarisation interface developed for investigating web search behaviour, [WJR01, WRJ01]. The system we developed for the experiments in this paper also incorporates a component that displays sentences from the retrieved set of web pages. These sentences are ones that have a high degree of match with the user's query. The set of sentences and the ranking of the sentences automatically updates in the presence of relevance information from the user (relevance feedback). That is the content of the summary is used to form a new query which is used to create a new list of important sentences for display to the user.

Two interfaces were developed; one which uses explicit feedback and one which uses implicit feedback. The explicit feedback interface had checkboxes to allow users to explicitly mark documents relevant; the implicit interface assumed that any document for which a summary was requested was of interest to the user. Our experimental hypothesis was the degree to which the implicit interaction could substitute for the explicit relevance assessments. Details of the systems can be found in [WRJ02].

3. Experimental details

In total, 16 subjects participated in our experiments. All subjects were educated to graduate level in a non-computing, non-LIS discipline, with three exceptions, all our subjects were recruited from the Information Technology course at the University of Glasgow. All users, with one exception, used the Internet on a regular basis.

The average age of the subjects was 24.75 with a range of 11 years. Most users used computers and the Internet frequently – the average time spent online per week was 14 hours. With three exceptions, all users cited Google as amongst their favourite search engines.

Figure 1 shows the tasks we used in the experiments.

Medical

- Find a website likely to contain reliable information on the effect of second-hand smoke.
- Tell me three categories of people who should or should not get a flu shot and why.
- List two of the generally recommended treatments for stomach ulcers.
- Identify two pros or cons of taking large doses of Vitamin A.

Buying

- Get two price quotes for a new digital camera (3 or more megapixels and 2x or more zoom).
- Find two websites that allow people to buy soy milk online.
- Name three features to consider in buying a new yacht.
- Find two websites that will let me buy a personal CD player online.

Travel

- I want to visit Antarctica. Find a website with information on organized tours/trips there.
- Identify three interesting things to do during a weekend in Kyoto, Japan.
- Identify three interesting places to visit in Thailand.
- I'd like to go on a sailing vacation in Australia, but I don't know how to sail. Tell me where can I get some information about organized sailing cruises in that area.

Project

- Find three articles that a high school student could use in writing a report on the Titanic.
- Tell me the name of a website where I can find material on global warming.
- Find three different information sources that may be useful to a high school student in writing a biography of John F. Kennedy.
- Locate a site with lots of information for a high school report on the history of the Napoleonic wars.

Figure 1 : Tasks used in TREC-10 interactive track experiments

Users were allowed a maximum of 10 minutes for each task. They were asked to use the system presented to them (either implicit or explicit, depending on the particular Greco-

Latin square allocation) to search the Internet and attempt to find an answer to the task set. Users were allowed to browse away from the result list to any degree.

4. Results & Analysis

Most of the data used to assess search effectiveness came from the logs generated by the system during the experiments.

4.1 Number of Results Pages Viewed

The total number of result pages viewed and queries submitted during all the experiments was recorded. Table 1 shows the average results per user obtained.

Variation	Number of result pages	Number of query iterations
Implicit	3.375 *	3.5625
Explicit	2.5 *	2.625

* users occasionally refined query before result page appeared, so result pages ≠ query iterations

Table 1: Average result page views and query iterations per user

These differences are not significant using a Mann-Whitney Test at $p \lesssim 0.05$ ($p = 0.234$). Our system gave access to the first 30 documents retrieved by the underlying search engine, and in many cases this was sufficient to complete the tasks. This meant that there was no real need for users to browse to the next 30 results (i.e. results 30 to 60 in standard search engines). The lack of a significant difference between the 'implicit' and 'explicit' systems shows that the type of system used does not affect the number of result pages viewed or query iterations needed.

4.2 Task Completion

As part of the post-task questionnaire users were asked whether they felt they had successfully completed the task just attempted, it is these results that are presented in Table 2. The choice of whether a task was complete was left up to the user. It was thought that this best reflected real-world retrieval situations. However, the experimenter was occasionally asked to verify the correctness of the results obtained. Table 2 shows these results (out of 64).

Again these results are not significant using a Mann-Whitney Test at $p \lesssim 0.05$ ($p = 0.361$). There is no significant difference between the number of tasks that users completed on the 'implicit' and the 'explicit' systems.

Variation	Number of tasks completed
Implicit	61
Explicit	57

Table 2: Number of tasks completed

4.3 Task Times

The time taken to complete tasks on both systems was measured. When a task was incomplete, a figure of 600 seconds (10 minutes) would be recorded by the system. This was the time limit imposed on each task and users were not allowed to work past this. In Table 3 we can see these results.

Variation	Average time per task (secs)
Implicit	372.29
Explicit	437.43

Table 3: Average time per task

Again these are not significant using a Mann-Whitney Test at $p \leq 0.05$ ($p = 0.228$).

From an analysis of the log files we were able to establish that no significant difference existed between the two variations. This appears to add a little weight to our claim that perhaps the 'implicit' and 'explicit' feedback are at least to some degree substitutable, although factors such as the similarity of the interface design may be important too. If the results obtained were significant we could suggest that one type of system promotes search effectiveness more than the other. In this case, there is no significant difference, and it is safe to assume that some degree of substitutability does indeed exist.

Further results and analysis are reported in [WRJ02].

Acknowledgements

We would like to thank fellow members of the Information Retrieval group for their thoughts and assistance. We would also like to thank those who participated in the experiments, their comments and enthusiasm were greatly appreciated.

References

[WJR01]White, R., Jose, J.M. and Ruthven, I. *Query-Biased Web Page Summarisation: A Task-Oriented Evaluation*. Poster Paper. Proceedings of the 24th Annual International

ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '01) New Orleans, USA, 9-13 September 2001.

[WRJ01] White, R., Ruthven, I. and Jose, J.M. *Web document summarisation: a task-oriented evaluation.* International Workshop on Digital Libraries. Proceedings of the 12th International Database and Expert Systems Applications Conference (DEXA 2001) Munich, Germany, 3-7 September 2001.

[WRJ02] White, R., Ruthven, I. and Jose, J.M. *The use of implicit evidence for relevance feedback in web retrieval*. Proceedings of the Twenty-Fourth European Colloquium on Information Retrieval Research (ECIR '02). Lecture Notes in Computer Science. Glasgow. 2002. *in press*.

Learning Components for A Question-Answering System

Dan Roth
danr@cs.uiuc.edu
Gio Kao Kao, Xin Li, Ramya Nagarajan,
Vasin Punyakanok, Nick Rizzolo, Wen-tau Yih
Department of Computer Science

Cecilia Ovesdotter Alm, Liam Gerard Moran
Department of Linguistics

University of Illinois at Urbana-Champaign

Abstract

We describe a machine learning approach to the development of several key components in a question answering system and the way they were used in the UIUC QA system.

A unified learning approach is used to develop a part-of-speech tagger, a shallow parser, a named entity recognizer and a module for identifying a question's target. These components are used in analyzing questions, as well as in the analysis of selected passages that may contain the sought after answer.

The performance of the learned modules seems to be very high, (e.g., mid 90% for identifying noun phrases in sentences), though evaluating those on a large number of passages proved to be time consuming. Other components of the system, a passage retrieval module and an answer selection module, were put together in an ad-hoc fashion and significantly affected the overall performance. We ran the system only over about 60% of questions, answering a third of them correctly.

1 Introduction

The QA system described in this paper is developed as a platform for studying and experimenting with a unified method for learning, knowledge representation and inference, required to perform knowledge intensive natural language based inferences.

Our 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 a knowledge representation support that allows, for example, to keep track of predictions as input to higher level predictors 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 of several of these predictors along with some constraints, e.g., those that are implied by the questions.

The system developed here makes some preliminary steps in these directions by putting forward a suggestion for a few of the learning components and placing them, for evaluation purposes, within a question answering system. We describe several key components in a question answering system, developed within a unified learning approach that makes use of a relational representation language. Some of the components presented also incorporate domain and task specific constraints as part of a general scheme for inference with outcomes of classifiers that, we believe, could have a more general use.

The learning components used here include a POS tagger, a shallow parser and a named entity recognition module (which is essentially the same module, only trained differently) and a module for identifying a question's target. These components were used in analyzing questions, as well as in the analysis of selected passages that may contain the sought after answer.

This project started as a summer project in early June, 2001. The reliance on learning methods allowed us to put together a system for this task in about six weeks. Needless to say, there are several important components that are still missing. The main part missing from our current approach, due to lack of time, is a learned module for selecting an appropriate answer given a candidate passage and the constraints identified in the question analysis. During the work on this task we realized that, given the vast amount of text available, there is almost always a "simply structured" correct answer among the large number of correct answers that exist in a corpus. This makes the task very different from the *story comprehension* task [5] because simple heuristics that rely on the existence of a simply structured answer can already give reasonable results; due to lack of time, we resorted to these in the current system. A second significant component that is missing is an information retrieval module, which was not in the focus of our study. We used the documents retrieved by TREC and a simple-minded approach to focus on candidate passages within those documents.

This report describes the main learning components (Sec. 2), how these are used in our system (Sec. 3), and some preliminary evaluation of the learning components and the system (sec. 4). We conclude with some questions that pertain to our approach and comments on future plans.

2 Learning Components for a QA system

A robust and accurate question answering system depends on a large number of classifiers that will be used at different levels of the process and will support a variety of functions, from knowledge acquisition to decision making and the integration of information sources to yield robust decisions.

In this project, we used a unified methodology to develop and study a few learning components that we believe are necessary. Each of our learning modules consists of a stage of generating expressive relational features (that could rely on previously learned predictors or knowledge acquired otherwise), learning using a network of linear classifiers over these, and an inference process that makes use of the outcome of the classifiers to make decisions that relate some domain and task specific constraints.

This report describes four learning components that are used in the current QA system. All components make use of the same learning approach and tools. Although several of the components are built on the results of previously learned predictors, learning is always done one stage at a time.

Learning is done using the SNoW learning architecture. SNoW [1, 11] is a multi-class classifier that is specifically tailored for learning in domains in which the potential number of information sources (features) taking part in decisions is very large, of which NLP is a principal example. It works by learning a sparse network of linear functions over a pre-defined or incrementally learned feature space. Typically, SNoW is used as a classifier, and predicts using a winner-take-all mechanism over the activation value of the target classes. However, in addition to the prediction, it provides a reliable confidence level in the prediction, which enables its use in an inference algorithm that combines predictors to produce a coherent inference. (See the descriptions of the components for pointers to the details on how SNoW is being used in each case.)

The input to SNoW is provided by a feature extraction stage. In this stage we use a knowledge representation language [2, 12] to generate expressive relational features over a collection of predicates. These predicates can be computed directly from the input data (e.g., words in a given sentence), previously learned predicates (e.g., pos tags) or predicates that are acquired in other ways (e.g., a semantic class of a word). Predicates of all these types are used in the learning components described below. A programmer needs only to define a small number of feature *types* (RGFs, in [2, 12]) believed to be necessary for representing the classifier, and many features of this type will be generated in a data driven manner, as input sentences are observed. In all problems discussed below, the number of potential features generated will be very large, but the learning approach is capable of learning in the presence of a large number of potential features.

Several of the components described below already include ways of interaction among classifiers. These include the sequential model [3], used for the pos tagger, as well as the question classifier and the CSCL shallow parser [10], used for shallow parsing and name entity recognition.

2.1 Part-Of-Speech tagger

The POS tagger used here is the one developed in [3]. This is a SNoW based pos tagger 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 [3], it achieves state-of-the-art results on this task and is significantly more efficient than other part-of-speech taggers. The tagger and a demo of it are available at `http://L2R.cs.uiuc.edu/~cogcomp`

2.2 Sentence Analysis

Our sentence analysis makes use of an *inference with classifiers* paradigm as one general method for identification of phrases in sentences. The same method, a SNoW-based CSCL parser [10, 9], is used by both the shallow parser and the name entity analyzer.

In CSCL (constraint satisfaction with classifiers), SNoW is used to learn several different word level classifiers – each detects the beginning or end of a phrase of some type (noun phrase, verb phrase, a location phrase, etc.). These classifiers are learned as a function of the words and pos tags in the context of the target word. The outcomes of these classifiers are then combined to a sentence level decision in a way that satisfies some constraints – *non-overlapping* constraints in this case – using an efficient constraint satisfaction mechanism that makes use of the confidence in the classifier's outcomes. This method can be used to identify the phrases of one type (as in [10, 9]) or of several different types at the same time [8]. We use two instantiations of this method below, with different training data.

2.2.1 Shallow parsing

Shallow parsing, also known as text chunking, is the task of identifying phrases, possibly of several types, in natural language sentences. It is simpler, conceptually and computationally, than full parsing, but still provides fundamental sentence structure information such as noun phrases and verb phrases. An additional advantage from a learning perspective is that limited training data can still be used to induce a shallow parser for the type of information available.

The shallow parser used here is based on [10]. As mentioned above, for each type of phrase, two learned classifiers are used, one learns to identify the beginning of the phrase, and the other its end. The final prediction is made using a constraint satisfaction based inference, which takes into account constraints such as "Phrases do not overlap".

In Question Analysis, we use this module to identify 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 [7]. When analyzing retrieved passages, in order to save processing time, only noun-phrases and verb-phrases are identified. Below are some examples to the type of processing provided by this module.

Question: *What was the name of the first Russian astronaut to do a space walk?*

> [NP What] [VP was] [NP the name] [PP of] [NP the first Russian astronaut]
> [VP to do] [NP a spacewalk]

Sentence: *The broad-shouldered but paunchy Leonov, who in 1965 became the first man to walk in space, signed autographs.*

> [NP The broad-shouldered but paunchy Leonov] , [NP who] in [NP 1965] [VP became]
> [NP the first man] [VP to walk] in [NP space] , [VP signed] [NP autographs] .

The shallow parser and a demo of it are available at `http://L2R.cs.uiuc.edu/~cogcomp.html`

2.2.2 Recognizing named entity phrases

The named entity recognizer annotates various types of named entities. Unlike other common named entity recognition systems which only annotate proper nouns, dates, time, and other numerical values, our named entity recognizer attempts to further extend the scope of annotated categories and the definition of a "named entity".

In addition to annotating typical categories such as person, organization, location, time, money, date, and percentage, we add several more categories that are typically not proper nouns. Some of the additional categories are title, profession, event, holiday & festival, animal, plant, sport, medical, unit, etc. These extra categories provide more information than a typical entity recognition system and can be viewed more as a step toward semantic categorization. When relevant, we further sub-divide categories into more detailed subclasses (e.g. Location-City, Location-Country).

To achieve this named entity recognition task, we have combined machine learning techniques with a manually based knowledge acquisition process that we used to acquire categorized lists, as well as some specific rules that are used to integrate these. Our plan was to use the categorized list as additional annotation for training but, in the current system, we had enough data to train only of a few of the large categories.

For three major categories, those of person, location, and organization, the SNoW based CSCL approach [10] described above. Phrases of these types were annotated using the categorized lists we generated, and CSCL was used to learn a phrase recognizer for these types of phrases. In evaluation, the list-based process first annotates the data as suggested named entities, and then the classifier uses this suggestion, along with the context in the sentence, to provide a more accurate annotation of the data. Other, smaller categories, are tagged using the lists and some rules that incorporate special keywords and stop lists. By processing the data through several large categorized lists, we are able to annotate the remaining categories. A few of the categories are exhausted by a combination of lists and rules.

It is important to mention that, although we find CSCL a promising approach to this problem, there are several interesting problems that we still need to address here. The most important are the training data problem and that of defining categories (and perhaps hierarchies of) in a satisfactory manner.

2.3 Question Classification

Inspired by many questions answering systems in previous TREC Q/A tracks [4, 6, 13], we believe identifying the target of a question is an important step in an attempt to answer it correctly. When a system is aware of being asked a who-question, it can focus on names or titles as potential answers. However, our working assumption is that it is better to classify questions into finer categories and not rely solely on the head of the question (e.g. what, who, or when). This will enhanced the performance of the system, by allowing it to look for more specific and accurate potential answers. it may also allow for the development of different strategies for answer selection that depend on the fine classification of the question class.

We developed a learning approach to this problem that utilizes the sequential model idea [3] in learning a hierarchy of question classes. The question classes were organized into a two-layered hierarchy, that also allows us to tradeoff the accuracy of the classification with the concreteness of question classes. In particular, classes in the first layer are easier to predict, whereas classes in the second layer provide a more concrete specification of the target answers. We defined *eight* top classes and about *fifty* final classes in this hierarchy. The top classes we use are *Abbreviation*, *Abstract Entity*, *Concrete Entity*, *Description*, *Human*, *Location*, *Number*, *Other Entity*. The final classes include *color*, *language*, *animal*, *sport*, *definition*, *reason*, *city*, *country*, *age*, *date*, *speed*, and so on.

The question classifier is trained using Sequential Model and the SNoW learning architecture. The training set includes about 6,000 questions, consisting of TREC-8 and TREC-9 questions, and other questions that we generated - all manually annotated. Features for the question classifier were generated using the FEX feature extractor [2] and used predicates that include information in the sentence (words and sentence length), previously learned predicates (pos tags, shallow parsing, named entity) and some semantic categorization information acquired using WordNet.

3 System Description

As shown in Fig.1, our QA system consists of three main modules, supported by the learning components described in last section. Using all the learning components, Question Analyzer extracts semantic and syntactic information from a question and stores it in question analysis records. Passage Retriever uses this information to extract relevant passages from the corresponding documents that are retrieved by the search engine. Given the question analysis record and relevant passages, Answer Selector analyzes the documents with the help of POS tagger, NE recognizer, shallow parser, and then finds answers.

3.1 Question Analysis

The goal of Question Analyzer is to transform questions into new representations, which provide further information to the other modules. Tasks that Question Analyzer performs include:

- Question Classification: deciding the types of potential answers by classifying question types. (supported by Question Classifier)

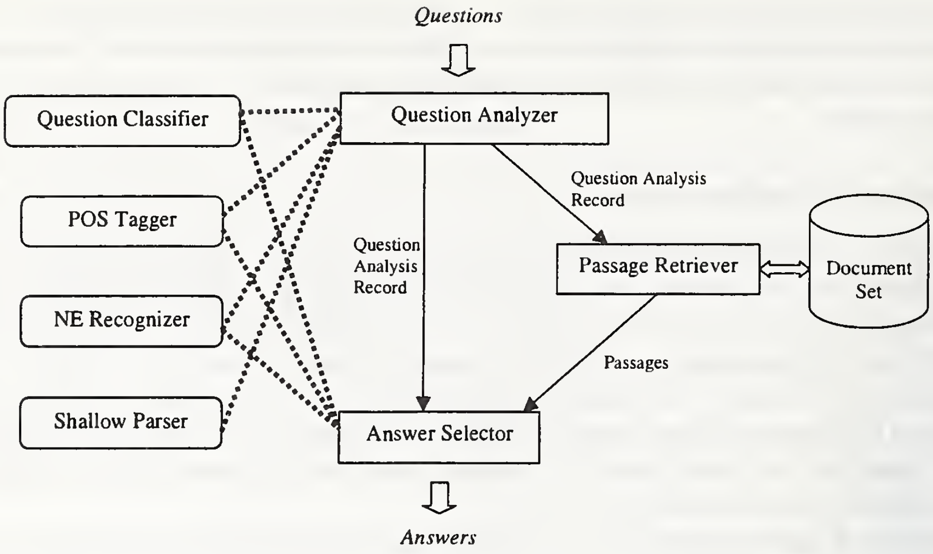


Figure 1: Top level system architecture

- Key Phrase Retrieval: finding keywords in the question.
- Semantic Representation: representing the semantic information from questions. Different types of questions may have different representations.

We describe the last two tasks as follows.

3.1.1 Key Phrase Retrieval

Key phrases are the query terms used in searching relevant documents and passages. In our system, there are seven types of key phrases extracted, including *Quotation*, *Named Entity*, *Noun*, *Verb*, *Extreme Adjective*, *Capitalized Word*, and *Unknown Word*.

Key phrases are extracted sequentially according to the same order. For instance, if a word in the question has been identified as part of a quotation, then it won't be taken into account as other key phrase types.

3.1.2 Semantic Representation

The purpose of acquiring some form of representation for a question is to locate and verify candidate answers. A search engine is used to obtain candidate documents containing search terms. However, the constraints of the semantic relation between those terms haven't been expressed in this process. Thus, we need a semantic representation to represent the real 'meaning' of a question so that we can locate the exact answer in a document and verify its correctness by comparing the representations of the question and an answer candidate.

Syntactic analysis is the underlying step for acquiring semantic representation. We apply POS tagging and shallow parsing to questions. Syntactic analysis can be somewhat easy and can achieve a high accuracy. The Name Entity recognizer is then used to get semantic tags for words and phrases in the question sentence.

It's not clear so far which form of representation is appropriate for this task and easy to acquire at the same time. But it's somewhat clear that we don't need very complicated logic representation, since the major application of it is matching rather than reasoning. As a beginning, we predefined a simple template to represent the knowledge in questions and use simple heuristics to get information defined in the template from questions and answers. Although it is relatively easy to convert a question to a simple format of semantic representation, it seems not enough.

Some fields in the template are: *Answer Entity Type*, *Action*, *Action Subject*, *Direct Object*, *Indirect Object*, *Target Description*, *Target Modifier*, *Action Modifier*, *Location*, *Time*, *Extreme Case*, and *Unit.*

3.2 Passage Retriever

To reduce the search scope of Answer Selector, Passage Retriever picks relevant passages from the 1000 documents that TREC provides for each question. Based on the key phrases extracted by Question Analyzer, Passage Retriever first filters out some documents, then retrieves relevant passages from the preserved documents. Key phrases are not directly used as query terms. Instead, they are expanded by selectively adding synonyms or related words according to the information in WordNet.

Criteria for relevant documents and relevant passages are described as follows.

Document Retrieval A document that is classified as relevant must have all the key phrases or their expansions in it.

Passage Extraction A passage is defined as a short paragraph which contains no more than five consecutive sentences. If all the key phrases or their expansions can be found in these five sentences, then the passage is considered as relevant and returned by Passage Retriever. However, if a key phrase is a full name of a person, then the last name or the first name are both treated as valid synonyms. This is because a document could mention a person's full name in the beginning, and then only use his first name or last name later.

3.3 Answer Selector

Given question analysis records, Answer Selector finds answers from extracted passages. It consists of the following three steps. Sentences in these passages are first analyzed syntactically and semantically by POS tagger, shallow parser, and name entity recognizer. Candidate answers are then located in these processed passages. Finally, each candidate answer is evaluated and ranked. The top five answers are extended or shrunk to satisfy the 50-byte length constraint and returned as final answers.

Although we believe that the robust inference procedure based on learning is the right way for choosing and verifying answers, the current version of our QA system uses only heuristic-based, ad-hoc procedure instead because of limited development time.

Decisions on locating candidate answers strongly rely on the results of the question classifier, name entity recognizer and shallow parser. For example, if a question asks for a person, a location, or some number, then only the phrases belonging to these name entity types will be treated as candidates. For other question classes, a noun-phrase or even a whole sentence will be picked as potential answers.

To rank all the candidate answers, we evaluate the confidence scores based on some heuristic rules. These rules generally test how closely candidate answers match the question in terms of keywords and phrases in different semantic fields. 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 identified from the question.

4 Evaluation

4.1 Evaluation of Learning Components

We first present results on the performance of the learning components as stand alone modules.

4.1.1 POS

The test corpus for our POS tagger is taken from the Penn Treebank WSJ and Brown corpora. It consists of 280,000 words, of which 5,412 are unknown words (words that do not appear in the training corpus). For the known words, the accuracy of our POS tagger is 96.86%, which is slightly better than Brill's POS tagger (96.49%), but the speed is 3000 times faster. For the unknown words, we still have reasonably high accuracy (73.0%). More details of the evaluation can be found in [3].

4.1.2 Shallow Parsing

The evaluation of our shallow parser consists of two parts. The first is to compare it with other shallow parsers, and the second is to compare it with a full sentence parser.

To compare our shallow parser with others, we chose the data used in the chunking competition in CoNLL-2000 [7]. In this competition, a full parse tree is represented in a flat form. The goal in this case is to accurately predict a collection of 11 different types of phrases. The chunk types are based on the syntactic category part of the bracket label in the Treebank. Using exactly the same training and testing data, our shallow parser ranks among the top ones.

Additionally, we also demonstrate that by focusing only on the most significant syntactic information, shallow parsing is not only much faster, but can also achieve more accurate results than the full parser. We design several experiments and compare our shallow parser to Michael Collins' full parser, which is one of the most accurate full parsers. For tasks of phrase identification on both WSJ data and Switchboard data, our parser outperforms Collins' full parser in every experiment. The overall experimental results in terms of F_β value are shown in Table 1. For more details, please refer to [8].

Table 1: **Precision & Recall for phrase identification (chunking)** for the full and the shallow parser on the WSJ data. Results are shown for an average of 10 types of phrases as well as for two of the most common phrases, NP and VP.

	Full Parser			Shallow Parser		
	P	R	$F_{\beta=1}$	P	R	$F_{\beta=1}$
Avrg	91.71	92.21	91.96	93.85	95.45	94.64
NP	93.10	92.05	92.57	93.83	95.92	94.87
VP	86.00	90.42	88.15	95.50	95.05	95.28

4.1.3 Named Entity

Although our named entity recognizer is designed to tag many detailed sub-classes, to make a fair comparison, we test it on the benchmark dataset from MUC-7, which contains only three tags: *Person*, *Location*, and *Organization*. 2000 sentences are used as training data and 430 sentences are used as testing data. The recall-precision results are shown in Tab. 2.

Table 2: Precision & Recall for named entity recognition

	Recall	Precision	$F_{\beta=1}$
Overall	75.97	92.64	83.48
Person	68.50	93.98	79.24
Location	85.75	91.49	88.53
Organization	70.22	93.12	80.06

4.1.4 Question Classifier

We manually label all the 500 questions in TREC-10, and use them as testing data for our question classifier. The classifier performs better on the top-level classes. It achieves 87% when it makes prediction on all questions. Due to the inherent ambiguity of this problem, about 5%-10% questions are difficult to classify in one single class. Therefore, to distinguish hard questions and easy questions, the classifier is restricted to make a prediction only when it has high enough confidence. In this setting, it doesn't make a prediction on 15% to 20% of the questions, but the accuracy of the prediction is enhanced to 93%.

4.2 System Evaluation

Unfortunately, we didn't manage to process all the questions in TREC-10. The main reason is that we were too optimistic about the processing time given a huge set of data. Although the learning components we used, such as the POS tagger and shallow parser, are quite efficient, it still took a lot of time to finish all processing work. In addition, when there were many passages that were considered to be relevant, both Passage Retriever and Answer Selector took a long time to process.

From the 324 processed questions, we answered 108 correctly. In particular, 54 are in rank 1; 23 are in rank 2; 17 are in rank 3; 8 are in rank 4; 8 are in rank 5.

5 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 that need to interact in various ways and be used as part of a reasoning process to yield the desired answer.

This report summarizes some preliminary steps we took in this direction. We built several learning components to facilitate question answering. These components work pretty well independently but still fail short of the supporting a robust overall approach. Some of our future research directions include developing our unified approach further in several directions. These include using it to learn better representations for questions, more efficient syntactic and semantic building blocks and developing robust approaches for unification or reasoning when selecting answers to questions.

References

[1] 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.

[2] C. Cumby and D. Roth. Relational representations that facilitate learning. In *Proc. of the International Conference on the Principles of Knowledge Representation and Reasoning*, pages 425–434, 2000.

[3] 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.

[4] S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus, and P. Morerescu. Falcon - boosting knowledge for answer engines. In *Proceedings of Text REtrieval Conference (TREC-9)*, 2000.

[5] L. Hirschman, M. Light, E. Breck, and J. Burger. Deep read: A reading comprehension system. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, 1999.

[6] A. Ittycheriah, M. Franz, W. Zhu, and A. Ratnaparkhi. Ibm's statistical question answering system. In *Proceedings of Text REtrieval Conference (TREC-9)*, 2000.

[7] 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.

[8] X. Li and D. Roth. Exploring evidence for shallow parsing. In *Proc. of the Annual Conference on Computational Natural Language Learning*, 2001.

[9] M. Munoz, V. Punyakanok, D. Roth, and D. Zimak. A learning approach to shallow parsing. In *EMNLP-VLC'99, the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 168–178, June 1999.

[10] 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. Acceptance Rate: 25/514 (4.8%) Oral Presentations; 152/514 (29%) overall.

[11] D. Roth. Learning to resolve natural language ambiguities: A unified approach. In *Proc. of the American Association of Artificial Intelligence*, pages 806–813, 1998. Acceptance Rate: 143/475 (30%).

[12] D. Roth and W. Yih. Relational learning via propositional algorithms: An information extraction case study. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 1257–1263, 2001. Acceptance Rate: 197/796 (25%).

[13] R. Srihari and W. Li. Information extraction supported question answering. In *Proceedings of Text REtrieval Conference (TREC-8)*, 1999.

TREC-10 Experiments at University of Maryland CLIR and Video

Kareem Darwish,[1] David Doermann, Ryan Jones, Douglas Oard and Mika Rautiainen[2]
Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742

Abstract

The University of Maryland Researchers participated in both the Arabic-English Cross Language Information Retrieval (CLIR) and Video tracks of TREC-10. In the CLIR track, our goal was to explore effective monolingual Arabic IR techniques and effective query translation from English to Arabic for cross language IR. For the monolingual part, the use of the different index terms including words, stems, roots, and character n-grams were explored. For the English-Arabic CLIR, the use of MT, wordlist based translation, and non-dictionary words transliteration was explored. In the video track, we participated in the shot boundary detection, and known item search with the primary goals being to evaluate existing technology for shot detection and a new approach to extending simple visual image queries to video sequences. We present a general overview of the approaches, summarize the results in discuss how the algorithms are being extended.

1 CLIR Track

1.1 Introduction

For the CLIR track, we were interested in testing the effects of the choice of Arabic index terms, the use of morphology, and transliteration of words that are not in the dictionary. To test the effects before the ad-hoc TREC runs, we used a small Arabic collection called Zad. In the ad-hoc experiments we relied on insight gained from the small Arabic collection. In post-hoc TREC experiments, we examined the effects of different index terms on the Arabic monolingual and English to Arabic cross language retrieval results.

1.2 Methodology

Many new techniques were employed for ad-hoc TREC runs. The ideas were initially tested on a small side collection to verify the effectiveness of the proposed techniques. The collection is called Zad which was provided by Al-Areeb Electronic Publishers, LLC [2]. The collection contains 4,000 documents. The documents were extracted from writings of the thirteenth century scholar Ibn Al-Qayim and cover issues of history, jurisprudence, spirituality, and mannerisms. Also, there are 25 queries with their relevance judgments associated with the collection. The queries are typically 3-6 words long and are available in Arabic and English. The author developed the queries in Arabic, generated the relevance judgments by exhaustively examining the documents in the collection, and translated them to English.

The techniques addressed the choice of Arabic index terms, the use of morphology, and transliteration of words that are not in the dictionary.

To ease work in Arabic, Arabic letters were transliterated to English letters. Also, some letters normalizations were applied and all diacritics were removed. Table 1 shows the mappings between the Arabic letters and their transliterated representations.

[1] Authors are listed in alphabetical order

[2] Visiting from the Media Team, University of Oulu Finland

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

Notice that some letters such as {ى, ي} and {ا, آ, أ, إ, ئ، ؤ، ء} were normalized to "y" and "A" respectively. For the case of {ى, ي}, they are often used interchangeably for each other because of different orthographic conventions or common spelling errors. For the case of {ا, آ, أ, إ, ئ، ؤ، ء}, they represent different forms of the letter hamza.

As for a stop-word list, we used the list that is distributed with Sebawai which includes 130 particles and pronouns [6]. Finally we used the default settings of InQuery with stemming disabled and case sensitivity using the –nostem and –case switches respectively.

For query translation, we used an online machine translation (MT) system developed by Sakhr called Tarjim and a bilingual dictionary [9]. The dictionary was built by extracting unique terms from a 200-megabyte collection of news articles sending them to Tarjim for translation [8]. When our wordlist was sent to Tarjim, Tarjim produced single word translations of the words without regard for context.

1.2.1 Arabic index terms

Several papers were published comparing the use of words, stems, and roots as index terms for purposes of retrieval. All of the studies showed that stems outperformed words and roots outperformed stems [1][3]. We tested the claim using the Zad Arabic document collection. By testing on Zad, we noticed no statistical significance in mean average precision between words, stems, and roots. We thus tried using a combination of words and roots as index terms and the performance was significantly better than using any of them alone. This is a case when using a combination of evidence outperforms using any single evidence alone. For significance testing, we used a paired two-tailed *t*-test. If the p-value of the test was below 0.05, we assumed the difference to be significant.

We investigated other index terms which were character n-grams for both words and roots. We used a combination of character n-grams of different length. For words we used a combination of 3-5 grams and for roots we used 2-4 grams. In combining the n-grams, all the n-gram tuples replace the existing word. For example, the word "Arabic" would be replaced by {Ara, rab, abi, bic}, {Arab, rabi, abic}, and {Arabi, rabic}. Although character n-grams did not outperform words or roots, using the combination of words, roots, and character n-grams of words and roots together was significantly better than any pervious run. Table 2 summarizes the results of using different Arabic index terms on the side collection.

Index term	Mean Avg. Precision
Words	0.3939
Stems	0.4158
Roots	0.4486
Word & Roots	0.4979
Word character n-grams	0.4885
Word and root character n-grams	0.5717

Table 2: Summary of results on side collection of choosing different index terms.

1.2.2 Arabic morphology

Since previous research indicated that using roots as index terms improved mean average precision, two morphology engines capable of generating roots were compared. The first is ALPNET [4][5]. ALPNET has an inventory of 4,500 roots and for any given word, it generates possible roots in random order. The second is Sebawai, which was developed by the first author. Sebawai has an inventory of 10,500 roots and uses a heuristic that guesses which of the roots is most likely.

On the Zad collection, we conducted 4 experiments in which we examined indexing using roots only. The first two experiments involved indexing one root and two roots from ALPNET. For the other two, the experiments involved indexing using the top root and the top two roots from Sebawai. Using Sebawai's guess of the most likely root resulted in a higher mean average precision than when using one root from ALPNET. Further, using the first two roots from ALPNET slightly improved mean average precision, but the improvement was not statistically significant. Using the top two roots of Sebawai significantly harmed retrieval. A likely reason for the fall in mean average precision when the second root was introduced is that the second root amounted to noise. Table 3 summarizes the results of using roots from the two analyzers.

Index term	Mean Avg. Precision
ALPNET – 1 root	0.34
ALPNET – 2 root	0.36
Sebawai – 1 root	0.45
Sebawai – 1 root	0.29

Table 3: summary of results on side collection of using different morphological analyzers

1.2.3 Transliteration and matching of words that are not in the dictionary

For cross-language (CL) runs, we used an MT system in addition to a bilingual English to Arabic dictionary. Each English query was replaced with the full MT suggested translation and the word-by-word translation of query using the bilingual dictionary.

The MT system automatically transliterated words that did not appear in its internal dictionary. However, the suggested MT transliterations were often crude and incorrect. Detecting which words were in the MT lexicon and which ones were transliterations was beyond the scope of the work.

For the word-by-word dictionary based translation, we employed a transliteration technique for words that were not found in the dictionary. We assumed that the words that were not in the dictionary were mostly named entities and required transliteration. The goal of the transliteration technique is to find possible Arabic words that correspond to the given English word. The process involved transliteration, matching, and clustering.

Transliteration: All the English letters are mapped to the closest Arabic sounding letters. For example, the letter “r” is mapped to “ر”. Letter combinations such as “ch” and “sh” are recognized and mapped to Arabic. Some letters such “j” and “g” are normalized to one letter. Table 4 lists the English to Arabic Transliteration mappings.

a*	A	b	b	c[iey]	s
c	k	d	d	[aeiou]	#**
f	f	g	g	h	h
j	g	k	k	l	l
m	m	n	n	p	b
q	q	r	r	s	s
t	t	v	f	x	k
y	y	z	z	th	O
al-*	#	ala*	A	[sc]h	P

Table 4: English to Arabic transliteration mappings (* initial letter(s) in the word, ** # represents nothing)

Matching: For the matching the transliterated words to the words in the collection to be searched, the prefixes {w,wAl,Al,wb,[wlbfk]} were removed from all the words; all the vowels are dropped; and some Arabic letters were normalized. Table 5 lists all the normalizations of Arabic letters.

[Ss]	s	[Zz]	z	[xk]	k	[AE]	A
[Hh]	h	[Tt]	t	[gj]	g	p	#

Table 5: Arabic letter normalizations (* # represents nothing)

Clustering: after the possible Arabic transliterations are found, all are used together in the Arabic queries using InQuery's #syn operator which sets all of them as synonyms to each other.

The effect of this technique is not completely clear given that most of the words in the queries for the side collection and TREC were in the bilingual dictionary.

1.3 Experiment Design

1.3.1 Arabic Monolingual Run

Automatic Arabic Run: Based on our experiments on Zad collection, we used words, stems, roots, character n-grams for words, and character n-grams for roots to index the TREC collection. To obtain stems and roots, we used the top suggestions of Sebawai. For n-grams, we used 2-4 character n-grams for roots and 3-5 character n-grams for words.

Manual Arabic Run: For the manual runs, the title, description, and narratives were used along with words that were manual introduced by the authors. We removed stop structures from queries such as " المقالات المتعلقة" (the articles relating to) and examples of what is not relevant. The final queries were run in exactly the same way as the automatic Arabic setup.

Post-hoc experiments: After the relevance judgments were available we explored the use of different index terms on our retrieval effectiveness. We examined indexing using words only, stems only, roots only, word character trigrams, root character bigrams, and words, stems, and roots together. The queries were automatically formulated using the full text of the title and descriptions of the queries.

1.3.2 English-Arabic CLIR Runs

For the CLIR runs, we tested three different configurations as follows:

Basic Configuration: All the queries were translated using Tarjim only. The output of the MT system was fed to the automatic Arabic IR configuration described above. It is noteworthy that Tarjim transliterates the words that do not appear its dictionary.

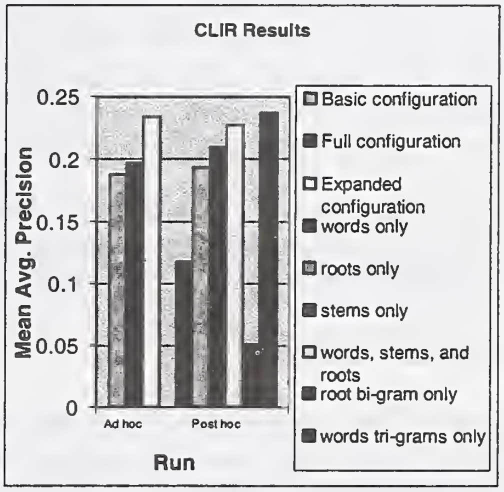
Full Configuration: In this configuration, the English queries were translated using Tarjim and the bilingual dictionary. If a word is not found in the dictionary, the word is transliterated, matched to Arabic words in the TREC collection, and the matches were clustered in the manner described above. The outputs of the MT system, the dictionary based translation, and the transliteration are combined and fed to the automatic Arabic IR configuration.

Expansion Configuration: The expansion configuration is identical to the full configuration but with expansion using blind relevance feedback on the English and the Arabic sides. For expansion on the English side, we used Associated Press articles from 1994-1998. They were part of the North American News Text Corpus (Supplement) and AP World Stream English Collection from the Linguistic Data Consortium [7]. The expansion collection was searched using the English queries without modification and the top 10 returned documents for every query were used to expand the query. For the expansion on the Arabic side, the TREC collection was used for expansion. The AFP collection was searched using the Arabic queries, which include the roots and n-grams, and the top 10 retrieved documents for ever query were used to expand the queries.

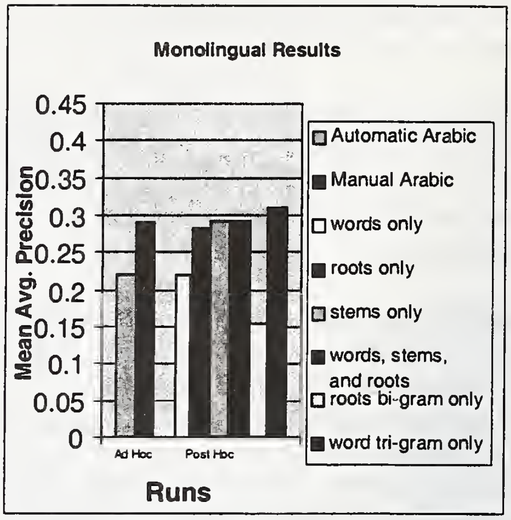
Post-hoc experiments: We examined indexing using words only, stems only, roots only, word character trigrams, root character bigrams, and words, stems, and roots together. The titles and descriptions of the queries were automatically translated using Tarjim alone.

1.4 Results

Official cross language runs (Ad hoc)	
Run	**Mean Avg. Precision**
bc - Basic configuration	0.19
fc - Full configuration	0.20
xp - Expanded configuration	0.23
Post hoc runs (all basic configuration)	
w - words only	0.12
r - roots only	0.20
s - stems only	0.21
wsr - words, stems, and roots	0.23
rg2 – root bigram	0.05
wg3 – word trigram	0.24



Official monolingual runs (Ad hoc)	
Run	Mean Avg. Precision
aa - Automatic Arabic	0.22
ma - Manual Arabic	0.29
Post hoc runs (all automatic)	
w - words only	0.22
r - roots only	0.28
s - stems only	0.29
wsr - words, stems, and roots	0.28
rg2 – root bigram	0.15
wg3 – word trigram	0.31



1.5 Discussion

The results point to a few important conclusions:

1. The translation technique used was effective. In fact, for the official results the mean average precision of the non-expanded CLIR run was 89% relative to the mean average precision of the automatic Arabic run. Also, none of the CLIR runs were significantly better or worse than any of the Arabic run.

2. For the official runs, the results of individual queries were better than the median in 10 queries and 18 queries for the automatic non-expanded monolingual and cross language runs respectively.

3. Perhaps the use of n-grams for roots may have hurt the monolingual result.. When we tried using roots only as the index terms in later monolingual experiments, the resulting mean average precision was significantly better than any of our official results. However, the CLIR results were slightly hurt, but not significantly by the use of n-grams. Also using bigrams for roots seems to be a bad idea especially for CLIR runs.

4. The use of word character trigrams and stems produced the best results among the post-hoc experiments. Perhaps other experiments examining the effect of indexing using other n-grams, terms produced by morphological analysis, or combinations of both are warranted.

1.6 CLIR References

[1] Abu-Salem, Hani, Mahmoud Al-Omari, and Martha Evens, "Stemming Methodologies Over Individual Query Words for Arabic Information Retrieval." JASIS. 50 (6): 524-529, 1999.

[2] Al-Areeb Electronic Publishers, LLC. 16013 Malcolm Dr., Laurel, MD 20707, USA

[3] Al-Kharashi, Ibrahim and Martha Evens, "Comparing Words, Stems, and Roots as Index Terms in an Arabic Information Retrieval." JASIS. 45 (8): 548-560, 1994.

[4] Beesley, Kenneth, "Arabic Finite-State Morphological Analysis and Generation." COLING-96, 1996.

[5] Beesley, Kenneth, Tim Buckwalter, and Stuart Newton, "Two-Level Finite-State Analysis of Arabic Morphology." Proceedings of the Seminar on Bilingual Computing in Arabic and English, Cambridge, England, 1989.

[6] Darwish, Kareem, "Building a Shallow Morphological Analyzer in One Day", www.glue.umd.edu/~kareem/hamlet/arabic/sebawai.tar.gz

[7] MacIntyre, Robert, "North American News Text Supplement", LDC98T30, LDC, 1998.

[8] NIST, Text Research Collection Volume 5, April 1997.

[9] tarjim.ajeeb.com, Sakhr Technologies, Cairo, Egypt www.sakhr.com

2 Video Track

2.1 Introduction

Our primary focus in this track was to get exposure to the process, test existing algorithms and determine the types of queries our current approaches was suited for. As previously stated, we participated in both the shot boundary detection and known item search.

2.2 Shot Boundary Detection

2.2.1 Overview

There has been a tremendous amount of work done on problem of "shot" detection in video. Our system was originally developed and extended in 1995 to process large quantities of MPEG-compressed video and provide a visual summary. In order to provide such a summary, we originally defined a shot change not only as a cut or gradual change, but also as the point where a significant amount of new content was introduced in the scene, either by new subjects appearing, or the camera panning to a new view of the current environment. The system runs at about 3x real-time and relies on a consistent and predictable coding of the video.

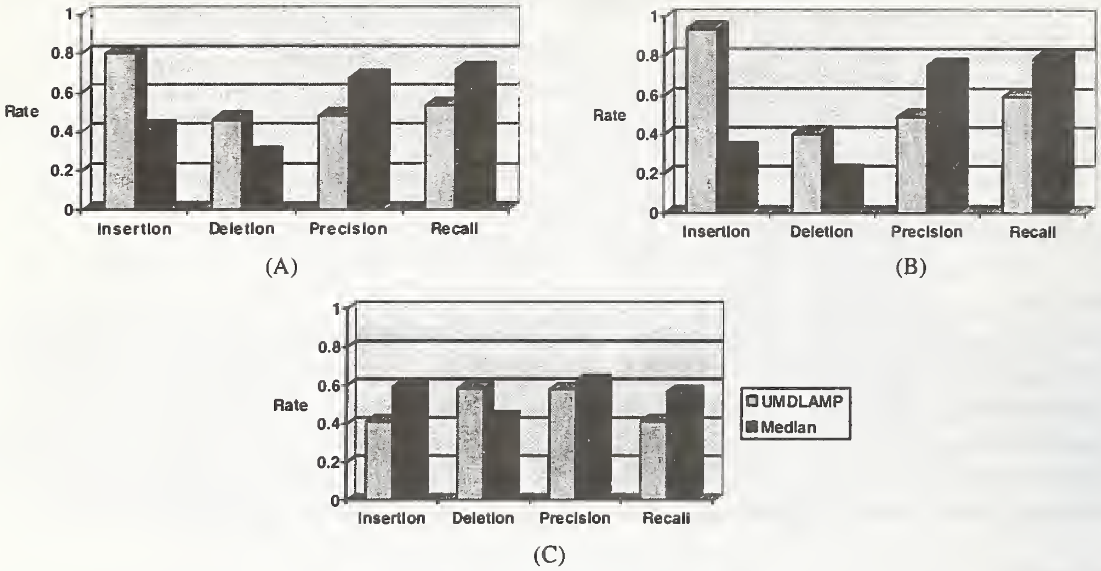
2.2.2 Approach

MERIT [21] detects cut shot changes by examining the MPEG macroblocks and DCT coefficients. If macroblocks of a frame rely very little on the proceeding or succeeding frames for encoding, the likelihood is high that there is shot change since same shot frames use the same information. Shot changes are determined by calculating the faction of macroblocks using information from other frames' macroblock to the total number of macroblocks. If this faction is below a threshold then is the potential for a shot change. All self-encoded frames are considered a potential for further processing the validation phase if it comes directly after a previous potential frame. In the validation phase DCT values of potential shot changes frames are decoded. If there is sufficient change in the DCT values, then the shot change is kept in the results. A shot change is validated by determining the difference between the DCT values of the frames before and after the potential frame. If the difference is above a threshold then it is considered a shot change. The thresholds for the system were determined by separately testing 12 minutes of video data of various genres (animations, commercials, movies, news, sports, and surveillance) that minimized the number of false and undetected transitions. No training was done with TREC collection. Details of the algorithm can be found in [21]. The MERIT system is available upon request to research organizations.

Gradual scene change detections are detected by projecting the DCT coefficient feature vector into a low dimensional space using a linear time reduction algorithm know as FastMap. The layout of these low dimensional points are tracked and if they do not cluster, a gradual change is detected. Details can be found in [22].

2.2.3 Experiments

Overall the system performed worse than the weighted median in all performance categories (Figure 1a). Incorrect cut transition had a severe impact on the overall results (Figure 1b). In gradual shot scene detection, the system performed better but missed more than the median performance system (Figure 1c). The system achieves its best results with database videos (ahf1, eal1, pfml) with a bit-rate of 1.4MB/sec. Although some videos (ann. i005, anni009) had higher bit-rates, the grainy quality of the video degraded the accuracy of the macroblocks and DCT coefficients. Database clips with lower bit rates had lower performance rates.



(A)

(B)

(C)

Figure 1: Overall Performance

2.2.4 Discussion

An examination of the undetected transitions indicates that reliance of DCT values for validation of shot changes makes it difficult to detect certain shot change situations. Cut transitions in which the two scenes were very similar in color or very dark were problematic. For example, in BOR08.mpg, there where many shot changes involving transition between old photographs that were prominently gold in color. Since the difference in the DCT would be minimal the difference did not produce a value above the threshold to indicate a shot change. This problem also occurred with gradual transitions that involved fades to black, fades from black, similar color or were dark in nature. The difference between frames did not produce a value greater than the threshold since the DCT values of the dark areas dominated varied very little from each other.

Our system often found transitions in clips where there were none.. The situations in which this error occurred were typically either camera jitter, when the camera made a sudden movement in a new direction, when a background object moved into a prominent position in the foreground, or when the camera zoomed in on a object. The macroblocks indicate a large change and was confirmed in the validation process since was a substantial change in color.

Although working with in the MPEG compressed domain is a quick way to analyze video it can produce errors. The reliance on DCT values makes it difficult to detect transitions that involve scenes that are dark or have similar prominent color scheme. In these cases the probability is high that the changes will not register above a threshold. In the future an adaptive threshold is needed to detect the presence of these situations during the validation phase.

2.3 Known Item and General Search

2.3.1 Overview

Content-based retrieval has been subject to active research since the early 1990's [3] and a large number of experimental image retrieval systems have been introduced, such as BlobWorld [4], Chabot [5], Mars [6], NeTra [7], Photobook [8], QBIC [9], Surfimage [10] and VisualSEEK [11]. These systems retrieve images based on cues such as color, texture, and shape, of which color remains as the most powerful and most useful feature for general purpose retrieval. Color-based retrieval first evolved from simple statistical measures such as average color to color histograms [9,5,8], but histograms alone suffer for large collections since different configurations can produce the same histogram.

2.3.2 Approach

The spatial correlation of colors as a function of spatial distance is an image feature introduced by Huang *et al.* [12] known as a correlogram. Our approach extends this method and uses a novel color content method, the Temporal Color Correlogram (TCC), to capture the spatio-temporal relationship of colors in a video shot using co-occurrence statistics. TCC is an extension of HSV Color Correlogram (CC), which is found very effective in content-based image retrieval [1]. Temporal Color Correlogram computes autocorrelation of the quantized HSV color values from a set of frame samples taken from a video shot. In this paper, the efficiencies of TCC and HSV Color Correlogram (CC) are evaluated against other retrieval systems participating VideoTREC track evaluation. Tests are executed using our retrieval system, CMRS, which is specifically developed for multimedia information retrieval purposes.

2.3.2.1 Correlogram extension in temporal domain

In digital video, color and intensity information change temporally over a shot, creating the illusion of object or observer movement. This knowledge is also used in modern video compression algorithms, where motion is estimated by moving rectangular blocks of quantized illumination and colors towards the expected direction of motion (MPEG) [19]. In order to create a content-based descriptor for a video shot, such structural information should be trasferred into computable features.

The temporal changes of video shot contents can be described using the temporal correlogram. The benefits over more traditional approaches, such as color histograms, derive from its ability to encapsulate the temporal changes in small spatial color environments. Figure 2 depicts a temporal color change in a small spatial environment. Whereas the color histogram would only portray the proportional amount of color in these frames, temporal correlogram will capture information about the spatial changes of these colors occurring over time.

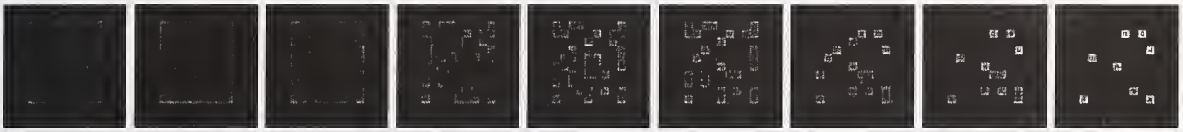


Figure 2: Temporal color change illustrated by frame sequence. Temporal correlogram captures the dispersion of the color element whereas histogram does not.

Let *N* be the amount of sample frames I^n taken from a shot *S*. Values of *n* vary from *1* to *N* indicating the index in the sample frame sequence. The temporal correlogram is calculated as

$$\bar{\gamma}^{(d)}_{c_i,c_j}(S) \equiv \Pr_{p_1 \in I^n_{c_i}, p_2 \in I^n} \left[p_2 \in I^n_{c_j} \middle| |p_1 - p_2| = d \right] \qquad (3)$$

which gives the probability that given any pixel p_1 of color c_i, a pixel p_2 at a distance *d* from the given pixel p_1 is of color c_j among the shot's sample frames I^n.

For computational benefits [1], the Temporal Color Correlogram (noted here as TCC) used for this study is computed as an autocorrelogram, which is obtained from Eq. 3. by replacing c_j with c_i. The quantization of HSV color space for TCC follows the quantization of CC.

2.3.3 Experiments

To evaluate the temporal correlogram efficiency, we used 11 hours database of MPEG1 videos available for VideoTREC track participants [2]. First, the video material was segmented to create shots using VideoLogger video editing software from Virage [20] and our own system (above) but the Virage results were used. For the 11 hours of video, 7375 shot segments were created with the average shot length of approximately 5 seconds. From the shot frames, the beginning frame was selected as a representative key frame, from which the static image feature, CC, was obtained. In order to calculate TCC non-exhaustively and to keep the number of samples in equal for varying shot lengths, each shot was sampled evenly with a respective sampling delay so that the number of sample frames did not exceed 40. After segmentation, shot features were fed into our CMRS retrieval system and queries were defined using either example videos or example images depending on the respective VideoTREC topic specification [2].

VideoTREC result submission contained retrieval results of two system configurations. First configuration was obtained using TCC for the retrieval topics that contained video examples in the topic definition. Second configuration used CC for topics that contained example images in their definition. Table 6 shows the average precisions of General Search results for the TCC feature in different topic categories. As the results show, TCC as a purely automatic method did worse in Interactive and Automatic+Interactive topics, since no other cues than this color structure feature were used in a query (meaning there was no human involvement to prune the results). The General Search overall results were not impressive in contrast to other participating systems as can be seen from the Figure 3 that depicts all system precisions ranked into evolving curve starting from worst system on the left. However, the average precision in Automatic topics ranks TCC higher, just below the median of all systems.

Topic Type (# of topics)	Average Precision
Interactive (3)	0.08
Automatic+Interactive (8)	0.13
Automatic (17)	0.24
Overall Average (28)	0.19

Table 6:.General Search Results for TCC with different topic categories.

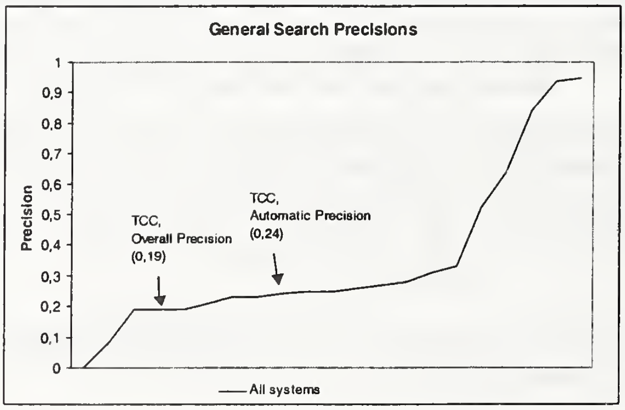


Figure 3: TCC General Search performance against other systems. The curve indicates ranked list of system precisions, worst being in the left and best in the right end of the curve.

Table 7 shows the Known Item Search results with different match parameters. The parameters define when a retrieved item is a successful match to a known item. The loosest criteria (0.333/0.333) for the match expects the retrieved shot to be overlapping with known item at least one third of the shot duration having the same rule for the known item sequence. The tightest criteria require two thirds of the shot durations to overlap. It can be seen that the results for the CC configuration are dismal whereas TCC is doing better. In Figure 4 one can see that the TCC recall is ranked in the median (11th out of 21) of all systems for Known Item Searches.

RECALL

Parameters	0.333/0.333	0.333/0.666	0.666/0.333	0.666/0.666
TCC	0,181	0,117	0,07	0,02
CC	0,014	0,002	0,014	0,001

PRECISION

Parameters	0.333/0.333	0.333/0.666	0.666/0.333	0.666/0.666
TCC	0,011	0,005	0,005	0,001

Table 7: Recall and Precision averages for TCC and CC configurations with different match parameters.

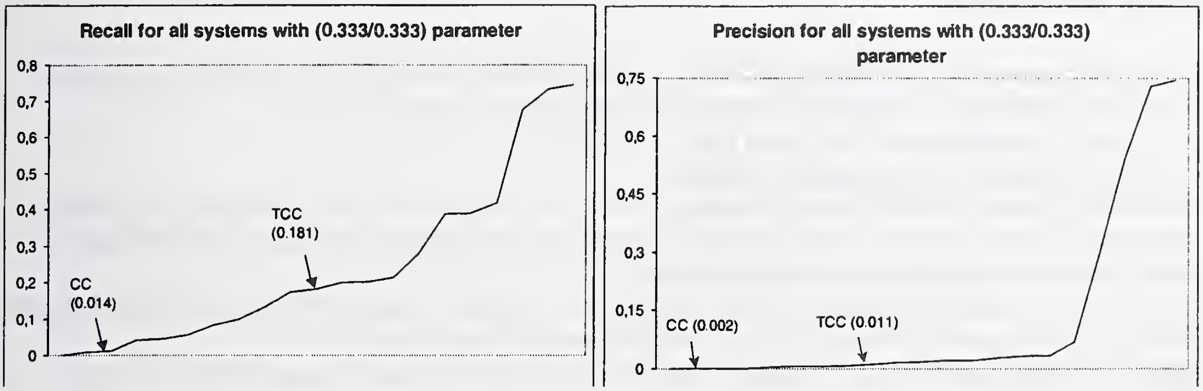


Figure 4: TCC and CC recall and precision against respective values of other systems. The curve indicates ranked list of system precisions/recalls, worst being in the left and best in the right end of the curve.

Table 8 shows the top 5 topics for the Known Item Searches. Topic 3 was the most successful. It considered finding video segments that depict a lunar vehicle traveling on the moon. Other topics in the list considered problems of finding a yellow boat, snow capped mountains or a student from classroom footage.

	Precision	Recall
1st	Topic 3	Topic 3
2nd	Topic 35	Topic 6
3rd	Topic 36	Topic 35
4th	Topic 4	Topic 36
5th	Topic 6	Topic 4

Table 8: Top 5 topic results by precision and recall

2.3.4 Discussion

The semantic gap is too large for video analysis features like TCC and CC in search problems such as in the General Search topics and the topics containing an example image. In other words, the structural, 'mechanical', content of example images and video shots doesn't convey the meaning of the semantic request that the person defining a query actually pursues. This can be improved by combining other cues such as audio and text to focus the search towards more meaningful locations in a video.

Better results were obtained in the topics that seek Known Items with similar structural shot properties. Topics that tried to locate footage from the same target with different camera angles or object positions gave the most successful results. The evaluation criteria for a search hit was rather strict. It leaves many questions whether people searching video databases want the exact locations of the known items returned, or rather, just a pointer inside a video where one can start to examine the video by himself. Is it more beneficial to provide the user with accurate segments together with very low retrieval ranks, rather than giving less accurate results with higher ranks? Surely users will rather watch a couple of longer segments from the top ranks than to wade through tens of useless clips in order to find the exact match with low rank. What makes the problem worse is that no automatic system will be accurate enough to successfully encapsulate the varieties in semantic definitions that people will use in their queries into heterogeneous video databases. In the retrieval results there will always exist loads of useless segments among the really significant ones.

2.4 Video References

[1] Ojala T, Rautiainen M, Matinmikko E & Aittola M (2001) Semantic image retrieval with HSV correlograms. Proc. 12th Scandinavian Conference on Image Analysis, Bergen, Norway, 621-627.

[2] TREC-2001 Video Retrieval Track Home Page, (10/25/2001) http://www-nlpir.nist.gov/projects/t01v/t01v.html

[3] Eakins J & Graham M (1999) Content-Based Image Retrieval: A report to the JISC Technology Applications Programme. Institute for Image Data Research, University of Northumbria at Newcastle, United Kingdom. *http://www.unn.ac.uk/iidr/research/cbir/report.html*.

[4] Carson C, Belongie S, Greenspan H & Malik J (1997) Region-based image querying. Proc. IEEE Workshop on Content-Based Access of Image and Video Libraries, San Juan, Puerto Rico, 42-49.

[5] Ogle V & Stonebraker M (1995) Chabot: retrieval from a relational database of images. IEEE Computer Magazine 28:40-48.

[6] Ortega M, Rui Y, Chakrabarti K, Porkaew K, Mehrotra S, Huang TS (1998) Supporting ranked Boolean similarity queries in MARS. IEEE Transactions on Knowledge and Data Engineering 10:905-925.

[7] Ma WY & Manjunath BS (1997) NeTra: a toolbox for navigating large image databases. Proc. International Conference on Image Processing, Santa Barbara, CA, 1:568-571.

[8] Pentland A, Picard R & Sclaroff S (1996) Photobook: content-based manipulation of image databases. International Journal of Computer Vision 18:233-254.

[9] Flickner M, Sawhney H, Niblack W, Ashley J, Huang Q, Dom B, Gorkani M, Hafner J, Lee D, Petkovic D, Steele D & Yanker P (1995) Query by image and video content: The QBIC system. IEEE Computer Magazine 28:23-32.

[10] Mitschke M, Meilhac C & Boujemaa N (1998) Surfimage: a flexible content-based image retrieval system. Proc. Sixth ACM International Conference on Multimedia, Bristol, UK, 339-344.

[11] Smith J & Chang S-F (1996) VisualSEEK: a fully automated content-based image query system. Proc. Fourth ACM International Conference on Multimedia, Boston, MA, 87-98.

[12] Huang J, Kumar SR, Mitra M & Zhu WJ (1997) Image indexing using color correlograms. Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Juan, Puerto Rico, 762-768.

[13] Pass G, Zabih R & Miller J (1996) Comparing images using color coherence vectors. Proc. Fourth ACM International Conference on Multimedia, Boston, MA, 65-73.

[14] Huang J, Kumar SR, Mitra M & Zhu WJ (1998) Spatial color indexing and applications. Proc. Sixth International conference on Computer Vision, Bombay, India, 602-607.

[15] Huang J, Kumar SR & Mitra M (1997) Combining supervised learning with color correlograms for content-based image retrieval. Proc. Fifth ACM International Conference on Multimedia, Seattle, WA, 325-334.

[16] Ma WY & Zhang HJ (1998) Benchmarking of image features for content-based retrieval. Proc. 32nd Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, 1:253-257.

[17] Hafner J, Sawhney HS, Equitz W, Flickner M, Niblack W (1995) Efficient color histogram indexing for quadratic form distance functions. IEEE Transactions on Pattern Analysis and Machine Intelligence 17:729-736.

[18] Tekalp AM (1995) Digital video processing. Prentice Hall signal processing series, US, 526.

[19] MPEG.ORG (10/25/2001) http://www.mpeg.org/MPEG/video.html

[20] Virage, Inc. (10/25/2001) http://www.virage.com/

[21] V. Kobla, D.S. Doermann, and A. Rosenfeld, Compressed domain video segmentation, CfAR Technical Report CS-TR-3688, University of Maryland, College Park, 1996.

[22] V. Kobla, D. DeMenthon, and D. Doermann, Special effect edit detection using VideoTrails: a comparison with existing techniques, Proceedings of SPIE conference on Storage and Retrieval for Image and Video Databases VII, Jan, 1999.

Arabic Information Retrieval at UMass in TREC-10

Leah S. Larkey and Margaret E. Connell

Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA USA 01002

{larkey|connell}@cs.umass.edu

1. Introduction

The University of Massachusetts took on the TREC10 cross-language track with no prior experience with Arabic, and no Arabic speakers among any of our researchers or students. We intended to implement some standard approaches, and to extend a language modeling approach to handle co-occurrences. Given the lack of resources – training data, electronic bilingual dictionaries, and stemmers, and our unfamiliarity with Arabic, we had our hands full carrying out some standard approaches to monolingual and cross-language Arabic retrieval, and did not submit any runs based on novel approaches.

We submitted three monolingual runs and one cross-language run. We first describe the models, techniques, and resources we used, then we describe each run in detail. Our official runs performed moderately well, in the second tier (3^{rd} or 4^{th} place). Since submitting these results, we have improved normalization and stemming, improved dictionary construction, expanded Arabic queries, improved estimation and smoothing in language models, and added combination of evidence, increasing performance by a substantial amount.

2. Information Retrieval Engines

We used INQUERY [2] for two of our three monolingual runs and our cross-language run, and language modeling (LM) for one monolingual run. The processing was carried out using in-house software which implemented both engines, to insure that the stop lists, tokenization, and other details were identical. The same tokenization was used in indexing the Arabic corpus and processing Arabic queries. In fact, except for one minor difference in tokenization, Arabic strings were treated exactly like English strings – as a simple string of bytes, regardless of how they would be rendered on the screen. For both English and Arabic, text was broken up into words at any white space or punctuation characters. The minor difference in Arabic tokenization consisted of five additional Arabic punctuation characters included in the definition of punctuation. Words of one-byte length (in CP1256 encoding) were not indexed.

2.1. Inquery

Two of the three monolingual runs and the cross-language run used a version of INQUERY as the search engine. This version computes the belief function reported in UMass's TREC9 report [1]. 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.

2.2. Language Modelling (LM)

2.2.1. Monolingual

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 [7] and [9]. Unigram probabilities in our official run were estimated as a mixture of maximum likelihood probability estimates from the document and the corpus, as follows:

$$P(Q \mid Doc) = \prod_{q \in Q} (\lambda P(q \mid Doc) + (1 - \lambda) P(q \mid BG))$$

where *P(Q/Doc)* is the probability of generating the query from the document model, *q* are the words in the query, λ is a mixture parameter, *P(q/BG)* is the probability of the query word in the background model, and *P(q/DOC)* is the probability of the query word in the document. Normally, the maximum likelihood probabilities are estimated as:

$$P(q \mid Doc) = \frac{tf_{q,Doc}}{|Doc|}$$

where $tf_{q,Doc}$ is the number of occurrences of term q in document, and $|Doc|$ is the length of document, that is, the number of total term occurrences in the document. In an analogous manner, the background probabilities are estimated from a collection *C* which may or may not be the collection in which the document resides, as:

$$P(q \mid BG) = \frac{tf_{q,C}}{|C|}$$

where $tf_{q,C}$ is the number of occurrences of term q in the collection *C*, and $|C|$ is the number of total occurrences of all terms in *C*.

For our official run, we estimated background probabilities as above, and we estimated λ via the Witten Bell method [10], in which

$$\lambda = \frac{|Doc|}{|Doc| + N_{Doc}}$$ where N_{Doc} is the number of different terms in the document.

Posthoc work on Arabic and other data has shown improvements in monolingual LM retrieval by modifying how λ, the mixture parameter, is calculated. For long (expanded) queries, we set λ to a constant value of .4. For short (unexpanded) queries we use Dirichlet smoothing [11], that is,

$$\lambda = \frac{|Doc|}{|Doc| + k}$$ where $k = 800$.

We have also found better LM performance on Arabic and other data if we use document frequencies rather than term frequencies for background models, as in [3], that is:

$$P(q \mid BG) = \frac{df_{q,C}}{\sum_{t \in C} df_{t,C}}$$ where $df_{q,C}$ is the number of documents in *C* containing term *q*, and the summation is over all the terms in the collection.

3. The Arabic Corpus

The AFP ARB corpus of 383,872 documents in Arabic was converted to CP1256 encoding and normalized in the manner described above. The corpus was indexed in two different ways. For the non-stemmed conditions (UMass4), the corpus was normalized, tokenized using the Arabic tokenizer, and every token longer than one byte in length was indexed. For the stemmed conditions (UMass1, UMass2, and UMass3), stemmed tokens longer than one byte in length were indexed.

4. Arabic Resources and Techniques

4.1. Normalization of Arabic

In order to handle the variations in the way text can be represented in Arabic, we performed several kinds of normalization on text in the corpus and in the queries. The normalized form of the corpus was used for indexing (in the non-stemmed conditions), and queries were normalized before they were submitted to the search engine. Dictionaries were also normalized, so that their output would match the forms found in the queries and the corpus.

In our official runs, normalization consisted of the following steps:

- Convert to Windows Arabic encoding (CP1256), if necessary
- Remove punctuation
- Remove diacritics (mainly weak vowels) Most of the corpus did not contain weak vowels. Some of the dictionary entries contained weak vowels. This made everything consistent.
- Remove non letters
- Replace initial إ or أ with bare alif ا.
- Replace آ with ا
- Replace the sequence ىء with ئ
- Replace final ى with ي
- Replace final ة with ه

The definitions of punctuation, diacritics, and non-letters came from the Khoja stemmer, below.

We later improved normalization substantially via two minor changes – replacing إ or أ with bare alif ا regardless of position in the word, and removing tatweel. The label *norm* refers to the original normalization. *Norm2* refers to the modified normalization, and includes stop word removal.

4.2. Stemming

We obtained a stemmer from Shereen Khoja of the Computing Department at Lancaster University [4], which we modified to suit our needs. The stemmer included several useful data files such as a list of all diacritic characters, punctuation characters, definite articles, and 168 stop words, etc. We used some of these files in our normalization algorithm above. This stemmer attempts to find roots for Arabic words, which are far more abstract than stems. It first removes definite articles, prefixes, and suffixes, then attempts to find the root for the stripped form. If no root is found, then the word is left intact. The stemmer also removes stop words. We know both that roots are too abstract for effective information retrieval, and that the overall approach of not stripping any affixes at all is faulty. Although this stemmer made many mistakes, it improved performance immensely, nevertheless.

The changes we made to the Khoja stemmer were (1) If a root were not found, the normalized form was returned, rather than returning the original unmodified word. (2) We added the list of place names described in section 4.3.4 as "unbreakable" words exempt from stemming.

In addition to the Arabic stop word list included in the Khoja stemmer, we applied a script to remove stop phrases, which were translations of the stop phrases we had in our English stop-phrase removal script.

After TREC we developed a light stemmer which strips definite articles (ال، وال، بال، كال، فال) and و (*and*) from the beginnings of normalized words and strips 10 suffixes from the ends of words (ها، ان، ات، ون، ين، يه، ية، ه، ة، ي) [6]. With stop word removal this stemmer yielded higher performance than the khoja stemmer. In the Results sections below, *khoja* refers to the original Khoja stemmer, *khoja-u* refers to the version where words on city and country list are considered unbreakable and exempt from stemming. *Light* refers to the light stemmer.

4.3. Dictionaries

Our structural approach to query translation for cross-language retrieval required that we look up each individual English word in each query (including words added by query expansion), and get all available translations into Arabic words or phrases. We put together several different sources of translations for English words into Arabic, using free resources from the web as much as possible.

4.3.1. The Ectaco dictionary

The Ectaco dictionary is available online, at `http://www.ectaco.com/online`. We could not query this dictionary under program control, so we collected entries manually from the web site. For each English query term and expanded query term, we collected entries by cutting and pasting all the Arabic translations that were available. If an English word were not found, we searched for the word as stemmed by the *kstem* stemmer.

4.3.2. The Sakhr multilingual dictionary

The Sakhr multilingual dictionary is found at `http://dictionary.ajeeb.com/en.htm`. We were able to harvest entries from this dictionary under the control of a Java program which repeatedly queried the English to Arabic page with English words. We collected all available definitions for query words and expansion words. In addition, we collected Arabic-English entries for all available Arabic words in the AFP_ARB corpus.

4.3.3. Sakhr SET machine translation

The Sakhr SET machine translation engine was made available to TREC participants by Mark Meinke at `http://217.52.128.36/set/English/`. This was not used to translate queries. We used it only to look up individual words that we did not find in either of the two dictionaries. This MT engine has a transliteration component, which converts the English word into Arabic characters if a translation is not found. We used this as a substitute for a transliteration algorithm, which we did not yet have available.

4.3.4. Place name lexicon

A small bilingual lexicon of country and city names was derived from a list of world cities we found on the web at `http://www.fourmilab.ch/earthview/cities.html`. This list had 489 entries, and listed the names of most countries of the world, their capitals, and a few other major cities. To get the Arabic translations, we used the Sakhr SET engine, which performed machine translation from English to Arabic. Many of these translations were transliterations. This list of place names (and only this list, which was made independently of the queries) was hand corrected by an Arabic speaking consultant.

4.3.5. Small and large lexicons

Two bilingual lexicons were built. The first (*small*) consisted of the place names plus all the English-Arabic translations found for all of the English query words, including the additional query words added via expansion of the English query for the cross-language run. The second (*large*) lexicon consisted of all the entries from the small lexicon, plus the all the inverted Arabic-English entries. For convenience, we built stemmed versions of the lexicons for each stemmer that we tested. The small normalized English to Arabic lexicon contained 28,868 English words, 269,526 different Arabic translations, for an average of 9.3 different translations per word. The large normalized lexicon contained 50,358 English words, 1,692,408 translations, for an average of 33.6 different translations per word.

5. Other Resources and Techniques

5.1. Stop words and phrases

English stop words (used only for cross-language retrieval) 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). We built a list of Arabic stop phrases from this by translating the phrases. Arabic stop words are from the Khoja stemmer's list of 168 stop words.

5.2. Query Expansion

We expanded English queries in our official cross-language run, 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 20 documents for each query, and ranked the terms from these documents using the ratio method described in Ponte's thesis, chapter 5 [8]. The five top ranked new English terms were then added to the query. Each term in the query received a final weight of $2w_o + w_e$ where w_o is the original weight in the unexpanded query, and w_e is the score the term received by the ratio method expansion.

After submitting the official runs, we changed the expansion method. Terms from the top 10 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 weights were set to $2w_o + w_e$ where w_o is the original weight in the unexpanded query and w_e=1.

Due to technical problems involving the interaction of Arabic stemming with query expansion, and lack of time we did not submit any official runs in which the Arabic queries (monolingual, or translated for cross-language) had been expanded.

After TREC, we added Arabic query expansion, performed as follows: retrieve the top 10 documents for the Arabic query, using LM retrieval if the expanded query would be run in an LM condition, and using Inquery retrieval if the expanded query would run in an Inquery condition. Terms from the top ten documents were ranked using the same expansion score used in the post-hoc English expansion. The top 50 terms that were not already part of the original query were added. For Inquery conditions, the added terms were added to the original query as additional terms under the top level #wsum operator. For both Inquery and LM conditions, the weights on original terms were doubled, and the new terms received a weight of 1.

6. Monolingual Runs

6.1. Description of Runs

We entered three monolingual runs, which differed in stemming and in retrieval algorithms:

1. **Inquery Baseline.** (UMass4) : normalized but not stemmed
2. **Inquery Stemmed** (UMass1): stemmed using Khoja-u stemmer
3. **LM Stemmed** (UMass2): stemmed using Khoja-u stemmer. LM as described in section 2.2.1.

The following steps were carried out in processing all monolingual runs.

1. Convert queries to CP1256 encoding.
2. Remove all but the title and description fields.
3. Remove stop phrases from Arabic queries.

4. Normalize or stem the query, depending on the condition.
5. Rank the documents using either INQUERY or LM, depending on condition.

6.2. Results

Without stemming our system performed very poorly. With stemming it performed quite well, as summarized in Table 1. The table shows average precision for each run, and summarizes a query-by-query comparison with the median performance over 20 monolingual manual and automatic runs, with respect to average precision and the number of relevant documents returned in the top 1000.

As the table shows, stemming improves the results immensely. With stemming, average precision improved 49% over the INQUERY baseline. The LM stemmed condition was not as good as the Inquery stemmed condition. A striking pattern apparent in the table is a recall bias due to stemming. In both stemmed conditions the number of queries above the median in relevant documents returned in the top 1000 is larger than the number of queries above the median in average precision.

Table 1: Monolingual Results - official runs with normalization and khoja-u stemmer

CONDITION	Name of Run	Average Precision	Number of Queries at or Above Median	
			Average Precision	Rel Ret in top 1000
Inquery baseline	UMass4	.2104	10/25	10/25
Inquery stemmed	UMass1	.3129	18/25	24/25
LM baseline	not submitted	.1858		
LM stemmed	UMass2	.2597	16/25	20/25

6.3. Posthoc Monolingual Experiments

We compare the official results with runs using improved normalization, stemming, and with query expansion, and better language modeling. Table 1 shows the old and new conditions including the official runs, which are asterisked. *Raw* means that no stemming or stop word removal was applied. *Norm, norm2, khoja-u, khoja,* and *light* are defined in section 4.2 above. Since roots and lightly stemmed words are quite different representations of Arabic words, we reasoned that they could be productively combined. *Light+khoja* is a combination of evidence run, where the ranked lists from the light and khoja runs were averaged without any normalization of scores. Shaded cells were conditions that were not run.

Table 2: Monolingual results with improved normalization, stemming, and language modeling, with and without query expansion

	Raw	Norm	Norm2	Khoja-u	Khoja	Light	Light+ Khoja
Inquery	.1935	**.2104***	.2408	**.3129***	.3410	.3894	.4088
Inquery + Query Expansion	.2709	.3002	.3303	.3595	.3778	.4274	.4408
LM		.1858		**.2597***			
LMnew	.1879	.2020	.2431	.3189	.3479	.3736	.3981
LMnew+Query Expansion	.2629	.2990	.3335	.3490	.3772	.4130	.4465

* official runs

It is apparent from these runs that the light stemmer is superior to the khoja stemmer. Although it seemed like a good idea to have the list of unbreakable place names as part of the Khoja stemmer, performance was better without it. These results also show that the changes in background model estimation and smoothing bring language model performance to a level comparable to that of Inquery.

7. Cross Language Retrieval

7.1. Description

Our official cross language run (UMass3) used the INQUERY search engine, the Khoja stemmer (with unbreakables) for Arabic, the *kstem* stemmer for English [5], and query expansion of the English query, dictionary lookup of query terms in the small dictionary. The steps were as follows:

1. Remove stop phrases from English queries.
2. Remove stop words from English queries
3. Expand the English query
4. For each English word:
    a. Look for a set of translations in all of the English to Arabic lexicons described above
    b. If not found, stem the English word using the *kstem* stemmer and look it up again. Use all translations found in the dictionary.
    c. Stem the Arabic translations
    d. If any of the translations consist of an Arabic phrase rather than a single word, enclose the phrase in a **#filreq** operator. **#filreq** is like a Boolean *and*. If this version of INQUERY had proximity information, we would have used phrase or ordered window operators instead, but these were not available.
    e. If a set of translations was found, enclose all the alternatives in a **#syn** (synonym) operator
5. 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 above in the query expansion section.
6. Submit the weighted sum query to Inquery to retrieve Arabic documents.

7.2. Results

Table 3: Cross Language Results – official run – Inquery, expanded English, unexpanded Arabic

CONDITION	Name of Run	Average Precision	Number of Queries at or Above Median	
			Average Precision	Rel Ret in top 1000
Inquery baseline	not submitted	.1691		
Inquery stemmed	UMass3	.2795	20/25	20/25

Table 3 shows the results for the Cross Language run in the same format as the Table 1. In this case, query-by-query performance is compared with the median of 28 cross language runs, which include 2 French to Arabic, and 1 manual run. In 20 out of 25 queries, we performed at or above the median in both average precision and in the number of relevant documents returned in the top 1000.

Subsequent experiments showed improved results using the same general approach, but with the light stemmer, the large dictionary, and Arabic query expansion as well as English.

We compared the small and large dictionaries, described in Section 4.3.5.

Table 4: Comparison of small and large English-to-Arabic lexicons. Unexpanded cross-language retrieval

	norm	khoja-u	light8
Small lexicon	.1660	.2069	.3655
Large lexicon	.2624	.2514	.3794

Table 4 shows that the large dictionary performed substantially better than the smaller dictionary, in spite of the large number of translations for each word in the large dictionary.

The final set of experiments, summarized in Table 5, show that expanding both English and Arabic queries with the large dictionary and the light8 stemmer give the most effective cross-language retrieval. *Raw* means that no normalization or stemming were applied, *norm*, *khoja-u*, *khoja*, and *light* conditions refer to normalization only, Khoja stemmer with unbreakables, Khoja stemmer without unbreakables, and light stemming, respectively. *Light+khoja* is a combination of evidence run, in which scores from the *light* and *khoja* runs were averaged. Combination of evidence improves performance, but only slightly.

Table 5: Cross-language retrieval using large lexicon, different stemmers, and query expansion

	raw	norm	khoja-u	khoja	light	light+khoja
No query expansion	.1128	.2624	.2514	.2598	.3794	.3830
Expanded English	.1389	.3056	.2934	.3077	.4222	.4348
Expanded Arabic	.1544	.3371	.2917	.2931	.4106	.4189
Expanded English and Arabic	.1690	.3480	.3516	.3589	.4502	.4629

8. Acknowledgments

We would like to thank Shereen Khoja for providing her stemmer, Nicholas J. DuFresne for writing some of the stemming and dictionary code, Fang-fang Feng for helping with dictionary collection over the web, Mohamed Taha Mohamed , Mohamed Elgadi, and Nasreen Abdul-Jaleel for help with the Arabic language, Victor Lavrenko for the use of his vector and language modeling code and his advice.
This work was supported in part by the Center for Intelligent Information Retrieval and in part by SPAWARSYSCEN-SD grant number N66001-99-1-8912. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

9. References

[1] 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.

[2] 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.

[3] 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.

[4] 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, September 22, 1999.

[5] Krovetz, R. Viewing Morphology as an Inference Process. in *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1993, pp. 191-203.

[6] Larkey, L. S., Ballesteros, L., and Connell, M. E. *Improving stemming for Arabic information retrieval: Light stemming and co-occurrence analysis.* University of Massachusetts, Amherst, MA, CIIR Technical Report IR-249, 2002, submitted to SIGIR 2002.

[7] Miller, D. R. H., Leek, T., and Schwartz, R. M. A Hidden Markov Model Information Retrieval System. in *Proceedings of SIGIR '99: 22nd International Conference on Research and Development in Information Retrieval.* Berkeley: ACM Press, 1999, pp. 214-221.

[8] Ponte, J. M. A Language Modeling Approach to Information Retrieval. Dissertation, Department of Computer Science. Amherst, MA: University of Massachusetts, 1998, pp. 158 pages.

[9] Song, F. and Croft, W. B. A general language model for information retrieval. in *Proceedings of the Eighth International Conference on Information Knowledge Management (CIKM '99)*: ACM Press, 1999, pp. 316-321.

[10] Witten, I. H. and Bell, T. C. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37, pp. 1085-1094, 1991.

[11] Zhai, C. and Lafferty, J. 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.* New Orleans: ACM Press, 2001, pp. 334-342.

Important Cognitive Components of Domain-Specific Search Knowledge

Suresh K. Bhavnani
School of Information
University of Michigan
Ann Arbor, MI 48109-1092
Tel: +1-734-615-8281
bhavnani@umich.edu

ABSTRACT

Many users have acquired a sophisticated understanding of searching the Web in specific domains. For example, we often hear of users who can get amazing deals for electronic products on the Web. What knowledge do such users have, and how does it affect their search behavior? To address this question, we observed information retrieval experts in the domains of healthcare and online shopping, while they performed tasks within and outside their domains of expertise. When performing tasks within their domains of expertise, experts used declarative and procedural components of domain-specific search knowledge that enabled them to perform effective searches. In contrast, when they performed tasks outside their domains of expertise, they used a range of general-purpose search methods leading to comparatively less effective search results. The study demonstrates the role of domain-specific search knowledge, and pinpoints its cognitive components. The paper concludes by suggesting approaches that should make the components of domain-specific search knowledge explicit and available to many users.

Keywords

Domain-specific search knowledge, information retrieval.

INTRODUCTION

Despite huge advances in making information accessible to vast numbers of users, the effective retrieval of relevant information remains a challenge. Numerous user studies of different information retrieval (IR) systems repeatedly show that despite knowledge of basic search techniques, many users do not acquire strategic knowledge to find relevant information effectively [8, 9, 11].

To address this problem, several studies have attempted to identify effective IR strategies [1, 2, 5, 14], in addition to understanding the complex processes involved in search [4]. For example, early studies identified the *Building Block* strategy that specifies to break searches into smaller queries and then build it back to a larger one. More recent work attempts to predict browsing behavior such as when users continue to search within a site, and when they move on to another site [4].

The focus of such studies is on understanding *domain-general knowledge*, which is useful to perform tasks in many domains, whether one is searching for prices of a digital camera, or searching for health-related information. However, while domain-general knowledge may be important, is it sufficient for effective search?

Prior research has shown that searchers with *subject* knowledge in a domain know how to select terms that make them effective information searchers within specific domains [18]. However, evidence from Internet surveys [15], and from everyday experience suggest that some users have acquired domain-specific *search* knowledge that goes beyond knowing the subject-specific terms to enter in a query. For example, many university students often buy electronic gadgets at bargain prices on the Web because they know which sites to visit for different products, and in what order. This knowledge therefore appears to have declarative and procedural components that need a closer examination. What are the cognitive components of domain-specific search knowledge that such users have, and how does it affect their search behavior within their domains of expertise?

This paper attempts to identify the components of domain-specific search knowledge used by IR experts while they perform searches on the Web. In an exploratory study, IR experts in the domains of healthcare and online shopping were observed while they performed tasks within and outside their domains of experience. A fine-grained analysis of their interactions (using problem behavior graphs) and post-task interviews revealed declarative and procedural components of domain-specific search knowledge that enabled them to perform effective searches within their domains of expertise. In contrast, when they performed tasks outside their domain of expertise, they used a range of general-purpose search methods leading to comparatively less effective search results. The study therefore demonstrates the critical importance of domain-specific search knowledge, and identifies the specific components of such knowledge.

The paper concludes by suggesting approaches that make domain-specific search knowledge explicit and available to users in order to assist them in performing effective searches in unfamiliar domains.

EXPLORATORY STUDY TO IDENTIFY DOMAIN-SPECIFIC SEARCH KNOWLEDGE

The goal of our study was to identify the cognitive components of domain-specific search knowledge and their effects on search behavior. We therefore focused on recruiting participants who were experienced searchers in one of two domains (but not both), and observed them perform tasks within and outside their domains of experience.

Five healthcare search experts were recruited from three medical libraries on the University of Michigan campus. All healthcare search experts had six or more years experience accessing medical information and used Web browsers on a daily basis. Similarly, four online shopping experts were recruited from the student and recently graduated student community. All had three or more years of experience in shopping on the Web.

The participants were asked to perform eight tasks in two domains: four tasks related to healthcare, and four tasks related to online shopping. The tasks were adopted from the Text Retrieval Conference (TREC)[1], organized by the National Institute of Standards and Technology. (The entire set of TREC tasks is available from http://www.itl.nist.gov/iaui/894.02/projects/t10i/guidelines.html, and those selected for detailed analysis in this paper are described in the next section.)

The tasks were randomized within each domain, as was the order between the domains. Participants were told to perform the tasks, as they would normally do for themselves. They were asked to think aloud while performing the tasks and were reminded to keep talking if they stopped. After each set of tasks within a domain, they were asked questions in a structured interview regarding how they performed the task. The protocols, interactions, and interviews were recorded using screen capture tools.

ANALYSIS AND RESULTS

Our analysis and results focus on the interactions of all nine participants each performing the following two tasks:

> *1. Tell me three categories of people who should or should not get a flu shot and why?*
>
> *2. Get two price quotes for a new digital-camera (3 or more megapixel and 2x zoom). Stop when you feel you have found the lowest prices.*

The above *flu-shot task*, and *camera task* were selected based on the following criteria: (1) Tasks in which experts in each domain had the most experience; (2) Tasks that took the longest average time for the experts in each domain. These criteria were designed to choose those tasks that took a long time for experts to complete despite their expertise. The *flu-shot task* was within the domain of expertise for the healthcare search experts, but outside the domain of expertise for the online shopping experts. The opposite was true for the *camera task*.

Searching within and outside domains of expertise

Our initial observations revealed a difference in behaviors between participants performing IR tasks within and outside their domains of expertise. For example, when an IR expert in healthcare performed the *camera task*, her overall approach was to use a general-purpose search engine to find websites for cameras and their prices. She began by using the search engine alltheweb.com through which she found the site megapixel.net. Here she found a camera meeting the task criteria but found no prices. She therefore continued searching for websites containing cameras and prices in alltheweb, and after three more queries found CNET.com (a price-comparison and product-review site). She then returned to megapixel.net to retrieve the name and model of the camera, and searched for its price in CNET.com, where she found two prices for the camera. She ended the task by picking the lower price ($619).

The above search method relies on general-purpose knowledge or "weak methods" [12] such as entering and modifying queries in a general search engines. While such methods are useful to perform tasks in a wide range of domains, they are not as powerful as methods that are tailored to a specific domain.

In contrast to the above approach, an online shopping expert performed the same task using less general but more powerful methods that were specific to online shopping. His overall plan consisted of: (1) identifying cameras and prices from sites that provide reviews and prices such as Epinions.com; (2) comparing prices across vendors through mySimon.com that specializes in price comparisons; (3) searching for coupons from techbargains.com that apply to online stores such as STAPLES.com. He repeated the last two steps until he found a low price for a camera ($389) with features that exceeded the task requirements. Throughout the task, he accessed websites by directly typing their addresses or uniform resource locators (URLs) in the browser address field, and never used search engines such as Google.

The shopping expert's search behavior exhibits knowledge components ranging from how to sequence stages of the search, to knowledge of specific URLs. Such domain-specific search knowledge allowed him to perform an effective search with high-quality results. To understand more precisely the nature of these domain-specific components, we codified the interactions and developed formal descriptions of the behavior.

Codification of Interactions

The interactions of nine experts each performing two tasks were analyzed as working in a problem space [13]. A problem space is defined by a set of states, a set of

[1] TREC is a premier IR conference that specifies tasks that are researched by all participants of the conference. The goal is to set a baseline comparison across different research groups. We piloted and adopted the TREC tasks and guidelines for our research.

Operator	Definition	Example	
		Formal Description	Informal Description
Find-websites	Searches for one or more websites using a search engine.	**Find-websites** (Engine=Google Query="Digital Camera")	User accesses Google to enter the query "Digital Camera".
Scan-websites	Scans one or more websites with the same goal(s).	**Scan-websites** (Camera-models=? Review=? Feature>=2X, 3 MP Price<$400 TU=Epinions.com, CNET.com)	User has the goal of finding camera models and their reviews that meet the criteria of 2 times zoom, 3 megapixel, and a price less than $400 by visiting Epinions.com and CNET.com. Both sites are visited by typing in the URL (TU).

Compare	Compares information within a website or across websites.	**Compare**
		(Price<526?
		Vendor=?
		Vendor-reputation=high
		Camera-model=Olympus C-3030
		TU=PRICEWATCH.com)
			User has the goal of finding a camera price lower than $526 sold by a vendor whose reputation is high for the camera Olympus C-3030. He visits PRICEWATCH.com by typing in the URL.

Verify	Verifies information already found	**Verify**
		(Confirm=?
		List=9 categories of people
		CL=WYETH.com)
			User has the goal of verifying 9 categories of people [who should get a flu shot]. She does this by clicking on a link to visit WYETH.com.

End-task	Ends task either out of frustration, satisficing, or complete satisfaction. In some cases the experimenter informed the user that the task was completed.	**End-task** (Statement="That's the one to go for…I know this is a reputable site")	User has decided to buy a camera from a reputable site and ends the task.

Figure 1. High-level operators identified from the interactions and think-aloud protocols. Question marks denote a goal, TU and CL stand for *Type URL*, and *Click Link* [4]. The number and type of arguments for each of the above high-level operators vary depending on the interaction.

operators for moving between states, an initial state, a goal state, and a current state.

Identification of Operators

Through the analysis of protocols and interactions, we identified five operators that were sufficient to describe the search behaviors of the participants. Figure 1 shows definitions and examples for each of these operators. Because our goal was to analyze the overall search interaction, we defined operators at a higher level of abstraction than those used by Card et al., [4], in their development of similar graphs. We did, however, include their low-level operators as arguments of our high-level operators as different ways to visit websites. For example, the operator *Scan-websites* uses *Click Link* (CL), and *Type URL* (TU) to visit websites. Because the study by Card et al. restricted users to a single window, we added the argument *Click Existing Window* (CEW) to fully describe the behavior of our users.

Development of Problem Behavior Graphs

The operators were used to create descriptions of the search behaviors using problem behavior graphs (PBGs) [13]. Figures 2 shows an example of a PBG constructed from our data. The nodes of the graph (shown as boxes) represent knowledge states (knowledge that the user had acquired regarding the search task at a particular stage of the search process). The arcs of the graph (shown as arrows) represent the operators used to reach a particular knowledge state. Vertical lines represent backtracking to an earlier state (such as returning to the page of results in a search engine after following an unproductive link). Time therefore runs from left to right, and then from top to bottom. The information contained in the states contains the same arguments as those defined for the operators shown in Figure 1.

The PBG in Figure 2 describes the search behavior of an online shopping expert (S-1) who regularly shops for electronic gadgets on the Web. The PBG shows S-1's interactions at three levels. At the highest level of abstraction, his search had the following stages: *Review, Compare, Discount.*

At the next level of detail, the PBG describes the states of knowledge he passed through, in addition to the operators he used to move from one state to another. For example, within the *Review* stage, he scanned three sets of websites each with different goals (as shown by the operators between states 2, 3, and 4 in Figure 2).

Finally, the lowest level of detail is expressed by the arguments of the operators. These arguments show the user's goals, and the operator inputs. The states have the same arguments and represent the outputs of the operators. These arguments allow us to reconstruct the behavior of the user. For example, the first four states show how S-1 identified highly reviewed cameras and their prices to get a general understanding of the features available and their price range. He began with the goal of finding the model and price of a camera that has 2 times zoom, and 3 or more megapixel resolution (State-1). He then scanned Epinions.com looking for reviews of a camera less than $400 (based on prior knowledge of such gadgets) that meets

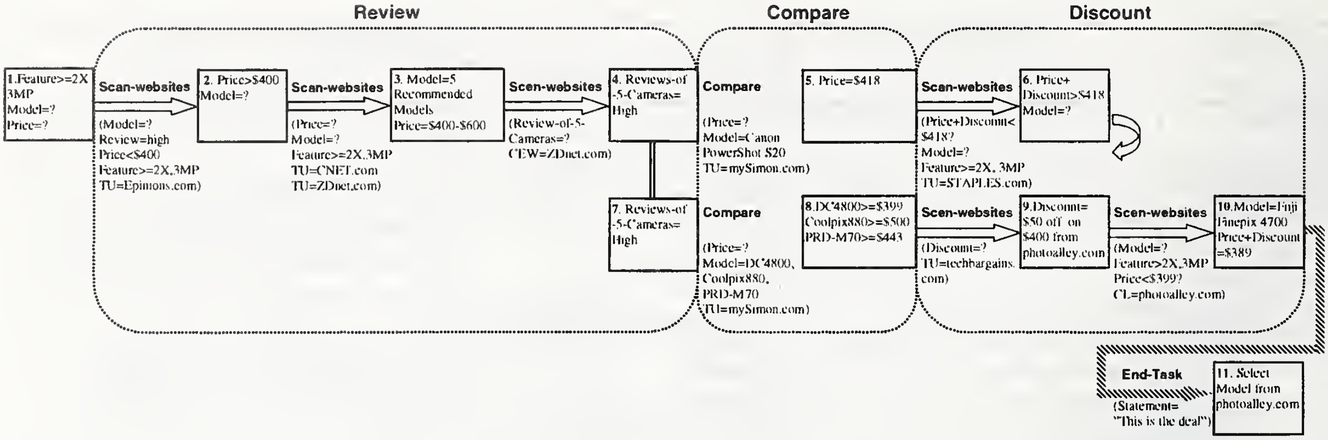

Review
Compare
Discount
1.Feature>=2X 3MP Model=? Price=?
Scan-websites
(Model=? Review=high Price<$400 Feature>=2X,3MP TU=Epinions.com)
2. Price>$400 Model=?
Scan-websites
(Price=? Model=? Feature>=2X,3MP TU=CNET.com TU=ZDnet.com)
3. Model=5 Recommended Models Price=$400-$600
Scen-websites
(Review-of-5-Cameras=? CEW=ZDnet.com)
4. Reviews-of -5-Cameras= High
Compare
(Price=? Model=Canon PowerShot S20 TU=mySimon.com)
5. Price=$418
Scan-websites
(Price+Discount< $418? Model=? Feature>=2X, 3MP TU=STAPLES.com)
6. Price+ Discount>$418 Model=?
7. Reviews-of -5-Cameras= High
Compare
(Price=? Model=DC4800, Coolpix880, PRD-M70 TU=mySimon.com)
8.DC4800>=$399 Coolpix880>=$500 PRD-M70>=$443
Scen-websites
(Discount=? TU=techbargains. com)
9.Discount= $50 off on $400 from photoalley.com
Scen-websites
(Model=? Feature>2X,3MP Price<$399? CL=photoalley.com)
10.Model=Fuji Finepix 4700 Price+Discount =$389
End-Task
(Statement= "This is the deal")
11. Select Model from photoalley.com

Figure 2. Problem behavior graph of S-1, an expert online shopper, performing the *camera task*. The graph shows his search behavior at three levels: (1) overall stages of the search method (Review, Compare, Discount); (2) knowledge states (boxes) and operators (arrows); (3) operator arguments specifying goals and interactions (text in boxes, and parentheses). The arguments TU, CL, and CEW stand for *Type URL*, *Click Link*, and *Click Existing Window* respectively as described in Figure 1. The back arrow represents the end of an abandoned search path, and the vertical lines represent a return to an earlier state.

those requirements but did not find any (State-2). Next, he scanned CNET.com and ZDnet.com to find a model meeting the same criteria and its price, and found 5 recommended models ranging in price from $400-$500 (State-3). Finally, he scanned ZDnet.com by clicking on an existing window in order to read reviews of the 5 models and found that all cameras had high ratings (State-4).

The PBG therefore makes salient, in a compact representation, the overall structure of search, in addition to the details of the interactions. Furthermore, the PBG provides a more precise understanding of the knowledge used during search. Such details are easily overlooked when analysis is based solely on observations or transcripts. Therefore, to identify the components of domain-specific search knowledge across the 9 participants, we developed and analyzed 18 PBGs in addition to analyzing transcripts of the interviews.

COMPONENTS OF DOMAIN-SPECIFIC SEARCH KNOWLEDGE

Analysis of 18 PBGs and transcripts of the interviews provided evidence for the declarative and procedural components of domain-specific search knowledge.

Declarative Components

Experts knew three types of declarative knowledge: (1) Classification knowledge consisting of classes of websites within a domain; (2) URL knowledge of specific websites; (3) Content knowledge consisting of the nature and type of information within a website.

Classification Knowledge

Search experts in both domains had well-formed classifications of websites within their domains. For example, the following quote is typical of a healthcare expert's classification of health-related websites:

> "I classify websites by the type of audience they are designed for, so if they are designed for the general population, consumers, and their families you know, that certainly is one big category. ... And then sites like MEDLINE, which bring you – which take you deep into the journal literature and very specific."

Shopping experts similarly classified shopping sites into review, comparison, discount, and product sites. However, they had far less clarity in their classification of healthcare sites:

> "I think there are two main [categories], one is just providing general information, or answer to questions, or stuff like that, and one is just about like hospitals - how you can make appointments, and other stuff. That's not information about uh diseases, but how to get medical care, how to schedule appointments and stuff like that. Other than that, I have no experience on medical sites".

URL Knowledge

Experts knew specific URLs of collections and sources when performing tasks within their domains. For example, all the healthcare experts directly typed in the URL of MEDLINEplus.gov, a collection that indexes a large number of reliable healthcare sources for consumers. Similarly, online shopping experts knew the URLs of sources such as Epinions.com that contain reviews and prices of mainly electronic products. Because experts knew the URLs of most of the sites they visited, there were relatively fewer sites visited by clicking on links. However, the reverse was true when they performed tasks outside their domain. Figure 3 shows this relationship by comparing the occurrences of *Type URLs* (TUs) and *Click Links* (CLs) for tasks within and outside their domains of expertise[2].

Content Knowledge

Besides knowing the URLs of sites, experts also knew the nature of information contained in the sites. For example,

[2] TUs and CLs that were used to revisit a site were not counted. The count is therefore a measure of the unique sites visited.

healthcare experts performed the *flu-shot task* by visiting 14 dot-org and dot-gov sites, compared to 3 dot-com sites that were commercial pharmaceutical websites that provided detailed information about side effects from the flu shot. The following exemplifies the healthcare experts' distinction between reliable government sources, and not so reliable sources that provide healthcare information:

> "I do not trust Adam.com. I would want other information to confirm that information before I would trust it. And I would be personally uncomfortable giving that information to a patient."

In contrast, the shopping experts performed the *flu-shot task*, by visiting 25 dot-com sites (out of a total of 29 total unique sites). Only 4 of those visited were dot-org and dot-gov sites.

Similarly online shopping experts knew which price comparison engines indexed small vendors ("mom and pop vendors") that typicaliy had lower prices but were not that reliable. On the other hand, they also knew which price engines had more reputable vendors.

Procedural Components

While the declarative components provide critical concepts within a domain, procedural components provide the methods to use such concepts to perform effective searches. Our analysis revealed that experts knew two types of procedural knowledge: (1) Sequencing knowledge that allowed them to order classes of websites in an overall search plan; (2) Termination knowledge that specified when to end a search.

Sequencing Knowledge

As shown earlier in Figure 2, the online shopping expert S-1 performed the *camera task* by sequencing different classes of websites (Review, Compare, Discount) to perform an effective search. All the shopping experts used variations of this sequence. Figure 4 shows two other online shopping experts S-4 and S-3 who also used similar sequencing for the camera task. S-4 first scanned websites such as CNET.com to review cameras, then visited sites such as PRICEWATCH.com to review cameras and compare prices, and finally looked for coupons and discounts at sites. like myCoupons.com. S-3 explicitly demonstrated only two of the stages by going to review sites like CNET.com, and comparison sites like mySimon.com. An analysis of the post-task interview revealed that S-3 had performed a quick mental calculation about a discount from amazon.com (based on his prior knowledge of discounts from Amazon), and decided that the discount was not worth it as the base price from that site was already too high. The *Review-Compare-Discount* sequence therefore appears to be an important domain-specific strategy for such online shopping tasks.

All the experts ın healthcare also used sequencıng knowledge. As shown in Figure 4, H-1 demonstrated the stages of first accessing a reliable collection like MEDLINEplus, and then finding sources for information within that collection. H-3 also followed these two stages but, in addition, verified the information by visiting a commercial pharmaceutical company that produces the flu vaccine. There she found a comprehensive and detailed list of people who should and who should not get a flu shot.

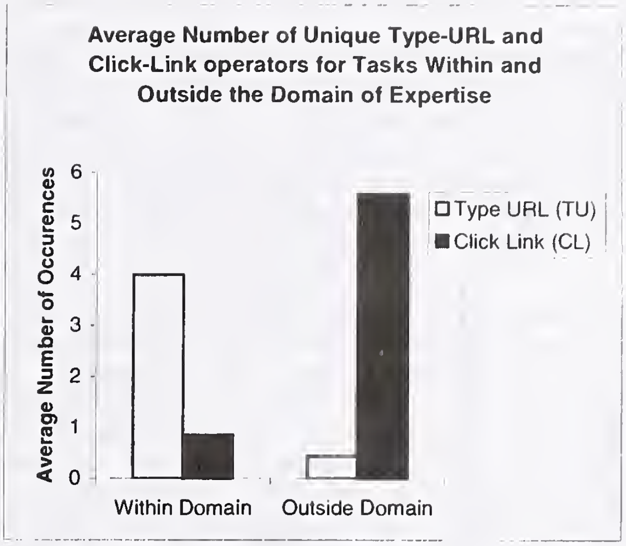


Figure 3. Reversal in the average number of TU and CU occurrences between search tasks within and outside domain of expertise. The differences between TUs and CLs are significant in both types of tasks (p<0.05 based on a two tailed t-test).

Such sequencing knowledge was absent in the search behavior of these very same experts when they performed tasks outside their domains. As shown by the solid black arrows (which represent finding a website by typing a query in a general-purpose search engine) in Figure 4, three participants used general-purpose search engines to perform tasks outside their domains. The healthcare expert H-1 aborted the task, as she could not find a price comparison engine through Google; H-4, with similar expertise, used the subject index in Yahoo to find digital cameras but because she did not look anywhere else for a price, ended up with an incorrect price due to an error in Yahoo's price of the camera. Online shopping expert S-4 used three general search engines (Google, AskJeeves, and YAHOO) to perform the *flu-shot task*, and S-3 used Google with many different queries to perform the same task. These variations demonstrate a continuum ranging from general and weak methods, to specific and strong methods as first described by Newell [12].

The absence of sequencing strategies (and the associated declarative knowledge) had a direct effect on the search results. The shopping experts found cameras on average at $60 less than the healthcare search experts. In addition, although the healthcare search experts did not explicitly search for more categories than required by the task, they found a comprehensive list of 9 categories of people who

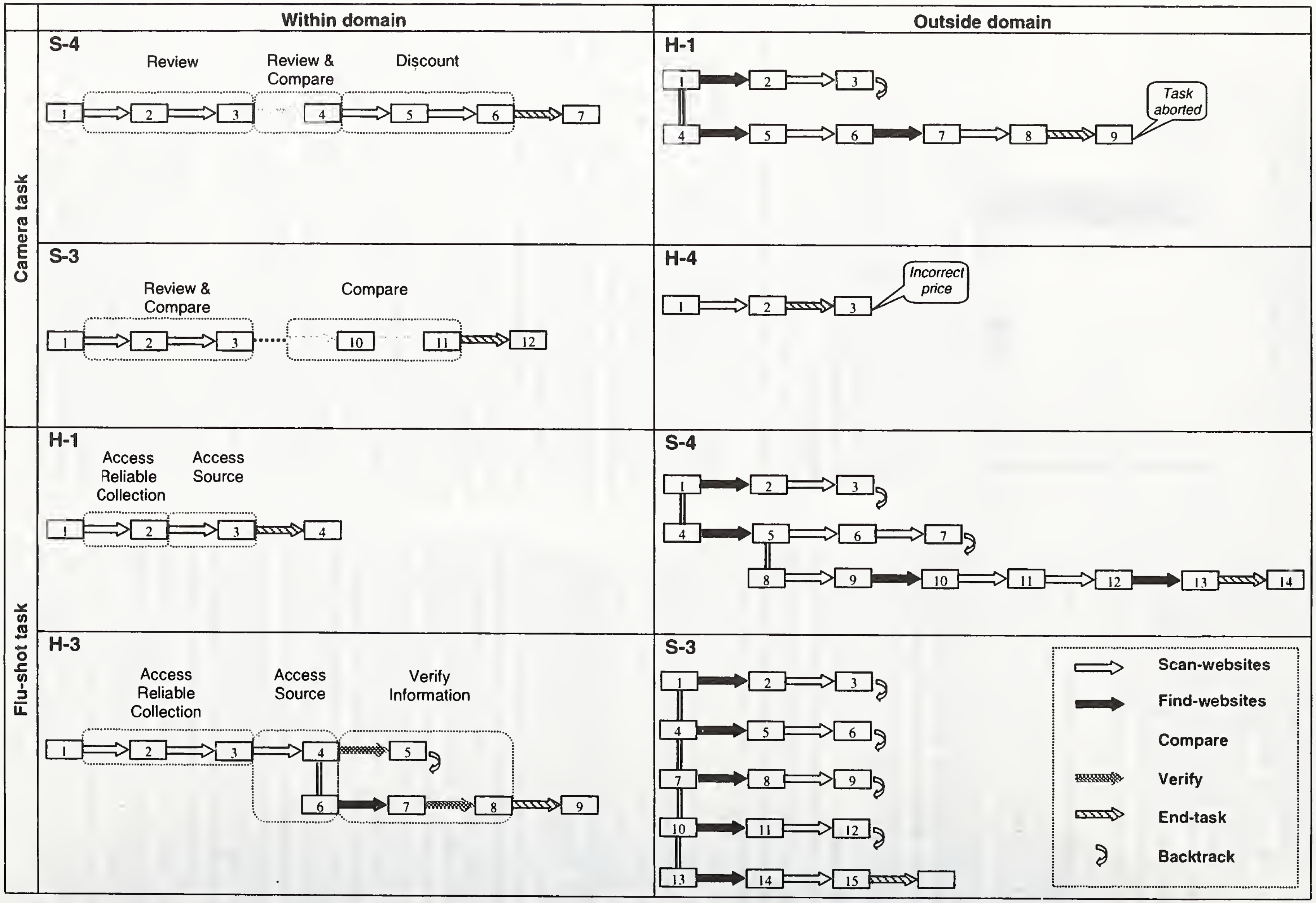


Figure 4. Eight problem behavior graphs of tasks performed within and outside domains of expertise. The above IR experts in shopping (S) and in healthcare (H) were chosen to show the variations we observed in sequencing knowledge. The within domain tasks have clear sequencing of classes of sites, whereas the tasks performed outside the domains of expertise are mainly performed using general-purpose search engines as shown by the solid black arrows.

should get a flu shot by going to an average of 3.7 reliable sites in very few steps. In contrast, none of the shopping experts found all the categories despite the fact that they visited on average 12 sites[3].

Experts, when performing tasks within their domains, therefore used sequencing knowledge to perform effective searches. However, these same experts used a range of weak methods to perform searches outside their domains leading to less effective search results.

Termination Knowledge

Experts in both domains demonstrated knowledge of when to terminate a search based on coverage. For example the shopping expert ended his search after checking three search engines:

> "So I had those prices to look at - three independent [price] search engines... It seems like a pretty good deal"

In contrast, all inexperienced shoppers terminated searches too early based on what appeared to be a perception of how long it would take to continue the search. None of them used more than one price comparison engine and were satisfied after finding a low price for just one camera. For example, after finding and comparing the price for a single camera a novice shopper stated:

> "Probably couldn't find anything lower... I wouldn't spend any more time fooling around with this, even for my own personal use".

DISCUSSION

The importance of domain-specific search knowledge should be no surprise to psychologists who have known the limited utility of domain-general knowledge or "weak methods" [6, 12] when solving non-trivial tasks within specific domains. These results should also be no surprise to librarians who have taken many classes during their education on sources and collections. However, we were still surprised to see the powerful effect such knowledge has on search behavior. One of the five reference librarians aborted the *camera task* because she lacked online shopping knowledge, while the other librarians performed largely ineffective searches compared to their expert counterparts. Similarly, the online shopping experts, who reportedly spend hours each day surfing the Web had strong skills in using browsers and search engines. However, despite this knowledge they performed relatively ineffective searches in a critical area such as healthcare. These results are therefore at odds with the growing confidence of large numbers of users who increasingly rely on general-purpose engines like Google to perform searches in many domains.

Appropriate to the nature of studies that do fine-grained analysis of interactions, our sample size is small. However we have found corroborative evidence for our results from a survey of Internet users [16]. This survey reports that most users, who search for health information on the Web, do so by searching directly through general-purpose search engines like Google. This is a critical problem given the prevalence of inaccurate, out-of-date, and often wrong information provided by a large number healthcare sites [3]. While this issue may not be as critical in online shopping, unreliable information can have serious consequences in the domain of healthcare. The identification and dissemination of domain-specific search knowledge should therefore be an important research effort in such domains.

There have been numerous attempts at organizing and making domain-specific knowledge available to users [7]. For example, sites like *InvisibleWeb.com* and *SearchEngineGuide.com* index numerous domain-specific search engines ranging from architecture to healthcare. However, such sites typically ignore the procedural components and the user is left to decipher first how to navigate the hierarchies to reach an appropriate search engine, and then how to sequence appropriate engines to perform a task. For example, we were unable to find reliable healthcare collections such as MEDLINEplus through either of the above sites, making it difficult to perform the *flu-shot task* effectively. Furthermore, even if we did find the appropriate search engines, there is no instruction on how to sequence them. Such tasks require much more knowledge than a search engine's URL.

Besides the above attempts to organize search engines across many domains, there are numerous portals at universities and organizations that provide lists of "helpful" links that are organized by expert librarians. For example, HealthWeb.org provides links to many different healthcare sources. Such sites provide the URL knowledge, but again do not make the procedural knowledge explicit to enable the selection and sequencing of sources and collections to perform specific tasks. The components of domain-specific search knowledge therefore help to pinpoint where such designs are lacking. This should lead to approaches to make the procedural components in addition to declarative components explicitly available to users so that they can make effective use of Web resources.

SUMMARY AND FUTURE RESEARCH

To understand the components and role of domain-specific search knowledge in information retrieval, we observed IR experts perform tasks within and outside their domain of experience. We found that experts were more effective when they used domain-specific search knowledge compared to when they had to rely only on domain-general knowledge. We also showed how this search behavior could be modeled adequately using problem behavior graphs at different levels of abstraction. Analysis of these problem behavior graphs and interviews helped us identify the declarative and procedural components of domain-specific search knowledge used by the participants.

Because current general-purpose search engines and portals do not provide all the components of domain-specific

[3] To enable a fair comparison, data from two participants who did not complete the entire task were not included in this calculation.

search knowledge, we are exploring approaches to make such knowledge explicit and available to users. Our identification of the components of domain-specific search knowledge should help to facilitate the design of structured interviews to elucidate the components in various domains. For example, very little is known about when experts decide to terminate a search, and structured interviews could enable the rapid elicitation of such knowledge from experts.

Once domain-specific search knowledge is made explicit, this knowledge could be made available to users through the design of classroom instruction. For example, there is an increasing need for healthcare professionals to understand how to retrieve accurate, reliable, objective, current, and comprehensive information [10] to make effective healthcare decisions. We are actively engaged in including domain-specific and domain-general knowledge in an IR course at the University of Michigan directed to freshman students entering the healthcare professions.

The components of domain-specific search knowledge could also be made available to users through new kinds of websites. For example, we are exploring the design of a new kind of site called a *Strategy Portal.* Such a portal will be designed to provide users classes of tasks within specific domains. For example, the online shopping domain would contain task categories such as "Shopping for an electronic gadget" and the healthcare domain would contain task categories such as "Prognosis for a disease". When a user selects a particular task category, the system will provide a sequence of recommended search stages as an overall plan for that task. For example, selecting "Shopping for an electronic gadget" will produce the search stages: *Review, Compare, Discount.* Each stage will provide links to specific sites including tips on how to evaluate information, and how to terminate tasks. The goal of such efforts is to prevent what we saw all too often in our study: users typing terms like "flu shot", and "digital camera" in Google and getting thousands of hits, and then ending searches more out of exhaustion rather than systematic coverage.

Our study has shown the importance of domain-specific search knowledge and helped to identify its components. Consequently, these results may help to design future systems and training, which should take into account many more components of search knowledge compared to what general-purpose search engines and portals provide today. This could lead users to become more effective when searching for information in unfamiliar domains.

ACKNOWLEDGMENTS

This research was supported by the National Science Foundation, Award# EIA-9812607. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of NSF or the U. S. Government. The author thanks P. Anderson, X. Lu, S. Mathan, M. O'Grady, P. Redman, F. Reif, and G. Vallabha, for their contributions.

REFERENCES

1. Bates, M. The design of browsing and berrypicking techniques for the online search interface. *Online Review* 13, 5 (1985), 407–424.
2. Belkin, N., Cool, C., Stein, A., and Thiel, U. Cases, scripts, and information-seeking strategies: on the design of interactive information retrieval systems. *Expert Systems with Applications* 9, 3 (1995), 379-395.
3. Biermann, J.S., Golladay, G.J., Greenfield, M.L. and Baker, L.H. Evaluation of cancer information on the Internet. *Cancer*, 86, 3 (1999), 381-90.
4. Card, S.K., Pirolli, P., Van Der Wege, M., Morrison, J.B., Reeder, R.W., Schraedley, P.K., and Boshart, J. Information scent as a driver of Web behavior graphs: results of a protocol analysis method for Web usability, in *Proceedings of the CHI' 01* (2001), 498 - 505.
5. Drabenstott, K. Web search strategies. In: *Saving the User's time through subject access innovation*; Papers in honor of Pauline Atherton Caochrane, W. Wheeler (ed.) Champaign, Ill.: University of Illinois (2000), 114–161.
6. Barr A., & Feigenbaum, E. *The Handbook of Artificial Intelligence* (v. 2), William Kaufmann, Los Altos, CA, 1982.
7. Ipeirotis, P., Gravano, L., and Sahami, M. Probe, Count, and Classify: Categorizing Hidden Web Databases in *Proceedings of the 2001 ACM SIGMOD* (2001), 67 – 78.
8. Jansen, B.J., Spink, A., and Saracevic, T. Real life, real users, and real needs: A study and analysis of user queries on the web. *Information Processing and Management*, 36, (2000) 207-227.
9. Kirby, M., and Miller, N. MEDLINE searching in Colleague: Reasons for failure or success of untrained end users. *Medical Reference Services Quarterly* 5 (1986), 17–34.
10. Kapoun, J. Teaching Undergrads Web Evaluation: A Guide for Library Instruction. *College & Research Libraries News* 59 (1998).
11. Marchionini, G. Information seeking in electronic encyclopedias, *Machine-Mediated Learning*, 3, 3 (1989), 21-26.
12. Newell, A. Heuristic programming: ill-structured problems. In J. Aronofsky, (ed.), *Progress in operations research*, New York: Academic Press (1969), 361-414.
13. Newell, A., and Simon, H. *Human Problem Solving*. Prentice Hall, New Jersey, 1972.
14. O'Day, V., and Jeffries, R. Orienteering in an information landscape: how information seekers get from here to there, in *Proceedings of INTERCHI '93* (1993), 438-445.
15. Pastore, M. Discounts, marketing practices working against e-tailers. Available at http://cyberatlas.internet.com/markets /retailing/article/0,,6061_809501,00.html. (July 26, 2001).
16. Pastore, M. Online health consumers more proactive about healthcare. Available at http://cyberatlas.internet.com /markets/healthcare/article/0,,10101_755471,00.html. (April 30, 2001).
17. Pirolli. P., and Card. S. K. Information Foraging. *Psychological Review*, 106 (1999), 643-675.
18. Shute, S., and Smith, P. Knowledge-based search tactics. Information Processing & management, (1993) 29, 1, 29-45.

The QUANTUM Question Answering System

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

We participated to the TREC-X QA main task and list task with a new system named QUANTUM, which analyzes questions with shallow parsing techniques and regular expressions. Instead of using a question classification based on entity types, we classify the questions according to generic mechanisms (which we call *extraction functions*) for the extraction of candidate answers. We take advantage of the Okapi information retrieval system for one-paragraph-long passage retrieval. We make an extensive use of the Alembic named-entity tagger and the WordNet semantic network to extract candidate answers from those passages. We deal with the possibility of no-answer questions (NIL) by looking for a significant score drop between the extracted candidate answers.

1 Introduction

We shall describe here our new question answering system called QUANTUM, which stands for QUestion ANswering Technology of the University of Montreal. QUANTUM was designed specifically for the TREC-X QA-track based on our experience with XR^3, which we used last year at TREC-9 [LKL00]. We shall introduce the architecture and the performance of the version used for the main task. Then, we explain how it was adapted to the list task. We did not participate to the context task.

2 Components of questions and answers

Before we describe QUANTUM, let us consider question # 302[1] and its answer shown in Figure 1. The question is divided 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 section 3.3.2, 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 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 using world knowledge.

*Work performed while at the University of Montréal.

[1] Whenever we cite a question from a TREC competition, we indicate its number. Questions 1–200 are from TREC-8, 201–893 from TREC-9 and 894–1393 from TREC-X.

$$\textbf{Question:}\quad \underbrace{\textit{How many}}_{\textit{question word}}\ \underbrace{\textit{people}}_{\textit{focus}}\ \underbrace{\textit{die from snakebite poisoning in the US per year?}}_{\textit{discriminant}}$$

$$\textbf{Answer:}\quad \underbrace{\textit{About 10 people}}_{\textit{candidate}}\ \underbrace{\textit{die a year from snakebites in the United States.}}_{\textit{variant of question discriminant}}$$

Figure 1: Example of question and answer decomposition. The question is from TREC-9 (# 302) and the answer is from the TREC text collection (document LA082390-0001).

We shall use the term *candidate* to refer to a word or a small group of words, from the text collection, that the system considers as a potential answer to the question. For the purpose of TREC-X, a candidate is seldom longer than a noun phrase or a prepositional phrase.

In this article, the term *answer* designates the string that results from the expansion of a candidate to a 50-character string.

3 System architecture for the main task

The input for the QA main task is a question set and a text collection. The system must output a ranked list of five 50-character answers to each question. We describe the 5 steps that QUANTUM follows when performing this task.

3.1 Question analysis

To analyze the question, we use a tokenizer, a part-of-speech tagger and a noun-phrase chunker. These general purpose tools were developed at the RALI laboratory for purposes other than the TREC QA-track. A set of about 40 hand-made analysis patterns based on lexical form, grammatical and noun phrase tags are used to determine the most appropriate extraction functions to apply. Table 1 shows the 11 function we have implemented. Each function triggers a mechanism for the extraction of candidates in a passage that can involve the passage's syntactic structure or the semantic relations of its component noun phrases with the question focus. More formally,

$$C = f(\rho, \varphi)$$

where f is the extraction function, ρ is a passage, φ is the question focus and C is the list of candidates found in ρ. Each element of 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.

We observed that in most TREC-9 QA systems a class fits a particular type of entity that the system is able to identify: toponyms, proper nouns, animals, weights, lengths, etc. In order to pair a question with an expected type of entity, one needs to anticipate all possible question forms that could focus on this type of entity. This introduces a supplemental difficulty, given the large number of possible reformulations of a question.

However, a lexical and syntactic analysis of all possible forms of English questions — that are applicable to TREC — showed that the number of required search mechanisms is rather limited. By considering these mechanisms (our 11 functions) as classes, we facilitate the question classification task because the number of classes is small and because the classes are closely related to the syntax of questions. Even though the number of classes in such a function-based classification is smaller than in an entity-based classification, we can achieve the same level of precision by parameterizing our functions with the question focus when needed. The automated process of parameterizing a generic mechanism can suit questions about virtually any kind of entities, whereas an entity-based classification is limited to the entities it contains. In the worst cases, the chosen function f and parameter φ could lead to a generic, non-optimal search. Yet the correct answer can still be retrieved.

3.2 Passage retrieval and tagging

The extraction of candidates is a time-consuming task. Therefore, we look for the shortest, albeit most relevant, passages of the text collection. We tried two different techniques: variable-length paragraphs retrieved with Okapi and fixed-length passages retrieved with our own algorithm based on last year's XR^3 system.

Extraction function	Example
definition(ρ, φ)	*(# 897) What is an* **atom***?*
specialisation(ρ, φ)	*(# 910) What* **metal** *has the highest melting point?*
cardinality(ρ, φ)	*(# 933) How many* **Great Lakes** *are there?*
measure(ρ, φ)	*(# 932) How much* **fiber** *should you have per day?*
attribute(ρ, φ)	*(# 894) How* **far** *is it from Denver to Aspen?*
person(ρ)	*(# 907) Who was the first woman to fly across the Pacific Ocean?*
time(ρ)	*(# 898) When did Hawaii become a state?*
location(ρ)	*(# 922) Where is John Wayne airport?*
manner(ρ)	*(# 996) How do you measure earthquakes?*
reason(ρ)	*(# 902) Why does the moon turn orange?*
object(ρ)	Default function

Table 1: The analysis of a question determines which function to use for extracting candidates. An extraction function is a generic search mechanism, sometimes parameterized by the question focus φ. Examples of classified questions are provided with their focus in boldface.

Category	Weight
Quoted strings	20
Years	10
Named entities	10
Noun phrases of more than one word	10
Capitalized nouns	2
Common nouns	1

Table 2: Question keywords are fitted into one of these categories and their weight is set accordingly.

3.2.1 Variable-length passages with Okapi

Okapi is an information retrieval engine that has the ability to return relevant paragraphs instead of whole documents [RW98]. 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).

3.2.2 Fixed-length passages

We also tried our own passage retrieval algorithm in a different run. We first build a list of keywords from the question. Keywords are fitted into the categories listed in Table 2 and a weight is attached accordingly to each of them.

Then, the best 200 documents returned by the IR engine PRISE (provided by NIST to all participants) are scanned for the keywords. The 250-character-long strings centered around every keyword occurrence constitute our fixed-length passages. The score of a passage is the sum of the weights of all the keywords it encloses. Passages from the same documents that overlap by more than 125 characters are discarded if they both have the same score. The remaining passages are ranked according to their score and the 50 best ones are kept.

3.2.3 Passage tagging

Once we have found the most relevant passages, we run our tokenizer, our tagger and our noun-phrase chunker on them because those information are needed by the candidate extraction functions. We also feed them into a named entity extractor. Last year, we used hand-built regular expressions for named-entity tagging, but this year, we used the freely available version of the Alembic Workbench system[2] developed at Mitre Corporation for the Message Understanding Conferences (MUC) [ABD+95]. Table 3 lists the types of entities that Alembic can identify and that we use.

3.3 Extraction and scoring of candidates

3.3.1 Extraction

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 (see section 3.1). During this phase, we seek the best recall rate possible, no matter whether candidates are cited in a context that matches the question discriminant. Table 4 shows some examples of what extraction functions look for.

[2] Downloaded from www.mitre.org/resources/centers/it/g063/workbench.html

Entity	Example
<PERSON>	Persons (*G. Washington*, *Mr. George Washington*), titles (*the President*)
<ORGANIZATION>	Full name of organizations and acronyms (*NATO*, *Congress*)
<LOCATION>	Toponyms (*Lake Ontario*, *North Africa*)
<DATE>	Dates (*Sep. 12, 1943*), years (*1983*), months (*February*), days (*Monday*)
<TIME>	Times (*23:03:12*, *4 a.m.*, *8 o'clock*)

Table 3: Named entities recognized by Alembic and used by QUANTUM. An exhaustive description of these categories can be found in [ABD$^+$95].

Extraction function	Example of criteria
definition(ρ, φ)	Hypernyms of φ
specialisation(ρ, φ)	Hyponyms of φ
cardinality(ρ, φ)	Pattern: NUMBER φ
measure(ρ, φ)	Pattern: NUMBER UNIT φ
attribute(ρ, φ)	Various patterns
person(ρ)	<PERSON> entities
time(ρ)	<TIME> entities
location(ρ)	<LOCATION> entities
reason(ρ)	Not implemented for TREC
manner(ρ)	Not implemented for TREC
object(ρ)	Any noun phrase

Table 4: Sample of extraction mechanisms for each extraction function.

3.3.2 Scoring

The final score of a candidate is the sum of three partial scores: the extraction score, the passage score and the proximity score.

Extraction score The score s_i awarded to a candidate by an extraction function is called the *extraction score*. It depends on the technique used for extracting a candidate. Typically, we award a higher score to a candidate extracted by the named entity extractor or by hand-made patterns; a candidate extracted because it satisfies some WordNet hypernym/hyponym relation is given a lower score because of the higher risk of introducing noise.

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. This score is the one given to a passage during its retrieval by either Okapi or our fixed-length passage retrieval algorithm. 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.

Proximity score The combination of the extraction score and passage score favours candidates that have the type we looked for and that are related to the question context. We also give a *proximity score* to candidates contiguous to noun phrases that contain a question keyword. By *contiguous*, we mean that they are not separated by another noun phrase. This way to measure proximity is rather crude and its effectiveness is still to demonstrate; therefore, we choose a relatively low proximity score to minimize its influence. At least, this score is helpful to break a tie between two candidates.

3.4 Candidate expansion to 50 characters

We expand a candidate by taking the 50-character document substring that is centered around it. Then, we cut off truncated words at both ends, which allows us to shift the substring to the right or to the left so that the new 50-character string contains the maximum number of complete words. The purpose is to maximize the chances that the string contains the correct candidate in the unfortunate case where QUANTUM would have guessed wrong. The effect of chance is not to be neglected since we measured a MRR score improvement of 0.03 with our last year system, XR^3, only by expanding candidates.

Candidate expansion takes place in conjunction with a redundancy elimination process. We begin by expanding our best candidate. Then, the second best candidate is expanded only if it does not appear in the first answer. The third candidate is expanded only if it does not appear in a previous answer, and so on until we have the desired number of answers. To keep a better diversity of candidates, we eliminate duplicate candidates even if they do not come from the same document, even at the risk of eliminating a supported occurrence of a candidate to the benefit of an unsupported one. However, we find such a probability to be very low since only 1.5 % of the TREC-9 question set had a correct but unsupported answer found by the system we used last year.

3.5 No-answer questions

Until now, we have assumed that the answer of the question could be found in the text collection. However, this might not be the case: a *NIL* answer may thus be the correct answer indeed. To deal with this, we examine our candidates to determine whether a *NIL* answer should be amongst our 5 suggestions of answers and, if so, at what rank.

Since score scales differ from question to question (particularly when different extraction functions are used), we cannot use a unique score threshold below which we can say that a *NIL* answer is more likely than a low-score answer. Instead, we have used a threshold that depends on the score drop between two candidates and we have normalized it so that it can be applied to the candidates of any question.

Let a_i be the answer at rank i and δ_i^{i+j} be the score difference between a_i and its j^{th} successor a_{i+j}. We compute the normalized score drop Δ_i between a_i and a_{i+1} in the following manner:

$$\Delta_i = \frac{\delta_i^{i+1}}{\delta_i^{i+4}} = \frac{s_i - s_{i+1}}{s_i - s_{i+4}}$$

where s_i is the score of a_i. Our choice to normalize over a 5-rank score difference δ_i^{i+4} is arbitrary, though our experiments showed that the following observations still hold for normalization over different intervals.

We ran QUANTUM on the TREC-9 questions and kept all answers that were extracted (not only the 5 best). We then applied the official correction script to spot the rank r of the first correct answer (when found). We computed Δ_r to measure the normalized score difference between a correct answer and its successor, which was a wrong answer. We also computed the average Δ_i for any pair of answers. We found that the score drop between a correct answer and its successor is slightly higher than the average score drop between any answer pair. Table 5 shows that this is true for different normalization intervals.

Having that in mind, we applied the following reasoning. Suppose we have two ranked, different answers, a_i and a_{i+1}. For simplicity, suppose also that their scores are different. Since a_{i+1} has a lower score than a_i, we assume that a_{i+1} has a lower probability than a_i to be correct. If the score drop Δ_i between the two is above average, we have an additional hint that a_{i+1} is incorrect. When Δ_i reaches a threshold Δ_t, we consider that a NIL answer is more likely than a_{i+1} (and than any other a_{i+j}, where $j < 1$). Thus, we keep a_i at rank i but as a second choice, we would rather say that there is no answer than submit a_{i+1} and we insert a *NIL* between the two.

Normalization interval	Δ_r	Δ_i
δ_i^{i+4}	33 %	29 %
δ_i^{i+3}	40 %	35 %
δ_i^{i+2}	56 %	50 %

Table 5: The normalized score drop Δ_r between a correct answer and its successor is slightly higher than the average normalized score drop Δ_i between any answer and its successor, regardless of the interval. Results were obtained by running QUANTUM on the TREC-9 question set.

If the system finds less than 5 answers and no score drop justifies the insertion of a *NIL*, we add a *NIL* answer after the last answer found.

The Δ_r we computed previously between a correct answer and its successor is a lower bound for a threshold on Δ_i above which a *NIL* is inserted. We set this threshold Δ_t experimentally by creating a set of 400 questions in which we knew that 5 % of questions had no answer in the text collection (the remaining questions were from TREC-9). We then chose the threshold value Δ_t that maximized the overall MRR score on this new question set. We obtained a maximum MRR score of 0.257 with $\Delta_t = 80$ %. However, we are aware that this threshold may not be optimal if the proportion of no-answer questions in a set is not 5 %.

Our technique based on score difference suffers a major handicap: it does not allow for the insertion of a *NIL* at rank 1 because Δ_0 does not exist. The only possibility for QUANTUM to put a *NIL* at rank 1 is when no answers at all are extracted. We believe that this situation arise far less often than no-answer questions are encountered in a question set because since our extraction functions were designed to achieve a high recall rate, they are more permissive than restrictive.

3.6 Final answer

The *final answer* is defined as the rank of the answer the system would give if it were allowed only one suggestion. It can be a number from 1 to 5 or the string *UNSURE*. Since our most confident answer is always put at rank 1, the *final answer* field is set to 1 for

Run	Confident	Correct
UdeMmainOk80	456 (92 %)	56 (12 %)
UdeMmainOk60	456 (92 %)	51 (11 %)
UdeMmainQt80	456 (92 %)	39 (8 %)

Table 6: Number of questions QUANTUM was confident for (out of 493 questions) and number of confident questions to which QUANTUM found a correct answer.

every question. However, when QUANTUM is unable to analyze correctly a question and therefore relies on the default function to find candidates, we set *final answer* to *UNSURE*. Thus, in our system, the *final answer* indicator is merely a binary flag to express confidence. Table 6 shows QUANTUM confidence for the TREC-X questions.

4 Results to the main task

We achieved a best-score of 0.191 to the main task by using Okapi for passage retrieval and a *NIL* threshold of 80 %. We submitted 3 runs:

UdeMmainOk80: This run uses Okapi for passage retrieval (length = 1 paragraph) and a Δ_t of 80% for the insertion of NIL answers.

UdeMmainOk60: This run uses Okapi for passage retrieval (length = 1 paragraph) and a Δ_t of 60% for the insertion of NIL answers.

UdeMmainQt80: This run uses our own fixed-length passage retrieval algorithm and a Δ_t of 80% for the insertion of NIL answers.

Table 7 indicates the official scores for these runs. The results confirm our first intuition: a *NIL* threshold of 80 % is better than a threshold of 60 % (compare runs UdeMmainOk80 and UdeMmainOk60). Our second intuition was also confirmed: Okapi is better at retrieving relevant passages than our own fixed-length passage retrieval algorithm is (compare runs UdeMmainOk80 and UdeMmainQt80).

5 System architecture for the list task

In the list task, the number of answers to provide is specified in the question. The QUANTUM architecture for the list task differs little from the main task. Question analysis patterns are adapted to extract the number of answers from the question (in case our patterns would fail to identify that number, we take the first number smaller than 50 to appear in the question). Passage retrieval is performed using Okapi only. The extraction of candidates is done by extraction functions described above. Candidate scoring and candidate expansion are different than for the main task. Of course, no *NIL* answers insertion needs to be done. We shall describe below the techniques we tried for the two runs we submitted: UdeMlistP and UdeMlistB.

Run	MRR	
	Lenient	Strict
UdeMmainOk80	0.197	0.191
UdeMmainOk60	0.189	0.183
UdeMmainQt80	0.145	0.137

Table 7: Official results of our 3 runs for the main task. The differences between the 3 runs reside in the passage retrieval technique used and the threshold Δ_t for insertion of *NIL* answers.

Run UdeMlistP

We use the same candidate scoring algorithm than for the main task, that is:

$$s = s_{extraction} + s_{passage} + s_{proximity}$$

However, candidates are not expanded by taking extra characters to the left and to the right. Instead, they are expanded to the right only. This is due to the fact that an unsuspected correct candidate that would appear before a known candidate in the 50-character string might, at best, make no difference in the overall accuracy and, at worst, interfere with the redundant candidate elimination algorithm.

Candidates are expanded and added to the list of answers as long as we have not reached the desired number of answers and as long as candidates are not redundant. For the purpose of the list task, a candidate is considered redundant when an identical candidate has already been expanded (note the difference with the main task, where a candidate was considered redundant if it appeared *anywhere* in an already expanded answer).

Run UdeMlistB

For this run, the scoring of candidates has been modified. Let $\mathcal{C}$ be the list of all candidates found and $\mathcal{F}$ be the list of their frequencies f sorted in decreasing order (with duplicate frequencies eliminated). The

Run	Accuracy
UdeMlistP	0.15
UdeMlistB	0.07

Table 8: Official results of our 2 runs for the list track. The differences between the 2 runs reside in the scoring of candidates.

score s' of a candidate is given by

$$s' = s^2 * log \frac{s_{passage}}{rank(f)} * n$$

where s is $s_{extraction} + s_{passage} + s_{proximity}$, $rank(f)$ is the rank, in $\mathcal{F}$, of the candidate's frequency and n is the number of candidates contained in the same document as the candidate currently scored.

Candidates are sorted again according to their new score. They are expanded to the right and they are eliminated when redundant, in the same manner as for the run UdeMlistP described above.

6 Results to the list task

As Table 8 shows, QUANTUM achieved its best score with the run UdeMlistP, which used the same scoring algorithm than its best run for the main task (UdeMmainOk80). The other run UdeMlistB reached an accuracy of 0.07 with its particular scoring algorithm.

7 Conclusion

Last year, we participated to TREC-9 for the first time with an all hand-built QA system that relied heavily on regular expressions. This year, we tried to improve the system by incorporating specialized resources (Okapi, WordNet and Alembic) and by devellopping a new classification based on extraction functions. These rely on semantic relations between terms in the question and in the passages and on syntactic analysis of the passages.

We did not yet evaluate the effects of each of these modifications, especially our patterns for question analysis and our function-based classification. Last year, we obtained a strict MRR of 0.179 and 0.149 for our 50-character runs. Unfortunately, this year's scores do not seem significantly higher than last year's. However, last year, when we trained our XR^3 system with the TREC 8 corpus, we obtained 0.386 and 0.331 and noticed a significant drop on the TREC-9 corpus. A similar drop this year may not indicate a decrease in performance of our system, but an increase in difficulty of the task. This still remains to be investigated.

Acknowledgments

We wish to thank Sylvain Laganière for his help on installing Okapi and making it run with the TREC document collection; and Luc Bélanger for his insight on the list task and for implementing the extraction of answer cardinality. We also would like to thank the Mitre Corporation for making Alembic Workbench available for research purposes.

This project was financially supported by a scholarship from the Quebec *Fonds pour la formation de chercheurs et l'aide à la recherche* (FCAR), the Bell University Laboratories (BUL) and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[ABD+95] 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, 1995. Morgan Kaufman Publishers.

[LKL00] M. Laszlo, L. Kosseim, and G. Lapalme. Goal-driven Answer Extraction. In *Proceedings of TREC-9*, pages 563–572, Gaithersburg, Maryland, 2000.

[RW98] S.E. Robertson and S. Walker. Okapi/Keenbow at TREC-8. In *Proceedings of TREC-8*, pages 151–162, Gaithersburg, Maryland, 1998.

Report on the TREC-10 Experiment: Distributed Collections and Entrypage Searching

Jacques Savoy, Yves Rasolofo

Institut interfacultaire d'informatique, Université de Neuchâtel (Switzerland)
E-mail: {Jacques.Savoy, Yves.Rasolofo}@unine.ch Web site: http://www.unine.ch/info/

Summary

For our participation in TREC-10, we will focus on the searching distributed collections and also on designing and implementing a new search strategy to find homepages. Presented in the first part of this paper is a new merging strategy based on retrieved list lengths, and in the second part a development of our approach to creating retrieval models able to combine both Web page and URL address information when searching online service locations.

Introduction

The Web of today represents a new paradigm, one that generates new challenges for the IR community. Included among these are: managing huge amounts of documents via distributed IR models, crawling through 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 very short user requests often expressed in ambiguous terms, satisfying a large range of search types (ad hoc, question-answering, location of online services, and interactive searches for specific document types or Web pages in order to satisfy a particular geographical or time constraint).

For our participation in TREC-10, we are focusing on two problems. One involves the presentation of a new merging strategy (collection fusion problem, Chapter 1) for the Web ad hoc track, and the other developing a search strategy intended to resolve homepage search problems (Chapter 2).

In order to evaluate our hypothesis when implementing the Okapi probabilistic model (Robertson *et al.*, 2000) we will use the SMART system as a test bed. This year our experiments are fully automated.

1. Distributed collections

In order to evaluate the retrieval effectiveness of various merging strategies, we formed four separate collections from the WT10g test collection (Savoy & Rasolofo, 2001). The same indexing scheme and retrieval procedure is used for each collection involved in this study. This type of distributed context more closely reflects digital libraries or search engines available on the Internet than do meta search engines, where different search engines may collaborate in response to a given user request (Selberg, 1999; Le Calvé & Savoy, 2000).

This chapter is organized as follows: Section 1.1 explains our indexing and search model. Section 1.2 describes related work on database merging strategies, while Section 1.3 presents our merging procedure. Finally, in Section 1.4 we evaluate our search model.

1.1. Indexing and retrieval scheme

From the original Web pages, we retained only the following logical sections: <TITLE>, <H1>, <CENTER>, <BIG>, with the most common tags <P> (together with </P>) being removed. Texts delimited by <DOCHDR>, </DOCHDR> tags were also removed. For longer requests, various insignificant keywords were removed (such as "Pertinent documents should include ..."). Moreover, search keywords appearing in topic title sections were assigned a term frequency of 3 (a feature that should have no impact on short requests).

For the ad hoc Web track, we conducted different experiments using the Okapi probabilistic model, in which the weight w_{ij} was assigned to a given term t_j in a document d_i and was computed according to the following formula:

$$w_{ij} = \frac{(k_1 + 1) \cdot tf_{ij}}{K + tf_{ij}} \qquad (1)$$

$$\text{with } K = k_1 \cdot \left[(1-b) + b \cdot \frac{l_i}{avdl} \right] \qquad (2)$$

where tf_{ij} indicates the within-document term frequency, and b, k_1 are parameters. K represents the ratio between the length of d_i measured by l_i (sum of tf_{ij}) and the document mean length is denoted by advl.

To index each search keyword t_j included in the request, the following formula was used:

$$w_{qj} = \frac{tf_{qj}}{k_3 + tf_{qj}} \cdot \ln \left[\frac{N - df_j}{df_j} \right] \qquad (3)$$

where tf_{qj} indicates the search term frequency, df_j the collection-wide term frequency, N the number of documents in the collection, and k_3 is a parameter.

To adjust the underlying Okapi search model parameters, we used the values suggested by Walker *et al.* (1998): advl = 900, b = 0.75, k_1 = 1.2, and k_3 = 1000. We did however believe that it might be more effective to assign a lower value to the parameter b, and in order to verify this assumption. We also evaluated the Okapi model using b = 0.7 or b = 0.5, values, resulting in interesting retrieval performances for TREC-9 topics.

Finally, for the request q containing m search terms the retrieval status value (denoted RSV_i) of a Web page d_i was estimated as:

$$RSV(d_i,q) = RSV_i = \sum_{j=1}^{m} w_{ij} \cdot w_{qj} \qquad (4)$$

In order to obtain a broader picture of our evaluations, we considered two different query formulations: (1) using only the Title section (T) or (2) all three logical sections (Title, Descriptive and Narrative, noted T-D-N). Finally, we should mention that these queries were "real topics" in the sense that they were taken from a MSNSearch log.

1.2. Previous work on merging strategies

Various solutions have been suggested for merging separate result lists obtained from distributed collections. As a first approach, and taking only the rank of the retrieved items into account, we might interleave results in a round-robin fashion. According to previous studies (Voorhees *et al.*, 1995; Callan *et al.*, 1995; Savoy & Rasolofo, 2001; Rasolofo *et al.*, 2001), such interleaving schemes have a retrieval effectiveness of around 20% to 40% below that achieved from single retrieval schemes, working with a single huge collection representing an entire set of documents.

In order to account for document scores computed for each retrieved item (or its retrieval status value), we might formulate the hypothesis that each collection is searched by the same or very similar search engines and that RSV values are therefore directly comparable (Voorhees *et al.*, 1995; Savoy & Rasolofo, 2001; Rasolofo *et al.*, 2001). Such a strategy, called raw-score merging, produces a final list sorted by the document score computed by each collection. However, as indicated by Dumais (1994), collection-dependent statistics contained in document or query weights may vary widely among collections, and therefore this phenomenon may invalidate the raw-score merging hypothesis.

To deal with this fact, we could normalize document scores within each collection through dividing them by the maximum score (i.e., the document score of the retrieved record found in the first position).

Callan *et al.* (1995) and Xu & Callan (1998) suggested a merging strategy called CORI, one that incorporates scores achieved by both collection and document. The collection score corresponds to the probability that the related collection would respond appropriately to the current request. In this scheme, each collection is viewed as a huge document and we might therefore use an IR scheme to rank the various collections according to the submitted request, since IR systems rank these documents according to their retrieval status values. In a second step, we simply multiply the document scores by the corresponding collection scores and then sort the result lists according to this value.

1.3. Our merging strategy

Our new merging strategy, denoted LMS for "using result Length to calculate Merging Score", and as does the CORI model, begins by estimating a score for each collection. The underlying idea is to use these weights to increase document scores from those collections having scores greater than the average score, and to decrease those for any collections having scores less than the average score. Our approach has the advantage of being simple, since it only uses document scores and result lengths as input. Also, since collection statistics are not required, systems using our approach do not need to store collection information. By contrast, when collections statistics are required within a dynamic environment such as the Web, they need to be updated frequently, and this is not possible without establishing some sort of cooperation between the main system and collection servers. Thus, our approach is more practical.

Our merging strategy consists of calculating a collection score according to the proportion of documents retrieved (result length) by each collection. This score is based on our intuition that a collection would contain more relevant documents for a given query if its collection server were to find more documents. The score for the *k*th collection is determined by:

$$s_k = \ln\left(1 + \frac{l_k \cdot K}{\sum_{j=1}^{|C|} l_j}\right)$$

where

- K is a constant (set to 600 in our evaluations),
- l_k is the number of documents retrieved by the *k*th collection, and
- |C| is the number of collections.

Our model uses a constant K in order to normalize the collection score as well as the natural logarithm, an order-preserving transformation used in similar contexts (Le Calvé & Savoy, 2000). Based on this collection score, our merging algorithm calculates the collection weight denoted w_k for the kth collection as follows:

$$w_k = 1 + \left[\left(s_k - s_m \right) / s_m \right]$$

where

- s_k is the kth collection score, and
- s_m is the mean collection score.

As in the CORI approach, the final document score is the product of w_k and the document score RSV_i computed by the server for this document. The value of this product is used as the key to sort the retrieved items in the final single result list.

1.4. Evaluation

To evaluate our propositions, we first used the TREC-9 topics (50 queries, 2,617 relevant documents) taken from the WT10g test collection. Average precision comparisons (computed by the TREC-EVAL system based on 1,000 retrieved items) achieved by the raw-score merging approach are depicted in the second column of Table 1a. On the other hand, average precision decreases with the use of round-robin merging strategy (third column of Table 1a). As one can see, considering result lengths in the merging process may marginally improve average precision over this baseline (last column of Table 1a).

After having considered different values for the parameter b, a smaller value (e.g., b = 0.5) seems to improve average precision (from 20.04 to 20.64, meaning an enhancement of +3% using raw-score merging or +2.6% when using our result length merging scheme (20.24 vs. 20.76)).

From studying the retrieval performance using TREC-10 topics (Table 1b), our previous findings were confirmed: the round-robin strategy results in lower average precision. From an analysis of parameter b, one can see that when setting b = 0.5, there is an improvement of +3.4% (from 16.59 to 17.16 in average precision). This fact is not however confirmed by longer requests, where the best value for the parameter b seems to be 0.7.

The data in Table 1b also indicates that by taking more search terms into account we can increase retrieval effectiveness substantially (around +30%). Finally, Table 3 provides a detailed analysis of our official runs applied to the Web ad hoc track, and Table 1b lists their retrieval performance in bold characters.

In order to analyze our merging strategy, we listed the number and the percentage of relevant items provided by each collection in Table 4. As one can see,

	Average precision (% change)		
Query (Title only) Model / merging strategy	TREC-9 50 queries Raw-score	TREC-9 50 queries Round-robin	TREC-9 50 queries Result lengths
Okapi (b=0.75)	20.04	17.66 (-11.9%)	20.24 (+0.9%)
Okapi (b=0.7)	20.34	17.96 (-11.7%)	20.60 (+1.3%)
Okapi (b=0.5)	20.64	17.66 (-14.4%)	20.76 (+0.6%)

Table 1a. Average precision of various retrieval schemes based on TREC-9 topics

	Average precision (% change)				
Query Model / merging	TREC-10 Title only Raw-score	TREC-10 Title only Round-robin	TREC-10 Title only Result lengths	TREC-10 Title-Desc-Narr Raw-score	TREC-10 Title-Desc-Narr Round-robin
Okapi (b=0.75)	16.59	16.43 (-1.0%)	17.15 (+3.4%)	22.12 (+33.3%)	20.39 (+22.9%)
Okapi (b=0.7)	16.73	16.24 (-2.9%)	16.99 (+1.6%)	22.42 (+34.0%)	20.76 (+24.1%)
Okapi (b=0.5)	17.16	16.03 (-6.6%)	17.50 (+2.0%)	21.68 (+26.3%)	19.87 (+15.8%)

Table 1b. Average precision of various retrieval schemes based on TREC-10 topics

Run name	Aver. pr.	Query	Parameter	Merging
UniNEtd	16.59	T	b = 0.75	raw-score merging
UniNEtdL	17.15	T	b = 0.75	result-length merging
UniNEt7dL	16.99	T	b = 0.7	result-length merging
UniNEn7d	22.42	T-D-N	b = 0.7	raw-score merging

Table 3. Description of official Web ad hoc run

the first collection (WT10g.1) contains a larger number of pertinent Web pages and the third collection (WT10g.3) a smaller number. From looking at the top 10, the top 100 and the first 1,000 retrieved pages, we can see how the percentage of pages extracted from each collection varies. More precisely, these numbers increase for the first and fourth collection and decrease for the other two.

Number of queries	50
Number of relevant doc.	3,363
Mean rel. doc. / request	67.26
Standard error	11.81
Median	39
Maximum	372 (q#: 541)
Minimum	2 (q#: 506, 538, 548)

Table 2. Relevance judgment statistics (TREC-10)

	Percentage of retrieved items			
	WT10g.1	WT10g.2	WT10g.3	WT10g.4
# rel. items	1007	800	640	916
% of rel.	29.94%	23.79%	19.03%	27.24%
Round-robin	25%	25%	25%	25%
Top 10	13.2%	28.8%	39.6%	18.4%
Top 100	19.54%	27.02%	30.62%	22.82%
Top 100	23.53%	25.16%	27.50%	23.82%

Table 4. Distribution of retrieved items (UniNEtd, raw-score merging, 50 topics)

2. Homepage searching

In the previous chapter, users sending a request to our search engine would obtain a ranked list of Web pages containing pertinent information about their information need. In this chapter, our objective is to design and implement a search strategy that would retrieve, at the limit, only one pertinent Web page that corresponds to the entrypage or to the online service location being sought by the user. For example, when users submit a request for "Quantas", they will retrieve the Quantas Airlines homepage, not several Web pages about this airline company.

To achieve this objective, we will first search URL addresses (Section 2.1). As a second search strategy, we will implement a combined retrieval model (Section 2.2) that searches the Web pages (Section 2.3) and then reranks the retrieved list by URL address length (Section 2.4). We will then examine any similarity between the query and the corresponding URL addresses (Section 2.5), and finally combine these three approaches (Section 2.6). An evaluation of our official runs is given in Section 2.7.

2.1. Searching thc URL address only

For each of the 1,692,096 Web pages included in the WT10g test collection, we know thc corresponding URL address. Thus as a first approach, we will build a text collection from these URLs and then obtain a mean number of distinct indexing terms (5.58 per URL address, max = 28, min = 1). From the available requests, we might then search this text database using the various IR models described using SMART notations (Savoy & Picard, 2001).

In this first attempt, we considered using a classical retrieval scheme to find the correct URL address (e.g., "www.cdsnet.net:80/vidiot/") when responding to the query "Vidiot". From examining usability studies, it was recommended that URL addresses contain information about the owner's name (usually the company name) and/or about content that might help users find their way around the Web or within the site (Nielsen, 2000, p. 246). If this principle is applied, our approach may work well.

	Simple queries		Extended queries	
IR model	MRR	# top 10	MRR	# top 10
Okapi	0.161	29	0.141	27
Lnu-ltc	0.077	15	0.091	17
atn-ntc	0.013	3	0.016	6
dtu-dtn	0.108	20	0.108	21
ltn-ntc	0.010	2	0.014	5
ntc-ntc	0.215	37	0.187	41
ltc-ltc	0.217	40	0.192	44
lnc-ltc	0.203	37	0.197	47
bnn-bnn	0.009	1	0.009	1
nnn-ntn	0.009	1	0.012	3
nnn-nnn	0.004	1	0.005	1
Mean	0.093	16.91	0.088	19.36

Table 5. Evaluation of URL searches (TREC-10, 145 topics)

In this first experiment, we evaluated eleven IR models, and as a retrieval performance measure, we calculated the mean reciprocal rank (MRR) over 145 topics (see Table 5, Column 2). As a second measure, we noted the number of topics for which a correct entrypage was found in the top 10 (see Table 5, Column 3). Overall, this search strategy does not work well for the problem of finding homepages. Moreover, although the Okapi probabilistic model provides the best search engine performance (Savoy & Picard, 2001), it is not best in terms of retrieval performance. For this particular retrieval task it seems that the vector-space model "doc=ltc, query=ltc" represents the best IR scheme, being able to retrieve 40 correct entrypages in the top 10 (over a total of 145, or 27.6% of the cases).

This rather limited performance thus reflects the fact that words found in an URL address are not necessarily those one would place in a query. For example, URLs often contain abbreviations (e.g., "www.iti.gov.sg" is the URL for "Information Technology Institute"). Moreover, if a given query contains very frequently occurring words (e.g., "of" "at", "in"), we add a second form of acronym that ignores these terms (e.g., from the request "Point of View Cafe", we form a first acronym as "povc" and a second as "pvc").

Concatenation represents another form of URL construction, with two (or more) words being joined (e.g., "www.dogzone.com" and "Dog Zone's dog clubs") or only the first (of the first two) letter(s) of a word are concatenated with the second word (e.g., "Digital Realms" expressed as "www.drealms.co.uk"). In order to deal with these various word formations, we designed our system such that it considers various URL construction possibilities. For example, for a simple request such as "Worldnet Africa", our system would construct the following expanded query: "Worldnet Africa wa worldnetafrica worldneta aworldnet woafrica worldnetaf".

Evaluating these extended request forms does not however result in appreciable performance enhancements, as can be seen in the last two columns in Table 5. Although the number of correct entrypages found in the top 10 seems to increase, the MRR measure indicates degradation. Thus, using only URL texts does not seem to be an adequate strategy, since establishing a link between query words and a URL addresses is a difficult task (e.g., based on the query "PiperINFO" which finds the URL "www.hamline.edu/" or "DaMOO" that finds the URL "lrc.csun.edu").

2.2. Guidelines for our combined search model

In order to develop a better search strategy, we decided to construct a two-stage retrieval strategy. In the first stage we used the Okapi probabilistic model to search Web page content for relevant homepages, although we did not employ the exact same Okapi search model used in the Web ad hoc track (see Section 1.1). Rather, we adapted the search model described in Section 2.3, and from the list of retrieved Web pages we were able to generate a corresponding list of URL addresses.

In the second stage of our retrieval strategy we inspected the corresponding URL addresses in order to verify whether or not they could be considered as appropriate URL candidates. To do so, we considered URL length, attributing more importance to short addresses (Section 2.4). Thus, in order to appear within the first positions in our final result list, a Web page would contain the search's keywords within its first 50 terms and its URL address would have to be short. As an alternative, we believe that URL address content should bear some similarity to the submitted request (Section 2.5), meaning we would again rank our retrieved list according to this similarity.

So far we have considered three types of retrieval expertise. The first retrieves and ranks Web sites according to page content, the second reranks these results according to URL address length and the last reranks the results according to URL address and submitted request similarity. In order to account for the results of these three approaches, we suggested reranking the retrieved items according to these three expert opinions (Section 2.6). Thus, with this search strategy, an entrypage will be found within the first positions if its Web page shares common words with the request, if its URL address is short and if this address contains some of the search keywords (or an abbreviation, a concatenation of two search keywords, or some URL construction as shown in Section 2.1).

2.3. Okapi model adaptation

In the first and most important stages of our combined retrieval strategy, we employed the Okapi probabilistic model to search Web page content for relevant homepages, although we did not employ the exact same Okapi search model as used in the Web ad hoc track (see Section 1.1). In fact, we added a precision device that considered only the first 50 words of each item retrieved and ranked only those documents having at least one query term within the first 50 words. This precision device was based on our intuition that document titles (or document headings) should provide an adequate indication as to whether or not corresponding Web pages are relevant to a given homepage search.

This feature works as follows. First, we form the set of all search keywords pairs. For example, from the query q = (t_i, t_j, t_k), we may deduce the following six terms pairs (t_i, t_j), (t_j, t_i), (t_i, t_k), (t_k, t_i), (t_j, t_k) and (t_k, t_j). We then inspect the document to see if the corresponding couple of search terms appears within a maximum distance of 5 (or with a maximum of four terms between the pair of search words). For example, supposing we are looking for the pair of search terms (t_j, t_k), then if we find a word sequence (t_j, t_a, t_b, t_c, t_d, t_k), we search it for an occurrence of our pair of search terms within a maximum distance of 5. The weight assigned to the occurrence of this search keyword pair (t_j, t_k) in the document d_i is denoted as δw_{jk} and computed as follows:

$$\delta w_{jk} = 0.5 + {}^{5}\Big/\sqrt{\text{position}(t_k)}$$

where position(t_k) indicates the position (word number) of the term t_k from the beginning of the Web page. Thus, if the term t_k appears in the second position (in this case, the first position occupied by t_j), the function position(t_k) returns 1 and δw_{jk} is 5.5. On the other hand, if the distance is greater than 5, we ignore this occurrence.

It is possible however that a pair of search keywords (t_j, t_k) would appear more than once (within a maximum distance of 5) in the document d_i. To account for all occurrences of the search term pairs (t_j, t_k), we compute the following expression:

$$w_{i(j,k)} = \left[\frac{(k_1+1)\cdot \sum\limits_{occ(j,k)} \delta w_{jk}}{K + \sum\limits_{occ(j,k)} \delta w_{jk}}\right] \cdot \min(w_{qj}; w_{qk})$$

where $w_{i\,(j,k)}$ represents the weight attached to the search term phrase (t_j, t_k) in the document d_i, the parameter k_1 and K are evaluated as described in Equation 2 and w_{qj} and w_{qk} represent the weights of the search keyword t_j and t_k in the request (as shown in Formula 3).

In order to consider all occurrences of all search keywords pairs, we compute an additional weight $ph_{(d_i,q)}$ attached to the presence of these multiple search keywords pair occurrences in document d_i as follows:

$$ph_{(d_i,q)} = \sum_{\text{all }(j,k)} w_{i(j,k)}$$

The presence of search keyword pairs within the beginning of a given Web page d_i would change the value of its RSV_i, and this would be done simply by adding the phrase weight $ph_{(d_i,q)}$ to the previously computed RSV_i (see Equation 4), as follows:

$$RSV_i' = RSV_i + ph_{(d_i,q)} \qquad (5)$$

However, we suggest a variation that assigns a lower phrase weight $ph_{(d_i,q)}$ if the document d_i does not appear in the top ranked retrieved documents. Thus an alternative retrieval status value is assigned using the following formula:

$$RSV_i' = RSV_i + (1-\alpha)\cdot ph_{(d_i,q)} \qquad (6)$$

$$\text{with } \alpha = \frac{RSV_{max} - RSV_i}{RSV_{max} - RSV_{min}}$$

where RSV_{max} and RSV_{min} are the RSV values assigned by the first and the last retrieved items respectively (computed according to Equation 4).

Run name	MRR	# in top 10	# not found
Okapi stem	0.261	65 (44.8%)	31 (21.4%)
Okapi nostem	0.274	72 (49.7%)	29 (20.0%)
Eq.5, stem	0.348	85 (58.6%)	26 (17.9%)
Eq.5, nostem	0.354	84 (57.9%)	26 (17.9%)
Eq.6, stem	0.343	83 (57.2%)	24 (16.6%)
Eq.6, nostem	0.367	86 (59.3%)	24 (16.6%)

Table 6. Evaluation of results using various Okapi probabilistic models (TREC-10)

Table 6 lists the various results of our search models, in which we reported the mean reciprocal rank (MRR), the number of queries for which the correct homepage was found within the first ten retrieved items, and the number of queries for which the relevant entry-page could not be found in our result list. Row two of this table contains an evaluation of the classical Okapi probabilistic model, as described in Section 1.1. As a variation in this particular search problem, we decided to analyze the impact of the stemming procedure, and as seen in row three, we were able improve retrieval effectiveness compared to the classical Okapi model, when the stemming procedure was discarded.

In the fourth and fifth rows are listed our adapted Okapi model's retrieval performance, based on the retrieval status values computed according to Equation 5. Finally, the last two rows show the performance achieved by using Equation 6 with our adaptation of the Okapi model.

From analyzing this data we concluded that the stemming procedure was not really appropriate for this type of search. Moreover, our modified Okapi model provides better results than does the classical Okapi model, and computing retrieval status value using Equation 6 exhibits the best retrieval performance.

2.4. Reranking based on URL length

Upon examining the result lists obtained using our Okapi homepage search model, we can find corresponding URL addresses using a look-up procedure and pass this list to one of our reranking schemes. In this second step, we first consider the URL address length, its length being defined by the number of "/"s contained within it. If however a URL address ends with a "/", then this final slash is ignored. Also, for any URL addresses ending with "index.html" or "index.htm", these terms are removed before we compute the length. Thus, the URL "www.ibm.com" has the same length as "www.ibm.com/index.html" or "www.ibm.com/".

URL length	Number of observations		
	Number	Percentage	Cumul.
= 1	11,622	0.69%	11,622
= 2	253,250	14.97%	264,872
= 3	458,356	27.09%	723,228
= 4	451,255	26.67%	1,174,483
= 5	297,339	17.57%	1,471,822
= 6	119,035	7.03%	1,590,857
= 7	69,091	4.08%	1,659,948
= 8	19,612	1.16%	1,679,560
= 9	6,399	0.38%	1,685,959
= 10	1,235	0.07%	1,687,194
≥ 11	4,902	0.29%	1,692,096

Table 7a. URL length distribution

Table 7a shows the URL length distribution across our test collection, while Table 7b depicts the same distribution based on our two relevance sets (entrypage 2000 and entrypage 2001). From the training data (denoted "2000"), we found that correct answers usually correspond to short URL addresses, those with lengths of 1 (77 cases in Table 7b) or 2 (15 observations). Data from the Entrypage 2001 test collection provided by relevance assessments displays a similar pattern. Thus it seems reasonable to assign more importance to short URL addresses than to longer ones.

	All relevant items		One rel. item / query	
URL length	2000	2001	2000	2001
= 1	79	138	77	93
= 2	19	56	15	32
= 3	8	33	7	9
= 4	2	16	1	6
= 5	0	7	0	4
= 6	0	0	0	0
= 7	0	2	0	1
Total	108	252	100	145

Table 7b. URL length distribution for a set of relevant items

Based on these findings, we reranked the retrieved URL addresses according to the inverse of their length, with ties being broken by using retrieval status values (RSV'$_i$) computed according to our Okapi model (Section 2.3).

Table 8 shows the first five URLs retrieved after the first query, according to our adaptation of the Okapi model (top part) or according to our reranking scheme based on URL length (second part). As one can see. the Okapi model retrieved various Web pages from the same Web site ("africa.cis.co.za"). The retrieval status value computed according to this search model, as depicted in Column 4, does not vary greatly. When we reranked this result list according to the inverse of URL length, the relevant item ("africa.cis.co.za:81") appears in the first position. However, the first five URLs have the same length (1 in this case), and the second key used is always the retrieval status value computed by the Okapi system.

2.5. Reranking based on URL similarity

URL address length does not however account for similarity between request and URL terms. Based on the data depicted in Table 8, for the response to "Worldnet Africa" we can see that the URL address "www.kvvp.com" response is listed in second place. Thus, it seems a good idea to rerank the result list provided by our Okapi model, based on a similarity between the URL address and the request.

This type of scheme is advantageous because we can account for any similarity between requests and Web pages, as well as requests and URL addresses. To compute this similarity between requests and URL addresses, we must however take various phenomena into consideration, including acronyms and concatenations of two search keywords found in URL addresses, etc. (see Section 2.1).

The basic principals underlying our similarity measure are described in Table 9, where we distinguish mainly between three cases. First, when a request is one word only, our similarity function determines whether this word appears in the URL head (defined as the server name) or in the URL's last component (defined as the tail of the URL). If it does and the corresponding URL is short (with a length of 1 or 2), we return the maximum value of 1.0. One the other hand, if this search term appears within the URL, the similarity is defined as the inverse of the URL's length. Finally, we determine whether there might be a fuzzy match between the search keyword and the URL. In this "fuzzySimilarity()" function, we counted the maximum length of the ordered sequence of letters between the search word and each term appearing in the URL. For example, for the search word "market" and the word "markCie", the maximum ordered sequence is "mark" which has a length of 4. This length is then divided by the maximum length of the two corresponding terms (7 in our case, giving a final fuzzy similarity of 4/7).

As a second case, we computed the similarity between the request and a URL with a length of one. Here we tried to establish a similarity between the request and some variations of this short URL (its acronym, the concatenation of adjacent word pairs, the concatenation of a word with the two letters of the next term).

			Similarity measurement		
Query	URL address	Rank	RSV'_i, Okapi	URL request	URL length
	based on our adaptation of the Okapi model				
1	africa.cis.co.za:81/nibs/postnet/ad.html	1	5242.61	0.25	0.25
1	africa.cis.co.za:81/about/newprice.html	2	5140.34	0.333333	0.333333
1	africa.cis.co.za:81/buy/ad/kyalami/kyalami.html	3	5116.08	0.2	0.2
1	africa.cis.co.za:81/buy/ad/fasa/fbhs2.html	4	5110.59	0.2	0.2
1	africa.cis.co.za:81/cape/ctcc/business/taxation.html	5	5109.63	0.2	0.2
	rerank based on URL length				
1	africa.cis.co.za:81/	1	4967.93	0.9999	1.0
1	www.kvvp.com:80/	2	2672.09	0.12	1.0
1	www.krok.com:80/	3	2636.3	0.16	1.0
1	www.starhustler.com:80/	4	2560.11	0.18	1.0
1	www.lcrtelecom.com:80/	5	1987.4	0.2	1.0
	rerank based on the similarity URL - request				
1	africa.cis.co.za:81/	1	4967.93	0.9999	1.0
1	www.att.com:80/worldnet/	2	2607.29	0.997	0.5
1	www.legnetwork.com:80/worldnet.htm	3	2408.45	0.997	0.5
1	interknowledge.com:80/south-africa/index.html	4	2275.12	0.997	0.5
1	www.biodiv.org:80/africa.htm	5	2143.38	0.997	0.5
	Merged results from our three experts				
1	africa.cis.co.za:81/	1	9935.86	1.9998	2.0
1	africa.cis.co.za:81/facility.html	2	10086.1	0.778	1.0
1	africa.cis.co.za:81/fin&tegn/	3	9887.3	0.778	1.0
1	africa.cis.co.za:81/cpyright.html	4	9850.68	0.778	1.0
1	africa.cis.co.za:81/comp.html	5	9809.52	0.778	1.0

Table 8. Example of four ranking approaches for the request "Worldnet Africa" (relevant item depicted in bold)

For example as depicted in Table 9, the request "Worldnet Africa" and the URL "africa.cis.co.za:81" represents a similarity evaluated as 0.9999, and if no match was found, we applied our fuzzy match function.

In the latter case, we computed the similarity between each search keyword and a given URL (function inFuzzy()). To define the similarity measure, we took the number of matches, the length of the URL, the value of the match between the URL head and the URL tail into account, as shown in the last lines of Table 9.

In order to evaluate this reranking scheme, we ranked the URL address result list according to request their similarity. An example of the results of this reranking is shown in Table 8 (third part). For this query one can see the relevant item "africa.cis.co.za:81/" appears in the first position. The following four URL addresses have the same similarity value (depicted in fifth column of Table 8) and are ranked according to their retrieval status values, computed from our adaptation of the Okapi model (shown in the fourth column).

2.6. Evaluation and data mergers

So far, we have described two reranking schemes that might hopefully improve the ranking obtained from our adaptation of the Okapi model (for which the corresponding evaluation is shown in Table 6). Table 10a lists an evaluation of these two reranking schemes under the label "URL length" and "URL simil." The results depicted in this table indicate that we can enhance the mean reciprocal rank (MRR) and the number of queries for which the correct homepage can be found within the first ten retrieved items. Moreover, we may also decrease the number of queries for which the relevant entrypage cannot be found in our result list (having a size of 100). Based on these results, the best scheme seems to be that of reranking, based on URL length.

As shown in Table 8 however, each of our three search systems seems to retrieve different URL addresses. Thus, we have decided to combine these three expert techniques. To do so, we selected the first fifteen retrieved items from each of three result lists and separately added the corresponding similarities achieved by our three experts. This additive process was chosen because in other data merging contexts, it also provided the best results (Savoy *et al.*, 1997). The result lists are then sorted by URL length, with ties being broken by using the sum of retrieval status values computed by our Okapi model (Section 2.3). For example, the last part of Table 8 shows the first five retrieved items, listed according to this fusion scheme. For the first item, RSV'_i = 9935.86 because it was found by both the URL length expert (RSV'_i = 4967.93) and the URL similarity expert (RSV'_i = 4967.93). For the same reason, the new URL length is set to two and the new URL-request similarity is fixed at 1.9998 (= 2 · 0.9999).

```
similarity (aQuery, anURL) :  Real;
    queryLength := queryLength (aQuery);                        // john smith canada -> 3
    urlLength := urlLength (anURL);                             // www.ibm.com/uk/products/  ->  3
    urlHead := urlHead (anURL);                                 // www.ibm.com/uk/products/  ->  www.ibm.com
    urlTail := urlTail (anURL);                                 // www.ibm.com/uk/products/  ->  products

    if (queryLength = 1) {
        if ( in(aQuery, urlHead) & (urlLength = 1))  return(1);        // market  &  www.market.com/
        if ( in(aQuery, urlTail) & (urlLength <= 2))  return(1);       // market  &  www.iti.com/market/
        if  (in(aQuery, anURL))  return(1/urlLength);                  // market  &  www.iti.com/data/market/
        return (fuzzySimilarity (aQuery, anURL));                      // market  &  www.marketCie.ch/prod/data/
        }
    if (urlLength = 1) {
        expandQuery := acronym (aQuery);                        // chicago science center  +  csc
        if (in (expandQuery, anURL) >= 2)  return(1);           // chicago science center csc &  www.csc.science.com/
        if (in (expandQuery, anURL))  return(0.9999);           // chicago science center csc &  www.csc.com/
        expandQuery := concat (aQuery);                         // john smith  +  johnsmith
        if (in (expandQuery, anURL))  return(1);                // john smith johnsmith  &  www.johnsmith.com/
        expandQuery := concat2 (aQuery);                        // advice corp  +  adviceco
        if (in (expandQuery, anURL))  return(1);                // advice corp adviceco  &  www.adviceco.com/
        return (fuzzySimilarity (aQuery, anURL));               // advice corp  &  www.advicorp.com/
        }
    simHead := fuzzySimilarity (aQuery, urlHead);
    simTail := fuzzySimilarity (aQuery, urlTail);
    if (urlLength = 2) {
        if (simTail >= 0.8)  return (simTail);                  // sirius bar  &  www.store.com/sirius/
        if (simHead >= 0.8)  return (0.456)                     // iris corp  &  www.iris.com/ca/
            else return (0);                                    // iris corp  &  www.irt.com/canada/
        }
    aSetMatchFuzzy := inFuzzy (aQuery, anURL);                  // desk publ  &  www.publ.com/uk/desk/ -> (1, 0, 1)
    nbMatch := sizeSet (aSetMatchFuzzy);                        // (1, 0, 1)  ->  2
    if (nbMatch = 0)  return (0);                               // when no fuzzy match can be found
    if (urlLength - nbMatch <= 1) {                             // numerous matches
        if ((aSimHead >= 0.8) & (aSimTail >= 0.8))              // if good match with the head and the tail
            return (aSimTail - 0.1);                            // e.g., desk publ  &  www.publ.com/uk/desk  -> 0.9
        if (aSimHead >= 0.8)                                    // if good match with only the head
            return (aSimTail - 0.15);
            else return (0.234);                                // if numerous match inside the url
        }
    return (max (0.2, nbMatch / urlLength));                    // if some matches
```

Table 9. Outline of algorithm used to determine similarity between query and URL address

However, from considering only the top 15 items for each of our three search models, a maximum of 45 retrieved items per query could be obtained. In order to increase these results to 100 (and to help generate relevance assessments), we expanded this list by adding URL addresses found by our Okapi model.

Run name	MRR	# in top 10	# not found
Okapi only	0.367	86 (59.3%)	24 (16.6%)
URL length	0.653	112 (77.2%)	21 (14.5%)
URL simil.	0.470	95 (65.5%)	18 (12.4%)
Fusion	0.693	115 (79.3%)	10 (6.9%)

Table 10a. Evaluation of our three search models and their combinations (corrected results)

The evaluation of our combined search model is depicted in the last row of Table 10a. The retrieval performance seems to have increased when considering MRR values or the number of queries for which the relevant item appeared in the top ten records. Moreover, the combination of our experts seems to have a clear impact on the number of queries for which the retrieval system cannot find corresponding relevant items (last column of Table 10a). If our combined retrieval model seems to perform well, it also has a drawback in that it retrieves various items corresponding to the same Web site, as shown in the last part of Table 8. Thus incorporating a pruning process in our fusion scheme may hopefully enhance retrieval performance.

When we created our official results for the homepage search problem, we selected the wrong Okapi results list before considering our two reranking schemes and our combined approach. The evaluations based on this incorrect result list are shown in Table 10b, and they reveal the same conclusions as do our unofficial but corrected search schemes (Table 10a).

Run name	MRR	# in top 10	# not found
Okapi only	0.295	64 (44.1%)	38 (26.2%)
URL length	0.598	96 (66.2%)	21 (14.5%)
URL simil.	0.431	81 (55.9%)	31 (21.4%)
Fusion	0.637	100 (69.0%)	12 (8.3%)

Table 10b. Evaluation of our three search models and their combinations (official results)

2.7. Description of our official runs

Table 11 shows our official homepage search runs. The "UniNEep1" run corresponds to the search model merges described in Section 2.6. To produce the "UniNEep2" run in positions 45 to 50 we added the top five URL addresses found by our simple search model, as described in Section 2.1 (doc=nnn, query=ntn), if these URLs were not found previously. As depicted in Table 11, this feature does not have any significant impact on retrieval performance.

Run name	MRR	# in top 10	# not found
UniNEep1	0.637	100 (69.0%)	12 (8.3%)
UniNEep2	0.637	100 (69.0%)	11 (7.6%)
UniNEep3	0.529	99 (68.3%)	10 (6.9%)
UniNEep4	0.477	99 (68.3%)	16 (11.0%)

Table 11. Official run evaluation

The "UniNEep4" run represents a variation of our search model, based on the normalized merging of the URL address searches (more precisely the "doc=nnn, query=ntn" model using our both our extended queries (see Table 5)) and our adaptation of the Okapi model (search Web page content, Section 2.3). The last run labeled "UniNEep3" represents the combined search model based on the "UniNEep4" run (with reranking based on URL length and URL similarity).

Conclusion

The various experiments carried out within the Web track demonstrate that:

- our new merging strategy based on results list length may improve average precision slightly;
- using a lower value for the parameter b when dealing with short requests may improve retrieval performance;
- our adaptation of the Okapi model for the homepage search problem performs relatively well;
- reranking the URL addresses based on a combination of URL length and URL similarity with the request improves retrieval performance for our Okapi model.

Acknowledgments

The authors would like to thank C. Buckley from SabIR for allowing us the opportunity to use the SMART system. This research was supported by the SNSF (Swiss National Science Foundation) under grant 21-58'813.99.

References

Callan, J.P., Lu, Z. & Croft, W.B. (1995). Searching distributed collections with inference networks. In Proceedings of the ACM-SIGIR'95, 21-28.

Dumais, S.T. (1994). Latent semantic indexing (LSI) and TREC-2. In Proceedings of TREC'2, 105-115.

Le Calvé, A., & Savoy, J. (2000). Database merging strategy based on logistic regression. *Information Processing & Management*, 36(3), 341-359.

Nielsen, J. (2000). *Designing Web usability: The practice of simplicity*. Indianapolis: New Riders.

Rasolofo, Y., Abbaci, F. & Savoy, J. (2001). Approaches to collection selection and results merging for distributed information retrieval. In Proceedings ACM-CIKM'2001, 191-198.

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. & 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. In Proceedings TREC'9, (to appear).

Savoy, J., Le Calvé, A. & Vrajitoru, D. (1997). Report on the TREC-5 experiment: Data fusion and collection fusion. In Proceedings TREC'5, 489-502.

Selberg, E.W. (1999). Towards comprehensive web search. Ph.D. Thesis, University of Washington (WA).

Voorhees, E. M., Gupta, N. K. & Johnson-Laird, B. (1995). Learning collection fusion strategies. In Proceedings of the ACM-SIGIR'95, 172-179.

Walker, S., Robertson, S.E., Boughamen, M., Jones, G.J.F. & Sparck Jones, K. (1998). Okapi at TREC-6: Automatic ad hoc, VLC, routing, filtering and QSDR. In Proceedings of TREC'6, 125-136.

Xu, J. & Callan, J. P. (1998). Effective retrieval with distributed collections. In Proceedings of the ACM-SIGIR'98, 112-120.

Unbiased S-D Threshold Optimization, Initial Query Degradation, Decay, and Incrementality, for Adaptive Document Filtering

Avi Arampatzis

Information Retrieval and Information Systems, University of Nijmegen,
Postbus 9010, 6500 GL Nijmegen, The Netherlands.
avgerino@cs.kun.nl, http://www.cs.kun.nl/~avgerino

Proceedings of the Tenth Text REtrieval Conference (TREC-10).

Abstract

We develop further the *S-D threshold optimization* method. Specifically, we deal with the bias problem introduced by receiving relevance judgements only for documents retrieved. The new approach estimates the parameters of the exponential–Gaussian score density model without using any relevance judgements. The standard *expectation maximization (EM)* method for resolving mixtures of distributions is used. In order to limit the number of documents that need to be buffered, we apply *nonuniform document sampling*, emphasizing the right tail (high scores) of the total score distribution.

For learning filtering profiles, we present a version of Rocchio's method which is suitable and efficient for adaptive filtering. Its main new features are the *initial query degradation* and *decay*, while it is fully *incremental* in query updates and in calculating document score statistics. Initial query degradation eliminates gradually the contribution of the initial query as the number of relevant training documents increases. Decay considers relevant instances (documents and/or initial query) of the near past more heavily than those of the early past. This is achieved by the use of *half-life*, i.e. the age that a training instance must be before it is half as influential as a fresh one in training/updating a profile. All these new enhancements are consistent with the initial motivation of Rocchio's formula.

We, moreover, use a form of *term selection* for all tasks (which in adaptive tasks is applied repeatedly), and *query zoning* for batch filtering and routing.

1 Introduction

This paper describes the participation in the TREC-10 Filtering Track by researchers from the Katholieke Universiteit Nijmegen (KUN). We participated in all three subtasks: adaptive filtering, batch filtering, and routing. The description of the tasks and evaluation measures can be found in [10]. We have mainly used the FILTERIT system for all but one routing run which was made by the LCS system. Table 1 summarizes the runs we submitted.

Task	Optimized for	System	Run-tag
adaptive	T10SU	FILTERIT	KUNaU
adaptive	F05	FILTERIT	KUNaF
batch	T10SU	FILTERIT	KUNbU
batch	F05	FILTERIT	KUNbF
routing	—	FILTERIT	KUNr1
routing	—	LCS	KUNr2

Table 1: TREC-10 filtering runs submitted by KUN.

FILTERIT was developed for our TREC-9 participation [2]. It was initially designed as a pure adaptive filtering system, based on a variant of Rocchio's relevance feedback formula which is more suitable for adaptive tasks. It has recently been extended to provide mechanisms for batch training, non-adaptive filtering, and routing. LCS was developed in the context of the Esprit project DOcument ROuting (DORO)[1] [8]. It is based on the Winnow mistake-

[1] http://www.cs.kun.nl/doro

driven learning algorithm [6]. Both systems are described in length in [2]; here we will concentrate on the changes made in FILTERIT.

In the next section, the preprocessing applied to documents and topics is described. Section 3 expands on incremental profile training. It is shown how the initial query degradation and decay are integrated into Rocchio's method. Many technical details are given which may be proved useful for developing incremental and effective filtering systems. Section 4 deals with optimizing filtering thresholds. Finally, results are summarized in Section 5, and conclusions are drawn in Section 6.

2 Preprocessing

All tasks were performed using a keyword-based representation of documents and queries, with traditional stemming and stoplisting. There was no special treatment of proper names, all numbers were eliminated, and we made no use of multi-word terms such as phrases or word clusters. We did not use any external resources such as online dictionaries or thesauri. In summary, the pre-processing was quick-and-dirty. It reduces dramatically the number of indexing terms, however, it worked out well with the OHSUMED collection in the TREC-9 Filtering Track where we obtained very good results. This year, however, a programming bug in the preprocessor introduced a serious disadvantage in performing the adaptive tasks: all stems of the test stream which did not occur in the training stream were discarded. As a result, filtering profiles could not be expanded with new terms during adaptations. The impact of this error on the routing and batch filtering effectiveness, however, was negligible.

3 Incremental Profile Training

The query training of FILTERIT is derived from Rocchio's method [11]. Our version of the formula presents the following features:

1. It introduces *initial query degradation*. The initial query is considered as carrying a worth of a certain number of relevant documents. As a result, the contribution of an initial query in training a classifier decreases with the number of relevant training documents.
2. It incorporates the notion of the *half life* of a training document, i.e. the age that a document must be before it is half as influential as a fresh one in training a classifier.
3. It allows *accurate incremental training* without using any document buffers, resulting in low memory and computational power requirements.

Table 2 shows all the quantities used for incremental training. They are grouped (from top to bottom) into user-supplied parameters, document stream variables, filter variables, and system-wide parameters.

$\mathbf{Q}_0$	the initial user query vector
a	worth of $\mathbf{Q}_0$, in number of relevant docs
h	half life of training docs
N	total number of docs seen
$\mathbf{DF}_N$	diagonal $K \times K$ matrix of df's @ N
$\mathbf{Q}_n$	query vector after n training docs
$\mathbf{B}_n$	linear combination of relevant docs
$\mathbf{C}_n$	linear combination of irrelevant docs
a_n	worth of $\mathbf{Q}_0$ at time t_n
b_n	accumulated worth of relevant docs
c_n	accumulated worth of irrelevant docs
β'	relevant to irrelevant feedback ratio
k	term selection cutoff

Table 2: Parameters and variables.

The user-supplied parameters are three: the initial query $\mathbf{Q}_0$, its assumed worth a measured in number of relevant documents, and the half life h of training documents measured in actual time. This year, we used $a = 2$, a low value in comparison to last year's tasks where our formula behaved more like $a = 10$. The main reason for this was that we did not find the TREC-10 queries (consisting mainly of one or two keywords) too informative. Furthermore, we used a mild decay (despite the fast-changing nature of a news stream) setting the half life h to 6 months for the adaptive tasks, and no decay for batch and routing (due to the limited timespan of the training data). More about document decay and half life, and a discussion on *convergence* or *responsiveness* of classifiers can be found in [3] and [1].

Training documents $\mathbf{D}_i$, $i = 1, 2, \ldots$ (relevant and non-relevant) are the pre-classified documents given at the time of bootstrapping, and all retrieved ones during filtering since their relevance judgment is given. The index i denotes the position of a training document in time sequence of all training documents, e.g. $\mathbf{D}_1$ is the oldest. $\mathbf{Q}_0$ and $\mathbf{D}_i$ are *not* idf weighted, but only tf and length-normalized. The precise weighting scheme we currently use for documents is the *Lu* [12].

Stream variables are the total number N of documents seen by some point in time, and the $K \times K$ diagonal matrix $\mathbf{DF}_N = \mathbf{diag}(df_1, \ldots, df_K)$ of the document frequencies of a total K terms contained into the N documents. Each new document that arrives increments N by one, and updates $\mathbf{DF}_N$ (i.e. incremental df).

The filter variables are initialized as

$$\mathbf{B}_0 = \mathbf{C}_0 = [0, 0, \ldots, 0] , \quad b_0 = c_0 = 0 , \quad a_0 = a . \tag{1}$$

Now, let us assume that the system retrieves the nth document $\mathbf{D}_n$ at time t_n, and that the user feedback says that it is relevant. Then the filter variables are updated as

$$\begin{aligned} \mathbf{B}_n &= l_n \mathbf{B}_{n-1} + \mathbf{D}_n , \\ \mathbf{C}_n &= l_n \mathbf{C}_{n-1} , \\ a_n &= l_n a_{n-1} , \\ b_n &= l_n b_{n-1} + 1 , \\ c_n &= l_n c_{n-1} , \end{aligned} \tag{2}$$

where l_n is the *decay factor* calculated as

$$l_n = 0.5^{(t_n - t_{n-1})/h} . \tag{3}$$

Similarly, if $\mathbf{D}_n$ is non-relevant, then it is added to $\mathbf{C}_n$ instead of $\mathbf{B}_n$ and c_n is incremented instead of b_n.

The new filtering query $\mathbf{Q}_n$ is then built in 3 steps.

1. The *query zone* is built, i.e. a trained query using only the relevant and discarding all the non-relevant feedback:

$$\mathbf{Q}_{\mathbf{z},n} = \left(\frac{1}{a_n + b_n} (a_n \mathbf{Q}_0 + \mathbf{B}_n) \right) \mathbf{idf}(\mathbf{DF}_N) . \tag{4}$$

   (We use the same formula in batch filtering and routing to select which non-relevant documents are going to be used for training: currently, the top-500 non-relevant as ranked by Eq. 4.)

2. All terms in $\mathbf{Q}_{\mathbf{z},n}$ are ranked according to their weight, and only the top-k terms are selected. All the rest of the terms are removed from vectors $\mathbf{Q}_{\mathbf{z},n}, \mathbf{Q}_0, \mathbf{B}_n$, and $\mathbf{C}_n$. This year, we used 250 terms for routing and batch filtering, and 100 for adaptive filtering.

3. The new filtering query is calculated as

$$\mathbf{Q}_n = \beta' \, \mathbf{Q}_{\mathbf{z},n} - \frac{1}{c_n} \mathbf{C}_n \, \mathbf{idf}(\mathbf{DF}_N) . \tag{5}$$

The function $\mathbf{idf}(.)$ returns the diagonal matrix of the idf components. We should remind that $\mathbf{Q}_0$ and $\mathbf{D}_i$ are not idf weighted. The idf components are currently calculated with the t formula [12].

Note that for $h = +\infty$ (no document decay), $\beta' = 1$, and $a = 0$ (no initial query), the procedure we have just described calculates the original Rocchio formula. A relevance feedback setting with the traditional parameters α, β, and γ can be simulated using $a_n = b_n \, \alpha / \beta$ and $\beta' = (\alpha + \beta)/\gamma$. In short, our version of the formula can behave like most variants seen in the literature. Additionally, it allows accurate incrementality, it can consider the initial query as being an equivalent of some number of relevant training documents, and it incorporates the notion of decaying over time training documents and initial query. Moreover, it does not invalidate the initial motivation of Rocchio's formula. For example,

$$\frac{1}{c_n} \mathbf{C}_n = \frac{1}{\sum_i d_i} \sum_i d_i \mathbf{D}_i , \tag{6}$$

where $\mathbf{D}_i$ is a non-relevant document, and

$$d_i = \prod_{j=i+1}^{n} l_j = 0.5^{(t_n - t_i)/h} , \tag{7}$$

is still the (weighted) average vector of non-relevant documents.

3.1 Incremental Score Statistics

The incremental training we have just described allows us to calculate (weighted) score statistics, e.g. mean score and variance, incrementally without using any document buffers. Score statistics are necessary for thresholding as we will see in Section 4.

If the dot-product $*$ is used as a scoring function, the mean relevant document score $\mu_{\text{rel},n}$ at t_n is simply:

$$\mu_{\text{rel},n} = \frac{1}{b_n} (\mathbf{B}_n * \mathbf{Q}_n) . \tag{8}$$

The variance $\sigma^2_{\text{rel},n}$ can be calculated via

$$\sigma^2_{\text{rel},n} = \mu^{(2)}_{\text{rel},n} - \mu^2_{\text{rel},n} . \quad (9)$$

The mean of the squared scores $\mu^{(2)}_{\text{rel},n}$ is given by

$$\mu^{(2)}_{\text{rel},n} = \frac{1}{b_n} \left((\mathbf{Q}_n \, \mathbf{B}_{\text{dyad},n}) * \mathbf{Q}_n \right) , \quad (10)$$

where

$$\mathbf{B}_{\text{dyad},n} = \sum_i d_i \, \mathbf{D}_i^{\text{T}} \, \mathbf{D}_i \quad (11)$$

is 2-dimensional matrix. $\mathbf{D}_i^{\text{T}}$ denotes the adjacent of $\mathbf{D}_i$. $\mathbf{B}_{\text{dyad},n}$ can be updated incrementally as

$$\mathbf{B}_{\text{dyad},n} = l_i \, \mathbf{B}_{\text{dyad},n-1} + \mathbf{D}_n^{\text{T}} \, \mathbf{D}_n . \quad (12)$$

The derivations of Formulae 8 and 10 can be found in [1].

4 Threshold Optimization

Thresholds in FILTERIT are empirically optimized for batch filtering, and S-D (score-distributional) optimized for adaptive tasks.

4.1 The Empirical Method

The *empirical* technique for optimizing a threshold on training documents consists of the following steps: rank the documents according to their scores, calculate the effectiveness measure of interest at every position of the rank, find the position where the measure is optimal, and set the threshold somewhere between the score of the optimal position and the score of the next lower position. The technique works out well given sufficient training data. Our batch filtering runs, KUNbU and KUNbF, use this technique for optimizing thresholds.

The main drawback of the empirical technique becomes apparent when adaptivity is required, namely, it cannot be applied incrementally. A large document buffer should be carried along during filtering, and the scores of all its documents must be recalculated after every query update.

4.2 The S-D Method

The S-D method [2, 4] eliminates the need for a document buffer by using the statistical properties of scores rather than their actual values. Statistical properties like mean score and variance can be updated incrementally, as we have shown in Section 3.1.

The idea behind the S-D threshold optimization is the following. If relevant and non-relevant document scores are modelled separately using their probability densities, then the total score density is a (weighted) mixture of the individual score densities. Having determined the individual score densities and the mixing parameter, all measures that satisfy the *probability thresholding principle* (PTP) [5] can be optimized. The optimization of non-PTP measures requires moreover the knowledge of the number of documents to be filtered, but this is usually an unknown quantity. In such cases, the method can be applied by optimizing the measure only for the near future, e.g. for a certain assumed number of incoming documents.

The procedure has as follows. Let M be an effectiveness measure, a function of the four variables R_+, N_+, R_-, N_- (relevant retrieved, non-relevant retrieved, relevant not retrieved, etc.) of the traditional contingency table. The measure is calculated at m levels $i = 1, \ldots, m$, of decreasing score s_i as

$$M_i = M(R_+(s_i), N_+(s_i), R_-(s_i), N_-(s_i)) , \quad (13)$$

where, e.g., $R_+(s_i)$ gives the number of relevant documents that would have been retrieved for a threshold equal to s_i. That is

$$\begin{aligned} R_+(s_i) &= r \int_{s_i}^{s_0} P_{\text{rel}}(x) \, dx & (14) \\ &\approx R_+(s_{i-1}) + r \, (s_{i-1} - s_i) P_{\text{rel}}(s_i) , & (15) \end{aligned}$$

where P_{rel} is the probability density function of relevant scores, r is the number of relevant documents, and s_0 is the maximum possible score. The other three variables of the contingency table parameterized by the score can be similarly calculated.

Having calculated the M_i at all levels, the procedure goes on as in the empirical method. Note that Eq. 15 calculates numerically and incrementally a series of integrals. The method is simple to implement and efficient.

4.3 Score Distributions

The S-D optimization requires modelling of the score distributions of relevant and non-relevant documents. In [4] we introduced a numerical method for calculating the probability density of the score distribution of an arbitrary set of documents. The method needs as input the query and what we call *term probability* for each query term. The term probability of a term is simply the fraction of the documents in the set that it occurs in. Thus, the method has the desirable property of depending only on quantities which can be updated incrementally, and it does not need the actual documents. Nevertheless, it is computationally expensive.

In the aforementioned study, we also investigated whether the score distributions can be approximated with known distributions. Assuming that each score is a linear combination of the query weights (e.g. a dot-product), and that relevant documents cluster around some point in the document space with some hyper-ellipsoidal density (e.g. a hyper-Gaussian centered on the point), we proved that the relevant score distribution has a Gaussian Central Limit in a large number of dimensions (query terms). Moreover, we showed that the Gaussian limit appears fast. How fast depends also on the quality of a query; the better the query, the fewer the terms necessary for a Gaussian approximation. Practically, on the OHSUMED collection and for the 63 OHSU queries (TREC-9's data) trained with Rocchio on the 1st year of data, the relevant score distributions can be very well fitted with Gaussians at around 250 query terms.

In the case of the distribution of non-relevant document scores, we have empirically found in [2, 4] that the right tail (high scores) of the score density can be well fitted with an exponential. Further empirical evidence for the proposed Gaussian–exponential score modeling can also be found in [7].

4.4 The Bias Problem

Adaptive filtering presents a bias problem that arises from the fact that relevance judgements become available only for those documents retrieved. The implication of this for thresholding[2] is that calculating the mean score and variance or term probabilities only on documents retrieved can be very misleading.

An attempt to deal with the bias is seen in [13], where each document score is considered together with a *sampling constraint*, i.e. the current threshold at the time of retrieval of the corresponding document. Then the parameters of the Gaussian–exponential model are estimated by maximum likelihood. Although the method calculates unbiased S-D thresholds, it introduces new complications in updating the query. When the query is updated all sampling constraints change as well, nevertheless, there is currently no way of updating the sampling constraints. Abandoning query updates in exchange for a better threshold does not seem like a good solution for adaptive filtering.

4.5 Unbiased S-D: An EM Approach

We have developed another approach which calculates unbiased thresholds while allowing query updates. Since the problem arises from the fact that the relevance judgements are biased, we fit to the total score distribution a mixture model consisting of an exponential and a Gaussian, *without* using any relevance judgements. A standard approach to determining the mixing parameters and the parameters of the component densities is to use Expectation Maximization (EM) [9]. Recovering the parameters of the Gaussian–exponential score model with EM without relevance judgements has recently been described in [7] in the context of distributed retrieval.

Let $P(x|1)$ and $P(x|2)$ be the exponential and Gaussian densities respectively. The total score density can written as

$$P(x) = \sum_j P(j) P(x|j) \,, \tag{16}$$

where $P(j)$ are the mixing parameters satisfying

$$\sum_j P(j) = 1 \,, \quad 0 \le P(j) \le 1 \,. \tag{17}$$

The parameters to be estimated are four: the mean μ and variance σ^2 of the Gaussian, the λ of the exponential $P(x|1) = \lambda \exp(-\lambda x)$, and only the one of the two mixing parameters since the other can be determined from Eq. 17.

[2] Note that the bias problem in filtering does not show up only in thresholding, but also in query training/updating.

EM is an iterative procedure. The update equations for the discussed mixture model are:

$$P_{\text{new}}(1) = \frac{\sum_i P_{\text{old}}(1|x_i)}{\sum_i w_i}, \tag{18}$$

$$\lambda_{\text{new}} = \frac{\sum_i P_{\text{old}}(1|x_i)}{\sum_i P_{\text{old}}(1|x_i)\, x_i w_i}, \tag{19}$$

$$\mu_{\text{new}} = \frac{\sum_i P_{\text{old}}(2|x_i)\, x_i w_i}{\sum_i P_{\text{old}}(2|x_i)}, \tag{20}$$

$$\sigma^2_{\text{new}} = \frac{\sum_i P_{\text{old}}(2|x_i)|x_i - \mu_{\text{new}}|^2 w_i}{\sum_i P_{\text{old}}(2|x_i)}, \tag{21}$$

where $P_{\text{old}}(j|x)$ is given by Bayes' theorem

$$P_{\text{old}}(j|x) = \frac{P_{\text{old}}(x|j) P_{\text{old}}(j)}{P_{\text{old}}(x)}, \tag{22}$$

and $P_{\text{old}}(x)$ is given by Eq. 16.

In general, when all scores x_i have been obtained unconditionally, $w_i = 1, \forall i$. For thresholding purposes, however, we are interested in the right tail (high scores) of the total density. Moreover, in adaptive tasks, more and more scores are accumulated over time. Consequently, in order to reduce the total number of documents the system has to retain and to focus on the tail of the distribution, we apply *nonuniform sampling* of the documents according to their score. If x the score of a document, the document is sampled with probability $P_s(x)$. $P_s(x)$ should be an increasing function, so that more high than low scoring documents are collected. The sampling function we currently use is

$$P_s(x) = \exp\left(\frac{\log 1000}{x_{\max}}(x - x_{\max})\right), \tag{23}$$

where $x_{\max}$ is the maximum score. This sampling function retains most documents with scores near to $x_{\max}$, and only 1 out of a 1,000 documents with zero score.

The 20,000 (approximately) documents of the training set are first sampled like that for every topic. If after the sampling more than 1,000 documents remain, the buffer is further thinned down to 1,000 documents by uniformly (this time) discarding documents. The initial threshold is calculated on the scores of the remaining documents using EM, but now the scores x_i must be weighted as

$$w_i = \frac{1}{P_s(x_i)}. \tag{24}$$

As new documents accumulate, every time the buffer reaches the 2,000 documents, it is thinned down to 1,000 documents by random (uniform) removal. Note that if a profile update has taken place, all document scores should be recalculated for a threshold update. This can be computationally heavy for large documents buffers.

EM converges locally, this means that finding a global fit depends largely on the initial settings of the parameters. Initial values for the parameters of the Gaussian and exponential are selected randomly as

$$\mu_{\text{init}} \in [\mu_{\text{rel},n}/2,\ \mu_{\text{rel},n}], \tag{25}$$

$$\sigma^2_{\text{init}} \in [\sigma^2_{\text{rel},n}/4,\ \sigma^2_{\text{rel},n}], \tag{26}$$

$$\lambda_{\text{init}} \in [1/\mu_{\text{half lowest scores}},\ 1/\mu_{\text{all scores}}], \tag{27}$$

where $\mu_{\text{rel},n}$ and $\sigma^2_{\text{rel},n}$ are the biased parameters calculated using the formulae in Section 3.1. The initial mixing parameter is selected as

$$P_{\text{init}}(1) \in [0.5,\ 1 - b_n/N]. \tag{28}$$

To find a global fit, EM is run 10 times with initial parameters selected randomly from the ranges above. Then, the fit that has the least squared error with the empirical data is selected.

5 Results

Table 3 summarizes the official results we achieved in TREC-10. The rightmost column shows the final rank of the runs, and the number in parentheses is the total number of runs in the corresponding category submitted by all groups.

Run	T10SU	F05	Av.Prec.	Rank
KUNaU	0.203	0.437	—	12 (30)
KUNaF	0.141	0.356	—	12 (30)
KUNbU	0.307	0.507	—	4 (18)
KUNbF	0.264	0.489	—	8 (18)
KUNr1	—	—	0.136	4 (18)
KUNr2	—	—	0.137	3 (18)

Table 3: TREC-10 results of KUN.

In the routing task, the LCS system has performed very well this year (KUNr2), confirming that its bad performance in TREC-9 was due to the large number of Winnow-iterations that led to over-training. It has achieved a slightly larger average precision than the FilterIt system (KUNr1). According to those results our systems were ranked as the 2nd best.

The batch filtering runs (KUNbU and KUNbF) were performed by FilterIt. We used exactly the same parameter settings as for the routing run KUNr1, except that we thresholded the final document rankings using empirical thresholds estimated on the training set. Ironically, KUNbU optimized for T10SU resulted in larger F05 than KUNbF optimized for F05. The adaptive runs (KUNaU and KUNaF) were performed by FilterIt.

6 Concluding Remarks

Summarizing, we are satisfied with the profile training part of the FilterIt system. It is efficient since it allows incremental training, and it has proved effective as well. LCS and FilterIt are radically different systems, with different learning methods (Rocchio vs. Winnow), and different term selection and weighting schemes. Moreover, LCS did not use the initial queries at all. The fact that two so different systems have achieved similar results implies that we have either reached the "ceiling" of effectiveness for the current pre-processing and representation of the document collection, or the top-1000 documents used in the evaluation were not enough to distinguish between the two systems.

Concerning the threshold optimization for adaptive filtering, we have made a considerable step towards removing the bias introduced by the partial relevance judgements. However, numerous other parameters have been introduced that seem to require extensive tuning in order to achieve good end-results.

We have found EM especially "messy" and difficult to tune. It seems sensitive to the choice of the initial parameter values in converging to a global optimum rather than a local one. The update equations for EM which we have used, do not take into account the relevance judgements available. The available judgements have been used merely for determining usable ranges for initializing the parameters. Note that it may be possible to derive other update equations that will take into account the partial judgements. This may improve the quality of the fit.

Another source of inaccuracies lies onto the document sampling. The current sampling function is certainly not the best that can be used, considering the underlying total score distribution. The number of samples (1,000 to 2,000 max.) used did not seem enough for some topics. However, increasing the size of the document buffer introduces a serious computational overhead in threshold updates since all document scores must be recalculated after profile updates. A reasonable trade-off between threshold accuracy and efficiency has yet to be established.

Despite the "roughness" of these new methods we integrated into thresholding, and the fact that the "bug" in document preprocessing introduced a serious disadvantage into profile updates, our adaptive results ranked FilterIt above the median system.

Acknowledgments

I would like to thank Marc Seutter (KUN) for running KUNr2 with the LCS system, Panos Giannopoulos (University of Utrecht) and Kees Koster (KUN).

References

[1] A. Arampatzis. *Adaptive and Temporally-dependent Document Filtering.* PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, 2001.
Available from `www.cs.kun.nl/~avgerino`.

[2] A. Arampatzis, J. Beney, C. Koster, and T. van der Weide. Incrementality, Half-Life, and Threshold Optimization, for Adaptive Document Filtering. In *The Nineth Text REtrieval Conference (TREC-9)*, Gaithersburg, MD, November 13–16 2000. NIST.

[3] A. Arampatzis and T. van der Weide. Document Filtering as an Adaptive and Temporally-dependent Process. In *Proceedings of the BCS-IRSG European Colloquium on IR Research*, Darmstadt, Germany, April 4–6 2001.

[4] A. Arampatzis and A. van Hameren. The Score-Distributional Threshold Optimization for Adaptive Binary Classification Tasks. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New Orleans, September 9-13 2001. ACM Press.

[5] D. Lewis. Evaluating and optimizing autonomous text classification systems. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, July 1995.

[6] N. Littlestone. Learning Quickly when Irrelevant Attributes Abound: a NewLinear-threshold Algorithm. *Machine Learning*, 2:285–318, 1988.

[7] R. Manmatha, T. Rath, and F. Feng. Modeling Score Distributions for Combining the Outputs of Search Engines. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New Orleans, 2001. ACM Press.

[8] H. Ragas and C. H. A. Koster. Four Text Classification Algorithms Compared on a Dutch Corpus. 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*, pages 369–370, Melbourne, Australia, August 1998. ACM Press, New York.

[9] B. D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996.

[10] S. Robertson and I. Soboroff. The TREC-10 Filtering Track Final Report. In *The Tenth Text REtrieval Conference (TREC-10)*, 2001.

[11] J. J. Rocchio. Relevance Feedback in Information Retrieval. In *The SMART Retrieval System — Experiments in Automatic Document Processing*, pages 313–323, Englewood Cliffs, NJ, 1971. Prentice Hall, Inc.

[12] A. Singhal. AT&T at TREC-6. In *The Sixth Text REtrieval Conference (TREC-6)*, Gaithersburg, MD, November 19–21 1997. NIST.

[13] Y. Zhang and J. Callan. Maximum Likelihood Estimation for Filtering Thresholds. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New Orleans, 2001. ACM Press.

TREC10 Notebook Paper

Challenges of Multi-Mode IR Software

Gregory B. Newby
UNC Chapel Hill

Abstract

Web track results are presented. A software project, IRTools, is described. IRTools is intended to enable information retrieval (IR) experimentation by incorporating methods for multiple modes of IR operation, such as the vector space model and latent semantic indexing (LSI). Plans for the interactive track are described.

Introduction

For much of the past year, the author and his colleagues have been working towards general-purpose large-scale software for information retrieval experimentation. For TREC 10, the goal was to demonstrate this software's functionality using a "standard" IR approach: vector space retrieval. Functionality demonstrated in prior years' TRECs, notably the information space technique (Newby, 2001) and other approaches related to LSI (described in Rehder et al., 1998) was present but untested for TREC 10.

Submitted runs for the TREC10 Web track were irtLnua and irtLnut:

irtLnua:	Web track, all terms minus stopwords, Lnu.Ltc weighting
irtLnut:	Web track, title only, minus stopwords, Lnu.Ltc weighting

We also did some work on cross language retrieval but did not submit runs. Our work for the interactive track will not be completed in time for presentation at TREC10, but should be ready for the final proceedings.

Software Overview

We believe there is a lack of free, open source, high performance software for information retrieval. We desire to create software with these qualities:

1. Free and open source (e.g., licensed under the General Public License);
2. With implementations for multiple IR methods, including Boolean retrieval, vector space, probabilistic and LSI, as well as variations;
3. Including documentation and examples to enable interested persons to perform experiments, extend the software, or incorporate it with their own tools;
4. Suitable for medium (10GB to 100GB) to large (up to 1000GB) collections of documents; and
5. With a focus on semi-structured documents, including HTML and XML formats, but also compatible with plain text.

Software for IR experimentation that has seen great success includes SMART (Buckley & Walsh, 2001) and Okapi (Robertson & Walker, 2000). However, past versions of such systems have lacked one or more of the qualities above. INQUERY (Allan et al., 2001), like some other successful systems, is not open source. Free search software such as HT://DIG (http://www.htdig.org) offer high performance and open source, but are not readily adaptable for retrieval research.

In contrast to the systems that regularly appear at TREC conferences and other venues, some of the most successful and widely used systems for IR – Web search engines – are prohibitive of most forms of experimental IR research. Despite starting as open or publicly funded projects, popular Web search engines including Lycos, Yahoo and Google do not make their software or algorithms publicly accessible, and there are few opportunities for the utilization of their techniques for TREC-style experimental research.

The above is not intended as criticism of the software or the people behind it. In fact, the list above is indicative of the great success that IR has had in bringing people closer to the information they seek. Nevertheless, there is certainly room for at least one project with the goals above.

The software, which is called the Information Retrieval Toolkit (IRTools), is not intended as a panacea, nor does it propose to supplant existing systems. Instead, as the name implies, it is intended as one possible addition to the modern experimental IR researcher's collection of software and algorithms. The source code for IRTools is available at http://sourceforge.net/projects/irtools .

Web Track Results

Development of IRTools has been steady but slow. File structures, data structures and algorithms have been under constant development and reassessment, and it seems that at any time only part of the software works. To benchmark the software, we wanted to submit runs with fairly standard and well-known approaches. The VSM with Lnu.Ltc weighting was utilized for TREC10. For the pivoted document weights, a constant of 0.25 was chosen based on experiments with TREC9 qrels.

Two runs were submitted, irtLnua and irtLnut. IrtLnua included all non-stopworded terms, and resulted in abysmal results with average precision well under 1%. These results are worse than might be expected if randomly retrieved documents were submitted. There appear to be one or more bugs in the Boolean recombination or term weighting subsystem that resulted in documents with low-value terms being ranked highly. These are disappointing results, but appear to be the outcome of one or more bugs.

IrtLnut is better, though not as good as we expected. As a benchmark run, we anticipated performance similar to our work with post-hoc evaluation of TREC9 runs, in which we typically gained average precision of .25 or so.

For this run, only terms from the <TITLE> section of each topic were used, minus terms on the 622-term stop list (similar to the SMART list). Results are presented in Table 1.

Table 1: irtLnut (judged run) Result Summary

IrtLnut Overall statistics	
Retrieved	46432
Relevant	3363
Rel_ret	838
Exact:	0.0321

Relevant at 1000 docs:	
Runs >= Median	2
Runs < Median	48
Runs with 0 relevant docs	16

Average Precision:	
Runs >= Median	3
Runs < Median	47
Runs with 0 ave_p	22

Generally, topics which other systems found "easy" (in terms of a high median relevant documents at 1000) were found easy in the irtLnut run. Such topics included 509, 513, 527, 530, 544 and 547.

Anomalous topics, in which irtLnut was very low but the median relevant documents found across all participants was high, were 511, 519, 541 and 549. These topics appear to be victims of the same bug that impacted irtLnua – unimportant terms (such as "info" in topic 519) were given higher weights than important terms (such as "frogs").

The best runs for irtLnut included 509 ("steroids what does it do to your body"), 517 ("titanic what went wrong"), 527 ("can you find info on booker t Washington") and 544 ("estrogen why needed"). In all cases, our suspicion is that the initial pre-weighting Boolean set of documents was of sufficiently high quality to offset bugs in term weights and ranking.

Interactive Track Plans

Our work on the interactive track is ongoing. The plan for the study is to test for differences in search results and performance between two versions of the results display interface. The control interface will display results in a traditional list format, whereas the experimental interface will display results in a browseable category hierarchy, based on the Yahoo categories.

Our intent is to produce testable hypotheses about the presentation of a fixed number of results in text and several non-text formats.

There will be 24 participants in the study. Each participant will do two searches on the control system (one fully specified, one partially specified) and two searches on the experimental system (one fully specified, one partially specified). The four topics are distributed evenly across the participants so that each participant is dealing with tasks from only two of the four topics, one partially specified and one fully specified task from each topic.

Searches will be run on Google against the live Web as indexed there. Mapping resulting hits to the Yahoo categories will occur via a proxy server on our local system, using a combination of standard vector space algorithms and some categorization algorithms to address granularity problems (e.g., to make sure we don't present dozens of low-level categories that all share higher-level categories).

As the participants are searching, we will automatically record the URL of each document they view via the proxy server. We will also ask the participants to record the URL(s) of document(s) they believe satisfy the requirements of each task. In addition, we will record the total amount of time each participant spends completing each task.

Each participant will complete a pre-search questionnaire that asks for basic demographic information as well as prior experience with web searching in general and web searching particularly related to the two domains (medical and travel) and two actions (buying online and researching a topic for a project) specified in the tasks. Participants will be given a post-search questionnaire to evaluate each system and express what they like or dislike about each.

Conclusion

The Web track results support our plan to first implement relatively well-known IR techniques in IRTools, in order to gain confidence in our system's performance. As has happened in prior years to other TREC participants, last-minute bugs appear to have thwarted our efforts at reasonable benchmarks.

Integrating well-known techniques into an integrated software package has proven to be challenging. File structures have been particularly problematic, as different data points are required for different IR schemes, yet we desire to minimize disk I/O while having generalized data structures stored to disk. Scaling for sparse-matrix techniques (LSI) as well as dense-matrix techniques (information space) has also been challenging.

Despite these challenges, we anticipate success in achieving our goals for IRTools. We speculate being able to approximate results from best-of-breed IR systems within IRTools, enabling controlled experiments comparing the impact of different manipulations. We hope that these efforts, combined with the work of other research groups, will improve retrieval and scaling for semi-structured textual data.

References

Allan, J.; Connell, M.E.; Croft, W.B.; Feng, F.; Fisher, D. & Li, X. 2001. "INQUERY at TREC9." In Voorhees, Ellen and Harman, Donna (Eds.). The Ninth Text REtrieval Conference (TREC-9). Gaithersburg, MD: National Institute of Standards and Technology. Special Publication 500-249.

Buckley, Chris & Walsh, Janet. 2001. "SabIR Research at TREC-9." In Voorhees, Ellen and Harman, Donna (Eds.). The Ninth Text REtrieval Conference (TREC-9). Gaithersburg, MD: National Institute of Standards and Technology. Special Publication 500-249.

Newby, Gregory B. 2001. "Information Space Based on HTML Structure." In Voorhees, Ellen and Harman, Donna (Eds.). The Ninth Text REtrieval Conference (TREC-

9). Gaithersburg, MD: National Institute of Standards and Technology. Special Publication 500-249.

Rehder, B.; Landauer, T.K; Littman, M.; Dumais, S. 1998. "Automatic 3-Language Cross-Language Information Retrieval with Latent Semantic Indexing." In Voorhees, Ellen and Harman, Donna (Eds.). The Sixth Text REtrieval Conference (TREC-6). Gaithersburg, MD: National Institute of Standards and Technology. Special Publication 500-240.

Robertson, S.E. & Walker, S. 2000. "Okapi/Keenbow at TREC-8." In Voorhees, Ellen and Harman, Donna (Eds.). The Eighth Text REtrieval Conference (TREC-8). Gaithersburg, MD: National Institute of Standards and Technology. Special Publication 500-246.

Acknowledgements

This work was supported in part by the National Science Foundation (NSF award #CCR-0082655).

Many students have worked on the software for this year's results. Their efforts are significant and valued. Some of these students include: Andre Burton, Sheila Denn, Miles Efron, Wei Gao, Xiaosi Li, Monique Lowe, Carolyn Mitchell, Corey Nickens, Zach Sharek, Chang Su, and Yuehong Wang and Jewal Ward. The author, as project director, takes all the blame for bugs and other errors.

Combining text- and link-based retrieval methods for Web IR

Kiduk Yang
School of Information and Library Science
University of North Carolina
Chapel Hill, North Carolina 27599-3360, U.S.A.
yangk@ils.unc.edu

0. Submitted Runs

uncvmss, uncvsmm, uncfsls, uncfslm[1] – WT10g automatic topic relevance task runs

1. Introduction

The characteristics of Web search environment, namely the document characteristics and the searcher behavior on the Web, confound the problems of Information Retrieval (IR). The massive, heterogeneous, dynamic, and distributed Web document collection as well as the unpredictable and less than ideal querying behavior of a typical Web searcher exacerbate conventional IR problems and diminish the effectiveness of retrieval approaches proven in the laboratory conditions of traditional IR. At the same time, the Web is rich with various sources of information that go beyond the contents of documents, such as document characteristics, hyperlinks, Web directories (e.g. Yahoo), and user statistics.

Fusion IR studies have repeatedly shown that combining multiple sources of evidence can improve retrieval performance. Furthermore, the nature of the Web search environment is such that retrieval approaches based on single sources of evidence suffer from weaknesses that can hurt the retrieval performance in certain situations. For example, content-based IR approaches have difficulty dealing with the diversity in vocabulary and quality of web documents, while link-based approaches can suffer from incomplete or noisy link topology. The inadequacies of singular Web IR approaches coupled with the fusion hypothesis (i.e. "fusion is good for IR") make a strong argument for combining multiple sources of evidence as a potentially advantageous retrieval strategy for Web IR.

Among the various source of evidence on the Web, we focused our TREC-10 efforts on leveraging document text and hyperlinks, and examined the effects of combining result sets as well as those of various evidence source parameters.

2. Text-based Method: VSM

The text-based retrieval component of the experiment was based on a Vector Space Model (VSM) using the SMART length-normalized term weights as implemented in IRIS[2] (Yang & Maglaughlin, 2000).

2.1 Text Processing

IRIS processed documents by first removing HTML tags and punctuation, and then excluding

[1] Submitted runs along with the entire retrieval system were lost due to a machine crash. Results discussed in this paper are based on post-submission data produced by a recreated system. At the time of submission, uncvsms and uncvsmm were vector space model (VSM) runs using short and medium length queries respectively, and uncfsls and uncfslm were VSM and HITS fusion runs using short and medium queries.

[2] IRIS (Interactive Retrieval Information System) is an experimental retrieval system developed in the School of Information and Library Science at the University of North Carolina.

390 high-frequency terms listed in the WAIS default stopwords list as well as "IRIS stopwords,[3]" which were arrived at by examining the inverted index and identifying low frequency terms that appeared to have little value.

After punctuation and stopword removal, IRIS conflated each word by applying the simple plural remover (Frakes & Baeza-Yates, 1992). The simple plural remover was chosen to speed up indexing time and to minimize the overstemming effect of more aggressive stemmers.

2.2 Term Indexes

In addition to body text terms (i.e. terms between <BODY> and </BODY> tags), IRIS extracted header text terms from document titles, meta keyword and description texts, and heading texts (i.e. texts between <Hn> and </Hn> tags). A combination of body and header text terms was also created, where the header text terms were emphasized by multiplying the term frequencies by 10.

In each of the three term sources, adjacent noun phrases were identified to construct noun phrase indexes as well as single term indexes. By using an online dictionary and the punctuation-based phrase window recognition algorithm, IRIS defined an adjacent noun phrase as consisting of up to three adjacent dictionary nouns or capitalized words within a phrase window.

2.3 Document Ranking and Pseudo-feedback

Documents were ranked in decreasing order of the inner product of document and query vectors,

$$\mathrm{q}^{\mathrm{T}}\mathrm{d}_i = \sum_{k=1}^{t} q_k d_{ik}, \qquad (1)$$

where q_k is the weight of term k in the query, d_{ik} is the weight of term k in document i, and t is the number of terms in the index. SMART *Lnu* weights with the slope of 0.3 were used for document terms (Buckley et al., 1996; Buckley et al., 1997), and SMART *ltc* weights (Buckley et al., 1995) were used for query terms. *Lnu* weights attempt to match the probability of retrieval given a document length with the probability of relevance given that length (Singhal et al., 1996).

Top ten positive and top two negative weighted terms from the top three ranked documents of the initial retrieval results were used to expand the initial query in a pseudo-feedback retrieval process.

2.4 VSM systems

Table 1 enumerates the text-based method parameters for VSM systems, which are query length, term source, use of phrase terms, and use of pseudo-feedback. Query length range from short (topic title) and medium (topic title and description) to long (topic title, description, and narrative). Term sources are body text, header text, and body plus header text. The combination of parameters (3 query lengths, 3 term sources, 2 for phrase use, 2 for feedback use) resulted in 36 VSM systems.

[3] IRIS stopwords for TREC-10 Web track experiment were defined as all non-alphabetical words (exception: embedded hyphen), words consisting of more than 25 or less than 3 characters, and words that contain 3 or more repeated characters.

Table 1. VSM system* parameters

Query length	*Term Source*	*Noun Phrase*	*Pseudo-feedback*
short	body text	no	no
medium	header text	yes	yes
long	body + header		

*VSM system name = vsm$qform$index$phrase.$feedback (e.g. vsmsb0.1)
where $qform = query length (s, m, l)
$index = term source (b, h, bh)
$phrase = noun phrase (0, 1)
$feedback = pseudo-feedback (1, 2)

3. Link-based Method: HITS

Among the several possible link-based retrieval methods, the authority scores of documents computed by the HITS algorithm (Kleinberg, 1997) were used to generate a ranked list of documents with respect to a given query. PageRank scores (Page et al., 1998) could be used to rank documents as well, but effectiveness computation of PageRank scores is likely to require a much larger set of linked documents than WT10g corpus (Brin & Page, 1998). The Clever algorithm that extends HITS by incorporating the text around links into the computation of hub and authority scores has been shown to improve the performance of HITS (Chakrabarti et al., 1998). However, Clever combines link- and text-based methods implicitly and thus makes it difficult to isolate the contributions and behaviors of individual methods, which we wanted to study to better understand the effect of combining retrieval result sets.

HITS defines "authority" as a page that is pointed to by many good hubs and defines "hub" as a page that points to many good authorities. Mathematically, these circular definitions can be expressed as follows:

$$a(p) = \sum_{q \to p} h(q), \qquad (2)$$

$$h(p) = \sum_{p \to q} a(q). \qquad (3)$$

The above equations define the authority weight *a(p)* and the hub weight *h(p)* for each page *p*, where $p \to q$ denote "page *p* has a hyperlink to page *q*".

HITS starts with a root set *S* of text-based search engine results in response to a query about some topic, expands *S* to a base set *T* with the inlinks and outlinks of *S*, eliminates links between pages with the same domain name in *T* to define the graph *G*, runs the iterative algorithm (equations 2 and 3) on *G* until convergence, and returns a set of documents with high *h(p)* weights (i.e. hubs) and another set with high *a(p)* weights (i.e. authorities). The iterative algorithm works as follows: Starting with all weights initialized to 1, each step of the iterative algorithm computes *h(p)* and *a(p)* for every page *p* in *T*, normalizes each of them so that the sum of the squares adds up to 1, and repeats until the weights stabilize. In fact it can be shown that the authority weights at convergence correspond to the principal eigenvalues of A^TA and hub weights correspond to those of AA^T, where A is the link matrix of the base set *T*[4]. Typically, convergence

[4] The (*i,j*)th entry of A is 1 if there exists a link from page *i* to page *j*, and is 0 otherwise. In A^T, the transpose of the link matrix A, the (*i,j*)th entry of A corresponds to the link from page *j* to page *i*. The (*i,j*)th entry of AA^T gives the number of pages pointed to by both page *i* and page *j* (bibliometric coupling), while the (*i,j*)th entry of A^TA gives the number of pages that point to both page *i* and page *j* (cocitation).

occurs in 10 to 50 iterations for *T* consisting of about 5000 Web pages, expanded from the root set *S* of 200 pages while being constrained by the expansion limit of 50 inlinks per page.

3.1 Modified HITS Algorithm

The original HITS algorithm was modified by adopting a couple of improvements from other HITS-based approaches. As implemented in the ARC algorithm (Chakrabarti et al., 1998), the root set was expanded by 2 links instead of 1 link (i.e. expand *S* by all pages that are 2 link distance away from *S*). Also, the edge weights by Bharat and Henzinger (1998), which essentially normalize the contribution of authorship by dividing the contribution of each page by the number of pages created by the same author, was used to modify the HITS formulas as follows:

$$a(p) = \sum_{q \to p} h(q) \times auth\_wt(q,p), \qquad (4)$$

$$h(p) = \sum_{p \to q} a(q) \times hub\_wt(p,q). \qquad (5)$$

In above equations, *auth_wt(q,p)* is 1/*m* for page *q*, whose host has *m* documents pointing to *p*, and *hub_wt(p,q)* is 1/*n* for page *q*, which is pointed by *n* documents from the host of *p*.

3.2 Host Definitions

To compute the edge weights of modified HITS algorithm, one must first establish a definition of a host to identify the page authorship (i.e. documents belonging to a given host are created by the same author). Though host identification heuristics employing link analysis might be ideal, we opted for simplistic host definitions based on URL lengths. Short host form was arrived at by truncating the document URL at the first occurrence of a slash mark (i.e. '/'), and long host form from the last occurrence.

3.3 HITS systems

Among the 36 text-based system results, we chose the best performing system with all variations of query lengths. The combination of host definition and seed set parameters, as seen in Table 2 below, resulted in 6 HITS systems.

Table 2. HITS system* parameters

Host Definition	*Seed Set*
short	short query, body text, phrase, no feedback
long	medium query, body text, phrase, no feedback
	long query, body text, phrase, no feedback

*HITS system name = hit$hform$seed (e.g. hitssb1.1)
where $hform = host definition (s, l)
$seed = seed set (sb1.1, mb1.1, lb1.1)

4. Fusion Method

Since it is not clear from literature how much can be gained by using one fusion method over another, the Similarity Merge method (Fox & Shaw, 1993, 1994) was chosen for its simplicity and consideration of overlap, which is thought to be an important factor in fusion. Equation (6) below describes the fusion formula used to merge and rank documents retrieved by different systems:

$$FS = (\sum NS_i)*\text{overlap}, \quad (6)$$

where: FS = fusion score of a document,
NS_i = normalized score of a document by method i,
overlap = number of methods that retrieved a given document.

The normalized document score, NS_i, is computed by Lee's min-max formula (1996, 1997), where S_i is the retrieval score of a given document and S_{max} and S_{min} are the maximum and minimum document scores by method i.

$$NS_i = (S_i - S_{min}) / (S_{max} - S_{min}), \quad (7)$$

5. Results

Although various fusion combinations were tried, combining retrieval result sets did not improve on the performance of the best text-based method. In fact, fusion in general seemed to decrease retrieval performance, which is contrary to previous fusion research findings that suggest that combining results of various retrieval methods is beneficial to retrieval performance.

Curiously enough, past TREC participants who tried fusion with WT10g corpus also found that combining text- and link-based methods did not improve retrieval performance (Singhal & Kaszkiel, 2001; Gurrin & Smeaton, 2001; Savoy & Rasolofo, 2001). Whether this is simply an artifact of the WT10g test collection (i.e. link structure, relevance judgments, query characteristics) or the reflection of real inadequacies present in link analysis and/or fusion methods remains the main focus in our ongoing investigation.

5.1 Single System Results

The best performing VSM system, measured by average precision of 0.1406, was vsmlb1.1 (long query, body text, noun phrase, no feedback). The best HITS system was hitslb1.1 (short host, seed set system of vsmlb1.1) with average precision of 0.0399. The best text-based system not only outperformed the best link-based system (3.5 times better in average precision), but also outperformed all other systems, both single and fusion, as can be seen in subsequent sections.

Examination of single system results (Table 3) reveals some interesting phenomena regarding the effects of individual system parameters on retrieval performance. According to Table 3, the system parameters most influential to retrieval performance seem to be index source, query length, and host definition. VSM systems using header terms only show markedly worse performance than systems using body text terms, and longer query length systems generally perform better than shorter query systems using the same index source terms. The shorter host definition is obviously far superior to longer definition (over 13 times better in average precision) for HITS systems.

In post analysis, we constructed optimum seed sets from known relevant documents to ascertain the maximum performance level possible by HITS method for WT10g corpus. Although the HITS system with optimum seed set and short host definition resulted in an average precision value eight times that of the best HITS system (0.3144 vs. 0.0399), it is somewhat disappointing as a maximum performance threshold. One could even view this as the failing of HITS algorithm, which reduces the seed system performance by one third at best.

Table 3. Single System Results

VSM systems	*Average Precision*	*HITS systems*	*Average Precision*
vsmlb1.1	0.1406	hitsopt[1]	0.3144
vsmlb0.1	0.1387	hitlopt[2]	0.0447
vsmlb0.2	0.1345	hitslb1.1	0.0399
vsmlb1.2	0.1339	hitsmb1.1	0.0382
vsmmb1.1	0.1272	hitssb1.1	0.0314
vsmmb1.2	0.1254	hitllb1.1	0.0029
vsmmb0.1	0.1247	hitlmb1.1	0.0026
vsmmb0.2	0.1233	hitlsb1.1	0.0008
vsmlbh1.1	0.1148		
vsmlbh0.1	0.1114		
vsmlbh0.2	0.1103		
vsmlbh1.2	0.1079		
vsmsb1.1	0.1054		
vsmsb0.1	0.1038		
vsmsb1.2	0.1036		
vsmsb0.2	0.1032		
vsmmbh1.1	0.1017		
vsmmbh0.1	0.0998		
vsmmbh1.2	0.0988		
vsmmbh0.2	0.0973		
vsmsbh1.1	0.0842		
vsmsbh0.1	0.0830		
vsmsbh1.2	0.0819		
vsmsbh0.2	0.0815		
vsmmh0.1	0.0210		
vsmmh1.1	0.0208		
vsmlh0.2	0.0190		
vsmlh0.1	0.0182		
vsmmh0.2	0.0182		
vsmsh1.1	0.0181		
vsmsh0.1	0.0179		
vsmlh1.2	0.0176		
vsmlh1.1	0.0172		
vsmmh1.2	0.0163		
vsmsh0.2	0.0151		
vsmsh1.2	0.0133		

hitsoptimum[1] = short host, optimum seed set
hitloptimum[2] = long host, optimum seed set

The performance of the optimum HITS system in Table 3 may not necessarily reflect the true potential of link analysis approach. In addition to potential effects of incomplete relevance judgments and truncated link structure with heavy concentration of spurious links in WT10g collection (Gurrin & Smeaton, 2001), we note that 42 out of 50 TREC-10 topics have less than 100 known relevant documents. In fact, 31 of those 42 topics have less than 50 known relevant documents. The topics with small number of relevant documents mean noisy seed sets, even when the perfect results have been achieved by a seed retrieval system (i.e. over three quarters of the seed set of size 200 will consist of irrelevant documents for 31 topics), which are likely to bring in more noise during link expansion and thus result in expanded sets with dominant link structures unrelated to the original topics.

Another point to consider about the HITS method is its tendency to rank documents in relatively small clusters, where each cluster represents mutually reinforcing communities (i.e. hubs and authorities) on sufficiently broad topics. This tendency could rank clusters of non-relevant documents with dense link structure above sparsely linked relevant documents, which will adversely affect average precision but may not affect high precision.

5.2 Fusion System Results

Table 4 and 5 show the fusion performances of combining various VSM and HITS system results. It is interesting to note that the best VSM fusion result (0.1354 in Table 4) is worse than the best VSM single system result (0.1406 in Table 3), while the best HITS fusion result (0.0540 in Table 5) is better than the best HITS single system result (0.0399 in Table 3). One possible explanation for this phenomenon may be that the best VSM system dominates all other systems (i.e. additional relevant documents introduced by other system are negligible), while the best HITS system result is enhance by unique contributions from other HITS systems. In other words, HITS systems may produce more diverse result sets than VMS systems and are thus helped by fusion.

Combining text- and link-based systems (Table 6) resulted in performance degradation of text-based results, even when the best HITS and VSM systems were combined (0.1012 in Table 6 vs. 0.1406 in Table 3). When the optimum HITS result was combined with the best VSM result (0.3144 and 0.1406 in Table 3), however, the improvement by fusion was almost linear (0.4549). Although such fusion system is unrealistic, it does suggest the fusion potential where optimum performance level of one method can be raised by combining it with a reasonably effective method of a different kind.

Table 4. VSM fusion systems

Systems	*Query Length*	*Term Index*	*Pseudo-feedback*	*Average Precision*
fvsmb0.1	all	body text, no phrase	no	0.1331
fvsmb0.2	all	body text, no phrase	yes	0.1297
fvsmb1.1	all	body text, phrase	no	0.1354
fvsmb1.2	all	body text, phrase	yes	0.1309
fvsmh0.1	all	header text, no phrase	no	0.0193
fvsmh0.2	all	header text, no phrase	yes	0.0176
fvsmh1.1	all	header text, phrase	no	0.0196
fvsmh1.2	all	header text, phrase	yes	0.0166
fvsmbh0.1	all	body+header, no phrase	no	0.1046
fvsmbh0.2	all	body+header, no phrase	yes	0.1017
fvsmbh1.1	all	body+header, phrase	no	0.1074
fvsmbh1.2	all	body+header, phrase	yes	0.1039
fvsms.1	short	all	no	0.0729
fvsms.2	short	all	yes	0.0697
fvsmm.1	medium	all	no	0.0886
fvsmm.2	medium	all	yes	0.0840
fvsml.1	long	all	no	0.1055
fvsml.2	long	all	yes	0.0979
fvsm.1	all	all	no	0.0956
fvsm.2	all	all	yes	0.0920
fvsm	all	all	all	0.0947

Table 5. HITS fusion systems

Systems	*Host Definition*	*Seed Set*	*Average Precision*
fhitsb11	all	vsmsb1.1	0.0231
fhitmb11	all	vsmmb1.1	0.0303
fhitlb11	all	vsmlb1.1	0.0304
fhits	short	all	0.0540
fhitl	long	all	0.0032
fhit	all	all	0.0407

Table 6. HITS + VSM fusion systems

System Name	HITS	VSM	Average Precision
fhsopt	optimal system (hitsopt)	best system (vsmlb1.1)	0.4549
fhsbest	best system (hitslb1.1)	best system (vsmlb1.1)	0.1012
fhsv.1	all with short host	all with no feedback	0.1017
fhsv.2	all with short host	all with feedback	0.1019
fhlv.1	all with long host	all with no feedback	0.0999
fhlv.2	all with long host	all with feedback	0.1017
fhv.1	all	all with no feedback	0.0999
fhv.2	all	all with feedback	0.1017
fhvs.1	all	all with short query, no feedback	0.0782
fhvs.2	all	all with short query, feedback	0.0963
fhvm.1	all	all with medium query, no feedback	0.0879
fhvm.2	all	all with medium query, feedback	0.0980
fhvl.1	all	all with long query, no feedback	0.0999
fhvl.2	all	all with long query, feedback	0.0999
fhv.1	all	all without feedback	0.1018
fhv.2	all	all with feedback	0.1018
fhv	all	all	0.1018

6. Conclusion

In WT10g topic relevance task, we examined the effect of combining result sets as well as those of various evidence source parameters for text- and link-based methods. Analysis of results suggests that index source, query length, and host definition are the most influential system parameters for retrieval performance. We found link-based systems, HITS in particular, to perform significantly worse than text-bases systems, and combining results sets using the similarity merge formula did not enhance retrieval performance in general. Performance improvement by fusion occurred only on two occasions: once when HITS systems with short host definition were combined, and another time when the optimum HITS result was combined with the best VSM result.

The general failure of fusion evidenced in our results could be due to the characteristics of WT10g test collection, failings of link analysis, inadequacies of fusion formula, or combinations of all or any of the above. The optimum fusion combination result suggests to us that fusion potential exists despite possible shortcomings of the test collection and individual retrieval methods. Consequently, we believe the future fusion efforts should focus on discovering the fusion formula that can best realize the fusion potential of combining diverse retrieval methods.

References

Bharat, K. & Henzinger, M. R. (1998). Improved Algorithms for Topic Distillation in Hyperlinked Environments. *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 104-111.

Brin, S. & Page, L. (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Proceedings of the 7th International World Wide Web Conference*, 107-117.

Buckley, C., Salton, G., & Allan, J., & Singhal, A. (1995). Automatic query expansion using SMART: TREC 3. In D. K. Harman (Ed.), *The Third Text Rerieval Conference (TREC-3)* (NIST Spec. Publ. 500-225, pp. 1-19). Washington, DC: U.S. Government Printing Office.

Buckley, C., Singhal, A., & Mitra, M. (1997). Using query zoning and correlation within SMART: TREC 5. In E. M. Voorhees & D. K. Harman (Eds.), *The Fifth Text REtrieval Conference (TREC-5)* (NIST Spec. Publ. 500-238, pp. 105-118). Washington, DC: U.S. Government Printing Office.

Buckley, C., Singhal, A., Mitra, M., & Salton, G. (1996). New retrieval approaches using SMART: TREC 4. In D. K. Harman (Ed.), *The Fourth Text REtrieval Conference (TREC-4)* (NIST Spec. Publ. 500-236, pp. 25-48). Washington, DC: U.S. Government Printing Office.

Chakrabarti, S., Dom, B., Raghavan, P., Rajagopalan, S., Gibson, D., & Kleinberg, J. (1998b). Automatic resource list compilation by analyzing hyperlink structure and associated text. *Proceedings of the 7th International World Wide Web Conference.*

Fox, E. A., & Shaw, J. A. (1994). Combination of multiple searches. In D. K. Harman (Ed.), The Second Text Rerieval Conference (TREC-2) (NIST Spec. Publ. 500-215, pp. 243-252). Washington, DC: U.S. Government Printing Office.

Fox, E. A., & Shaw, J. A. (1995). Combination of multiple searches. In D. K. Harman (Ed.), *The Third Text Rerieval Conference (TREC-3)* (NIST Spec. Publ. 500-225, pp. 105-108). Washington, DC: U.S. Government Printing Office.

Frakes, W. B., & Baeza-Yates, R. (Eds.). (1992). *Information retrieval: Data structures & algorithms*. Englewood Cliffs, NJ: Prentice Hall.

Gurrin, C. & Smeaton, A.F. (2001). Dublin City University experiments in connectivity analysis for TREC-9. In E. M. Voorhees & D. K. Harman (Eds.), *The Nineth Text Rerieval Conference (TREC-9)*. Washington, DC: U.S. Government Printing Office.

Kleinberg, J. (1997). Authoritative sources in a hyperlinked environment. *Proceeding of the 9th ACM-SIAM Symposium on Discrete Algorithms.*

Lee, J. H. (1996). *Combining multiple evidence from different relevance feedback methods (Tech. Rep. No. IR-87).* Amherst: University of Massachusetts, Center for Intelligent Information Retrieval.

Lee, J. H. (1997). Analyses of multiple evidence combination. Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval, 267-276.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1998). The PageRank citation ranking: Bringing order to the Web. *Unpublished*

Savoy, J. & Rasolofo, Y. (2001). Report on the TREC-9 experiment: Link-based retrieval and distributed collections. In E. M. Voorhees & D. K. Harman (Eds.), *The Nineth Text Rerieval Conference (TREC-9).* Washington, DC: U.S. Government Printing Office.

Singhal, A., Buckley, C., & Mitra, M. (1996). Pivoted document length normalization. *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 21-29.

Singhal, A. & Kaszkiel, M. (2001). AT&T Labs at TREC-9. In E. M. Voorhees & D. K. Harman (Eds.), *The Nineth Text Rerieval Conference (TREC-9).* Washington, DC: U.S. Government Printing Office.

Yang, K. & Maglaughlin, K. (2000). *IRIS at TREC*-8. In E. M. Voorhees & D. K. Harman (Eds.), *The Eighth Text Rerieval Conference (TREC-8).* Washington, DC: U.S. Government Printing Office.

UNT TRECvid: A Brighton Image Searcher Application [1]

Mark Rorvig [2], Ki-Tai Jeong, Anup Pachlag,
Ramprasad Anusuri, Diane Jenkins, Sara Oyarce
University of North Texas
Texas Center for Digital Knowledge

Abstract

The results, architecture and processing steps used by the University of North Texas team in the 2001 TRECvid video retrieval trials are described. Only a limited number of questions were selected by the team due to resource limitations described in the paper. However, the average precision of the team results over thirteen questions from the General search category were reasonable at 0.59.

1 Background

The Brighton Image Searcher was constructed following the 3rd International Conference on the Challenge of Image Retrieval sponsored by the Institute for Image Data Research of the University of Northumbria in Brighton, United Kingdom (Rorvig et al., 2000; Goodrum et al, 2001). Details of the system architecture and performance are documented in two papers presented in 2001 at the 10th World Wide Web Conference in Hong Kong, Special Administrative Region (Jeong et al., 2001). The system may be used at <http://archive4.lis.unt.edu/tdt/www/> for a collection of still images from the NASA Hubble Space Telescope Repair Mission.

This system is intended for use in the retrieval of still images. However, it was adapted for the TRECvid task by sampling salient, or key frames from the video segments comprising the NIST TRECvid test collection. The key frame extraction algorithm was developed at NASA and is described in Rorvig (1993). This algorithm is presently used to summarize International Space Station video downlink at the Johnson Space Center in Houston, Texas, USA.

2 Procedures

TRECvid Test Collection files were processed at equal intervals of five seconds to extract key frame candidates. The key frame candidates were then processed to extract a number of measures corresponding to primitive image features first proposed in 1980 by Marr in his posthumously published work *Vision*. These measures are typically rendered as histograms but are treated in our system as Lorenz Information Measures This technique is more fully described in Jeong, et al. (2001). The image processing was performed using the relatively new Java Image Processing Libraries.

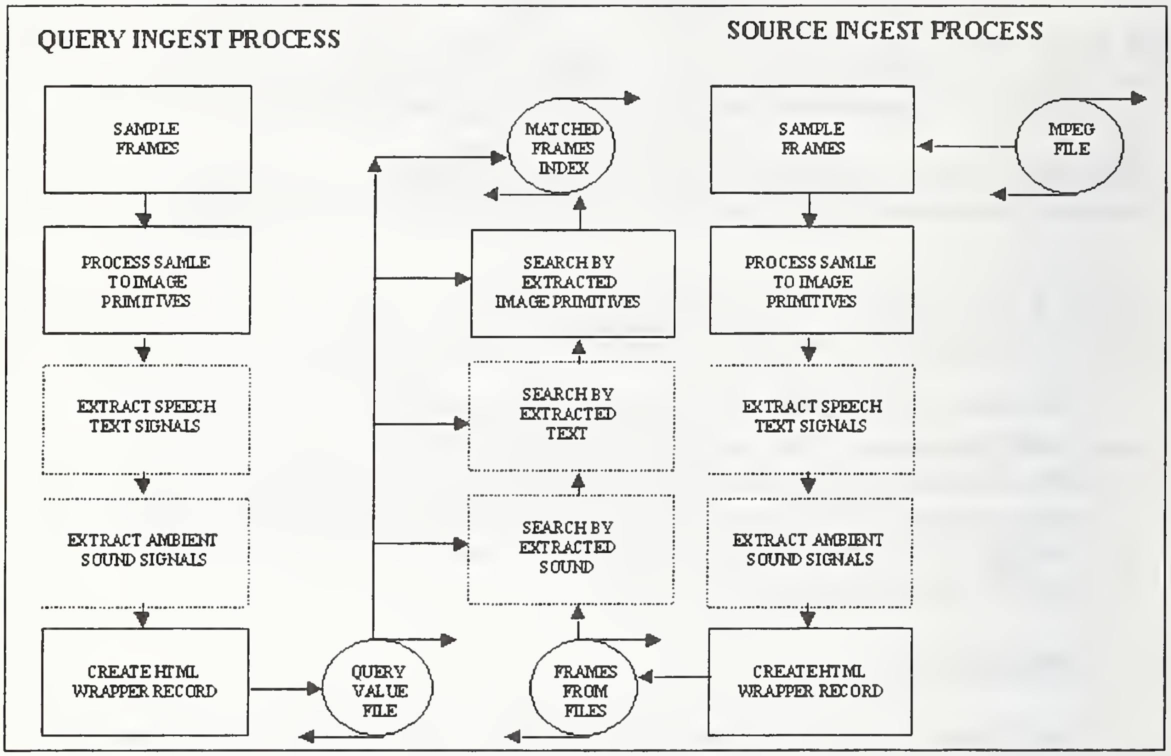
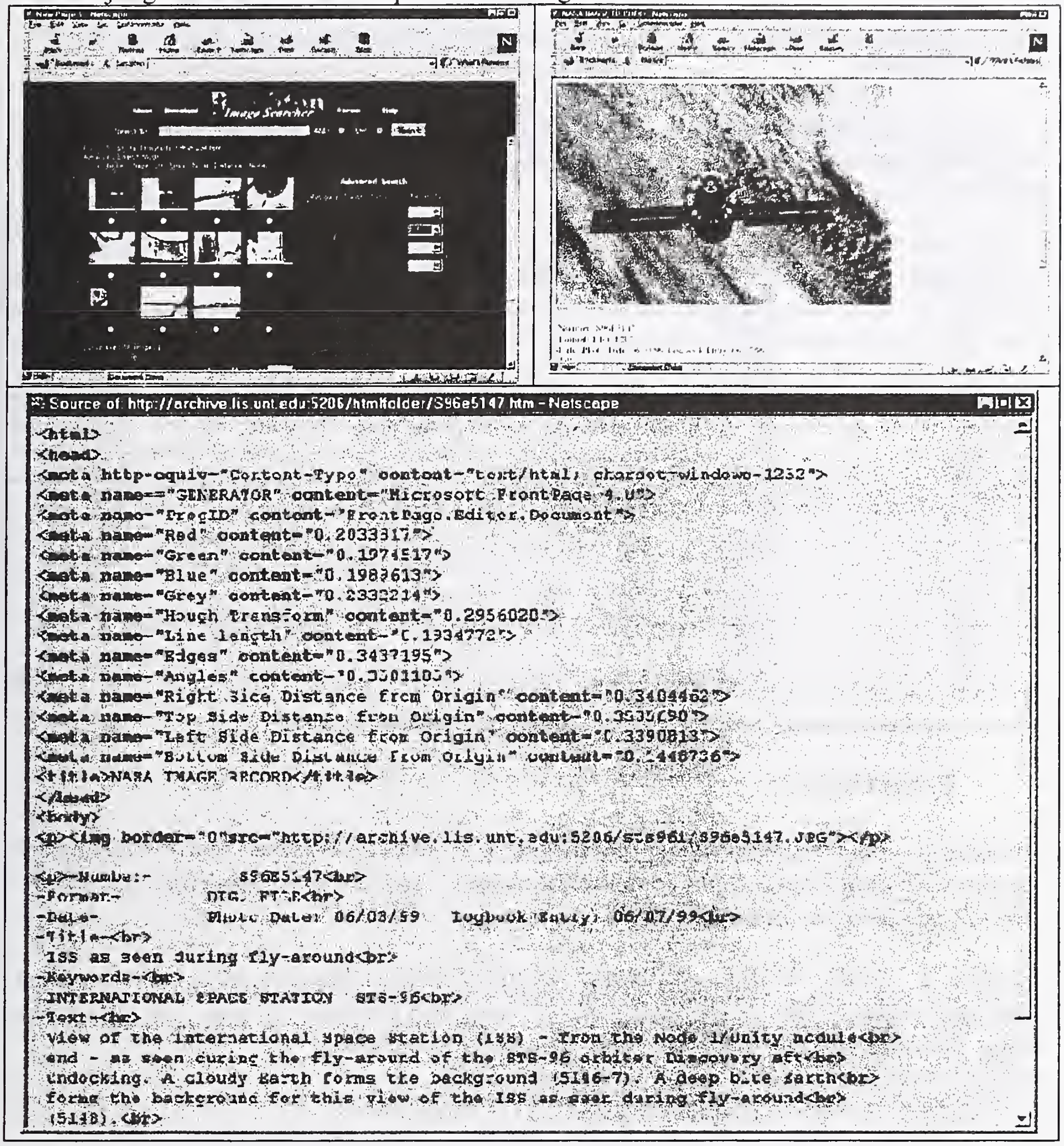

QUERY INGEST PROCESS
SOURCE INGEST PROCESS
SAMPLE FRAMES
PROCESS SAMLE TO IMAGE PRIMITIVES
EXTRACT SPEECH TEXT SIGNALS
EXTRACT AMBIENT SOUND SIGNALS
CREATE HTML WRAPPER RECORD
QUERY VALUE FILE
MATCHED FRAMES INDEX
SEARCH BY EXTRACTED IMAGE PRIMITIVES
SEARCH BY EXTRACTED TEXT
SEARCH BY EXTRACTED SOUND
FRAMES FROM FILES
SAMPLE FRAMES
MPEG FILE
PROCESS SAMLE TO IMAGE PRIMITIVES
EXTRACT SPEECH TEXT SIGNALS
EXTRACT AMBIENT SOUND SIGNALS
CREATE HTML WRAPPER RECORD

Illustration 1: System layout of the components of the Brighton Image Searcher adapted for use for the TRECvid trials. The dashed squares represent system functionality under construction and unavailable at the time of the TRECvid submission.

In future versions of this system, a query preprocessor will be built using blocks of this software to extract query representations from moving image clips. The extracted key frame measurements would then be matched by city block metric to key frames in the batch extracted collection. For this trial, however, only those test questions that provided either a known item or a search example that had already been processed in the key frame selection procedure were addressed. Further, use of the test question set even from this subset was limited to those examples or known item segments for which a key frame had been selected. (Salient or key frames are selected based on the frequency of the appearance of their features in the moving image document, and some examples were simply too short to have produced any frames in the prior batch process.)

Two of us (D.J. and S.O.) performed the searches by selecting a key frame from the collection that appeared in the example or known item intervals and then using that frame as a search exemplar for the Brighton Image Searcher. The key frame selection was based on the searcher's assessment that it was coextensive with the question semantics. In most cases the number of frames available for selection was limited, however, precision scores better than expected may have been due to this introduction of human judgment into the search process lending the final results an "interactive" status.



Source of: http://archive.lis.unt.edu:5206/htmlfolder/S96e5147.htm - Netscape

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<meta name=="GENERATOR" content="Microsoft FrontPage 4.0">
<meta name="ProgID" content="FrontPage.Editor.Document">
<meta name="Red" content="0.2033317">
<meta name="Green" content="0.1974517">
<meta name="Blue" content="0.1989613">
<meta name="Grey" content="0.2332214">
<meta name="Hough Transform" content="0.2956020">
<meta name="Line length" content="0.1934772">
<meta name="Edges" content="0.3437195">
<meta name="Angles" content="0.3301103">
<meta name="Right Side Distance from Origin" content="0.3404462">
<meta name="Top Side Distance from Origin" content="0.3535690">
<meta name="Left Side Distance from Origin" content="0.3390813">
<meta name="Bottom Side Distance from Origin" content="0.2448736">
<title>NASA IMAGE RECORD</title>
</head>
<body>
<p><img border="0"src="http://archive.lis.unt.edu:5206/sts961/S96e5147.JPG"></p>

<p>-Number-          S96E5147<br>
-Format-         DIG. FILE<br>
-Date-           Photo Date: 06/03/99   Logbook Entry: 06/07/99<br>
-Title-<br>
 ISS as seen during fly-around<br>
-Keywords-<br>
 INTERNATIONAL SPACE STATION  STS-96<br>
-Text-<br>
 View of the International Space Station (ISS) - from the Node 1/Unity module<br>
 end - as seen during the fly-around of the STS-96 orbiter Discovery aft<br>
 undocking. A cloudy Earth forms the background (5146-7). A deep blue Earth<br>
 forms the background for this view of the ISS as seen during fly-around<br>
 (5148).<br>
```

Illustration 2: The three frames above record (a-upper left) a retrieval session page for the image searching system, (b-upper right) a single image chosen by double clicking on an image from (a), and (c-below) the metadata tags for the image represented by the twelve measures used in this system.

In response to an image selected as an exemplar for searching, the Brighton Image Searcher can potentially retrieve all the images in the collection ranked by the order of metric correspondence to the features of the candidate images. However, for this trial, only the images retrieved from the first twelve matches in the system were included in the results.

3 Results

For the thirteen questions addressed by our team, the overall precision score was 0.59 with extremes ranging from 0.92 to 0.00. The table below records these scores from the TRECvid evaluation sponsors.

TRECVID QUESTION	UNT SYSTEM DESIGNATION	RELEVANT SEGMENTS	UNT RLEVANT	UNT SELECTED	PRECISION SCORE
002	Sys21	7	4	9	0.78
024	Sys21	1	1	2	0.50
041	Sys21	9	8	12	0.75
043	Sys21	2	2	5	0.40
048	Sys21	3	0	4	0.75
049	Sys21	3	2	5	0.60
050	Sys21	11	9	12	0.92
051	Sys21	6	5	7	0.86
052	Sys21	0	0	2	0.00
055	Sys21	0	0	1	0.00
056	Sys21	1	1	2	0.50
057	Sys21	1	0	2	0.50
059	Sys21	1	1	4	0.25
061	Sys21	8	7	9	0.89

Table 1: Average precision over 13 questions for the UNT system was 0.59.

4 References

Goodrum, A., Rorvig, M., Jeong, K., Suresh, C. (2001) "An Open Source Agenda for Research Linking Text and Image Content Features," *Journal of the American Society for Information Science*, [in press].

Jeong, K.T., Rorvig, M., Jeon, J., Weng, N. (2001) "Image Retrieval by Content Measure Metadata Coding," *10th International World Wide Web Conference, May 1-5, 2001 Hong Kong, International World Wide Web Conference Committee:*
<http://www.iw3c2.org/Conferences/Welcome.html>,
<http://www.www10.org.hk/cdrom/posters/p1142/index.htm>.

Rorvig, M. E., "A Method for Automatically Abstracting Visual Documents," *Journal of the American Society for Information Science*, 44(1):040-056, 1993.

Rorvig, M., Jeong, K., Suresh, C., Goodrum, A. (2000) "Exploiting Image Primitives for Effective Retrieval," *CIR2000 - Third UK Conference on Image Retrieval, 4 - 5 May 2000, Old Ship Hotel, Brighton, United Kingdom* (J.P. Eakins, P.G.B. Enser, Eds.), [unnumbered manuscript] 7pps., University of Brighton: School of Information Management.

Notes

[1] This work was supported by Intel Corporation, the Special Libraries Association, and the National Aeronautics and Space Administration.

[2] All correspondence should be addressed to Mark Rorvig, Ph.D., UNT/SLIS/TxCDK, P.O. Box 311068, Denton, Texas, USA 76203, Tel: 940-300-5344, Fax: 940-565-3101, <mrorvig@unt.edu>.

University of Padova at TREC-10

Franco Crivellari Massimo Melucci

Università di Padova
Dipartimento di Elettronica e Informatica

October 28, 2001

1 Introduction

This is the second year that the University of Padova participates to the Text Retrieval Conference (TREC). Last year we participated as well to this program of experimentation in information retrieval with very large document databases. In both years, we participated to the Web track, and specifically to the ad-hoc task, which consists in testing retrieval performance by submitting 50 queries extracted from 50 respective topics. This year we participated to the homepage finding task as well. This year we could devote more time to experiments than last year, yet some problems still arose because we indexed the full-text of documents, while we indexed only a portion of documents only.

2 Approach and Experimental Objectives

This year we participated to the ad-hoc and the homepage finding tasks of the Web track. Our objectives were to evaluate:

1. the effectiveness of passage retrieval in Web page retrieval and homepage finding,
2. the effectiveness of combining classic vector space similarity measure and PageRank measure using all links, and
3. the selection of links of some given types in the previous combination.

The baseline was given by document retrieval based on classic vector space similarity, both for the ad-hoc and the homepage finding tasks. The baseline results served as input to combine themselves with link information. Specifically, the runs being reported in Tables 1 and 2 have been performed. To extract text from Web documents, we employed a software agent that follows the Web links to retrieve the Web pages. This robot has been developed within the National

InterData research project [1]. For the purposes of the TREC experiments, a different version of the robot has been designed and developed because the data to be retrieved were locally stored, and not on the Web. Moreover, the data are encoded in SGML also and then the tool has been modified to deal with this additional format. To only extract the tagged text, our robot employed a tool for HTML syntax analysis, called Tidy, that is reported in [2]. Tidy allows for correcting HTML syntax by adding, for example, missing end tags. Documents have been fully indexed, i.e. all the full-text of each document has been processed to extract keywords and the individual positions at which each keyword occur has been recorded. A stoplist including common Web words, such as web, html, http, com, edu has been used to filter function words out. Words have been stemmed using the Porter's algorithm, yet the original word has been recorded as well. At retrieval time, both individual keywords and keyword pairs have been used. All the performed runs employed a variation of the classic $tf \times idf$ weighting scheme, as expressed below:

$$w_{ij} = (1 + \log tf_{ij})\ (\log \frac{N+1}{n_i})$$

where w_{ij} is the weight of keyword i in document or passage j, t_{ij} is the frequency of keyword i in document or passage j, n_i is the number of documents or passages including i, N is the total number of documents or passages.

A few notes about passage retrieval: In PR runs, a list of 10,000 passages were retrieved in response to the query. The retrieved passage list was then transformed into the corresponding 1,000 documents by summing the respective scores. (The passage list was then transformed into the corresponding 100 documents in case of the homepage finding task.) The more the document includes retrieved passages and the higher these passages are ranked, the higher the document is ranked. Passage size was fixed at 100 words. No formatting or logical structure were used because of the nature of data that made hypotheses on the quality of data a very hard task.

As regards the EP task, note that the same algorithms used for the ad-hoc task were employed. Then, the entry point topics were used as usual queries without any sophisticated processing.

PageRank values were computed for every page by considering all the incoming links up to 10 steps and damping factor at 0.85. The linear function used to combine PageRank values and classic VSM RSVs is $\alpha r + (1-\alpha)v$ where $\alpha = 0.5$, r is a PageRank value and v is a VSM RSV (α was set to 0.5 for the TREC experiments.)

A more detailed illustration is necessary to describe the experiments that tested the effectiveness of selecting links of some given type. Link semantics can enhance link-based retrieval or focused retrieval design. From the one hand, some link-based retrieval algorithms have recently been proposed, e.g. HITS or PageRank to simulate navigation being carried out by end users [3, 4]. Past experiments at TREC have not shown significant effectiveness improvements over the baselines. Probably, there are many noisy links and link filtering algorithms can be enhanced to consider types and filter noisy links out. We use two link types differing on the subgraph topology which they belong to. Such a link points to a given graph topology, independently on the node content. These links are likely to represent organizations of topics accordingly to the given structure, e.g. sequence or tree.

- A sequence link points to a sequence of pages. If the author(s) organize topics in a page sequence then topics are likely to be organized sequentially. We call (x, y) n-sequential link if points to a sequence of n pages. Note that, if (p_0, p_1) is a n-sequential link pointing to $(p_1, ..., p_n)$, then (p_0, p_1) is a 1-sequential link and (p_1, p_2) is a $(n-1)$-sequential link.

- A tree link points to page trees, i.e. page networks without cycles. We call (x, y) n-tree link if it points to a tree being rooted at y with minimum depth n from the root y. Note that, if (p_0, p_1) is a n-tree link, then (p_0, p_1) is a 1-tree link and there exists a $(n-1)$-tree link $(p_1, p_i), i \neq 1$.

Figure 1 depicts an example of 2-tree link (left) and of non-tree link (right), because of a cycle. Figure 2 depicts an example of sequence links. We report

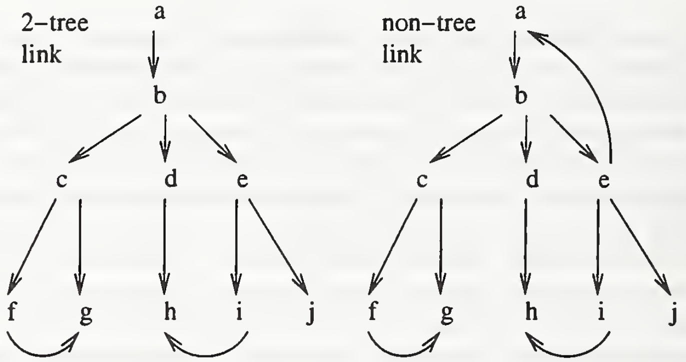


Figure 1: An example of tree and non-tree link.

some descriptive statistics on structure links. Test data consist of 1, 692, 096 full-text Web pages, 1, 532, 012 pages with in-going links, 1, 295, 841 pages with out-going links, 5.27 in-going links per page, 6.22 out-going links per page. All links are between pages that belong to the test collection; this means that no link points to a page, nor are pages pointed to by links starting outside the collection. Table 2 reports the distribution of sequence and tree links at different values of n, where n is defined above. Note that the percentages of structure, or tree links, out of the total number of in-going or out-going links vary. Most of the out-going links (92.5%) are n-tree links ($n < 10$), and 89.2% of the out-going links are 1-tree or 2-tree links. Only 7.1% out the out-going links are sequence out-going links. A small part of out-going links are not sequence nor tree links; for example, they may point to highly connected graphs. The large majority of structure in-going links are sequence links, yet they are a minority of the

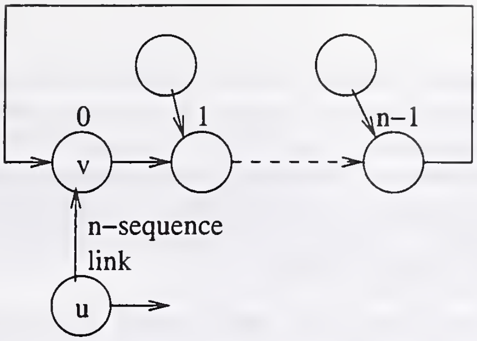


Figure 2: An example of sequences link.

set of in-going links (42.8%). The most apparent result from this preliminary experiment is that a link is likely to point to pages being entry points of small trees of depth 1 or 2. This means that the employed test sample of the Web is a sort of forest of many small trees. It is then likely that page contents are organized accordingly hierarchical structures. Further investigation would be needed to study the topology of these small trees and the relationship between sequence and tree links on the computation of estimates of the popularity and then the relevance of Web pages. The results that might be obtained can be used to enhance link-based retrieval algorithms.

3 Official Results

Table 4 and 5 report the summary of the official results for the ad-hoc task. DR performed better than any other run since 24 out of 50 topic resulted not below the median, while the other runs are below the median for many topics. This means that:

1. passage retrieval performed badly,

2. the combination of PageRank and classic vector space model gave no improvements,

3. selecting tree links gave no improvements in combining PageRank and classic vector space model.

4 Unofficial Results

PDWTAHSL and PDWTEPSL gave no significant variations with respect to other content-link runs.

References

[1] F. Crivellari and M. Melucci. Awir: Prototipo di un motore di ricerca per la raccolta, indicizzazione e recupero di documenti web sulla base dei loro frammenti. Rapporto tecnico T2-S12, Progetto INTERDATA - MURST e Università di Padova: "Metodologie e tecnologie per la gestione di dati e processi su reti Internet e Intranet". Tema 2: "Estrazione di informazioni distribuite sul WWW"., ftp://ftp-db.deis.unibo.it/pub/interdata/tema2/T2-S12.ps, Febbraio 1999. In Italian.

[2] World Wide Web Consortium (W3C) HTML Tidy. `http://www.w3.org/People/Raggett/tidy/`, October 2000. Last visited: October 25th, 2000.

[3] 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. Reprinted from [5].

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

[5] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the World Wide Web Conference*, 1998. `http://www7.scu.edu.au/programme/fullpapers/1921/com1921.htm`.

Run Id.	Run Type	Description	(Un)official
PDWTAHDR	content-only	standard vector space document retrieval using: single words and word pairs, no document normalization; documents retrieved against ad-hoc topics (501-550)	official
PDWTEPDR	content-only	standard vector space document retrieval using: single words and word pairs, no document normalization ; documents retrieved against entry-point topics (EP1-EP145)	official
PDWTAHPR	content-only	standard vector space using: single words and word pairs, no document normalization, prior retrieval 100-words passages, selection of 10000 top passages, retrieval of the corresponding documents; documents retrieved against ad-hoc topics (501-550)	official
PDWTEPPR	content-only	standard vector space using: single words and word pairs, no document normalization, prior retrieval 100-words passages, selection of 10000 top passages, retrieval of the corresponding documents; documents retrieved against entry-point topics (EP1-EP145)	official

Table 1: The summary of the performed runs. Legend: PD = Padova University, WT = Web Track, AH = Ad-Hoc topics, EP=Entry Point topics (homepage finding task), PR = Passage Retrieval, WL = Web In-Links: combination of content and pageranks, TL = Tree In-Links: like WL but only tree in-links are used, SL = Sequence In-Links: like WL but only sequence in-links are used

Run Id.	**Run Type**	**Description**	**(Un)official**
PDWTAHWL	content-link	PDWTAHDR is combined with Google pageranks using a linear function; pageranks are computed using the complete WT link file	official
PDWTAHTL	content-link	PDWTAHDR is combined with Google pageranks using a linear function; pageranks are computed using the tree links only discovered from the WT link file	official
PDWTAHSL	content-link	PDWTAHDR is combined with Google pageranks using a linear function; pageranks are computed using the sequence links only discovered from the WT link file	unofficial
PDWTEPWL	content-link	PDWTEPDR is combined with Google pageranks using a linear function; pageranks are computed using the complete WT link file	official
PDWTEPTL	content-link	PDWTEPDR is combined with Google pageranks using a linear function; pageranks are computed using the tree links only discovered from the WT link file	official
PDWTEPSL	content-link	PDWTEPDR is combined with Google pageranks using a linear function; pageranks are computed using the sequence links only discovered from the WT link file	unofficial

Table 2: The summary of the performed runs. Legend: PD = Padova University, WT = Web Track, AH = Ad-Hoc topics, EP=Entry Point topics (homepage finding task), PR = Passage Retrieval, WL = Web Links: combination of content and pageranks, TL = Tree Links: like WL but only tree links are used, SL = Sequence Links: like WL but only sequence links are used

	sequence		tree	
n	in	out	in	out
1	2,109,946	510,826	424,109	3,944,056
2	401,823	14,958	245,626	3,244,879
3	128,442	15,700	31,487	261,701
4	50,840	22,581	6,551	5,150
5	32,552	6,625	1,473	1,284
6	8,315	146	542	322
7	2,716	48	196	103
8	1,038	11	101	43
9	632	0	79	16
10	471	0	60	5
> 10	1985	0	196	0
total	2,738,760	570,895	710,431	7,457,559

Table 3: The distribution of sequence and tree links at different values of n.

Topic Id.	N.Rel.	Best	Median	TL	WL	PR	DR
501	62	18	7	4	13	3	14
502	81	18	6	0	8	5	6
503	33	12	6	2	1	1	1
504	18	13	8	1	9	5	9
505	24	17	11	2	8	0	8
506	2	2	1	0	1	0	1
507	17	11	5	0	1	0	1
508	47	16	7	5	4	2	4
509	140	25	18	3	10	5	10
510	39	25	18	1	9	9	14
511	165	21	16	6	10	7	10
512	14	7	4	0	3	1	3
513	58	13	6	6	3	4	3
514	79	17	8	6	8	5	9
515	41	11	6	5	6	7	7
516	30	9	3	1	3	5	4
517	60	15	3	0	2	3	2
518	84	16	2	1	8	3	7
519	149	20	6	3	4	8	5
520	18	6	3	1	2	0	3
521	57	16	1	0	2	0	2
522	6	5	3	0	3	1	3
523	79	19	3	2	2	11	2
524	35	12	2	2	0	1	0
525	41	11	4	0	3	2	3

Table 4: The summary of the official results (501-525).

Topic Id.	N.Rel.	Best	Median	TL	WL	PR	DR
526	49	9	3	5	2	5	5
527	93	23	15	1	20	15	22
528	6	4	3	0	3	2	3
529	39	21	12	4	11	9	12
530	124	25	19	5	14	9	15
531	22	12	0	0	0	0	0
532	34	12	9	4	9	10	8
533	77	21	13	0	8	1	8
534	8	2	0	0	0	0	0
535	46	9	2	1	5	4	4
536	19	11	5	0	2	2	3
537	25	13	1	0	0	0	0
538	2	2	2	0	2	2	2
539	29	7	2	1	3	1	3
540	12	3	1	1	1	0	1
541	372	23	13	9	16	6	15
542	38	1	0	0	0	0	0
543	24	9	0	0	0	0	0
544	324	30	24	21	27	28	26
545	32	11	2	3	5	1	5
546	36	11	2	2	1	2	1
547	144	17	7	2	5	4	4
548	2	2	2	0	2	2	2
549	367	22	9	9	19	17	20
550	60	12	5	0	1	1	3

Table 5: The summary of the official results (526-550).

PiQASso: Pisa Question Answering System

Giuseppe Attardi, Antonio Cisternino, Francesco Formica,
Maria Simi, Alessandrc Tommasi
Dipartimento di Informatica, Università di Pisa, Italy
{attardi, cisterni, formicaf, simi, tommasi}@di.unipi.it

"Computers are useless: they can only give answers" – Pablo Picasso –

Abstract

PiQASso is a Question Answering system based on a combination of modern IR techniques and a series of semantic filters for selecting paragraphs containing a justifiable answer. Semantic filtering is based on several NLP tools, including a dependency-based parser, a POS tagger, a NE tagger and a lexical database. Semantic analysis of questions is performed in order to extract keywords used in retrieval queries and to detect the expected answer type. Semantic analysis of retrieved paragraphs includes checking the presence of entities of the expected answer type and extracting logical relations between words. A paragraph is considered to justify an answer if similar relations are present in the question. When no answer passes the filters, the process is repeated applying further levels of query expansions in order to increase recall. We discuss results and limitations of the current implementation.

1. Architecture

The overall architecture of PiQASso is shown in Figure 1 and consists in two major components: a paragraph indexing and retrieval subsystem and a question answering subsystem.

The whole document collection is stored in the paragraph search engine, through which single paragraphs are retrieved, likely to contain an answer to a question.

Processing a question involves the following steps:

- question analysis
- query formulation and paragraph search
- answer type filter
- relation matching filter
- popularity ranking
- query expansion.

Question analysis involves parsing the question, identifying its expected answer type and extracting relevant keywords to perform paragraph retrieval. The initial query built with such keywords is targeted to high precision and to retrieve a small number of sentences to be evaluated as candidate answers through a series of filters. This approach was inspired by the architecture of the system FALCON [5]. PiQASso analyzes questions and answer paragraphs by means of a natural language dependency parser, Minipar [2].

The semantic type filter checks whether the candidate answers contain entities of the expected answer type and discards those that do not.

A semantic filter identifies relations in the question, and looks for similar relations within candidate answers. Relations are determined from the dependency tree provided by Minipar. A matching distance between the question and the answer is computed. Sentences whose matching distance is above a certain threshold are discarded. The remaining sentences are given a score that takes into account the frequency of occurrence among all answers. The highest ranking answers are returned.

If no sentence passes all filters, query expansion is performed to increase paragraph recall. The whole process is iterated using up to five levels of progressively wider expansions.

PiQASso is a completely vertical system, made by linking several libraries into a single process, which performs textual analysis, keyword search and the semantic filtering. Only document indexing is performed offline by a separate program.

2. Paragraph Search Engine

PiQASso document indexing and retrieval subsystem is based on IXE [1], a high-performance C++ class library for building full-text search engines. Using the IXE library, we built a paragraph search engine, which stores the full documents in compressed form and retrieves single paragraphs. However, we do not simply index paragraphs instead of documents: this approach is not suitable for question answering since relevant terms may not all appear within a paragraph, but some may be present in nearby sentences.

Our solution is to index full documents and to add sentence boundary information to the index, i.e. for each document, the offset to the start of each sentence. A sentence splitting tool is applied to each document before indexing.

The queries used in PiQASso consist of a proximity query involving the most important terms in the question

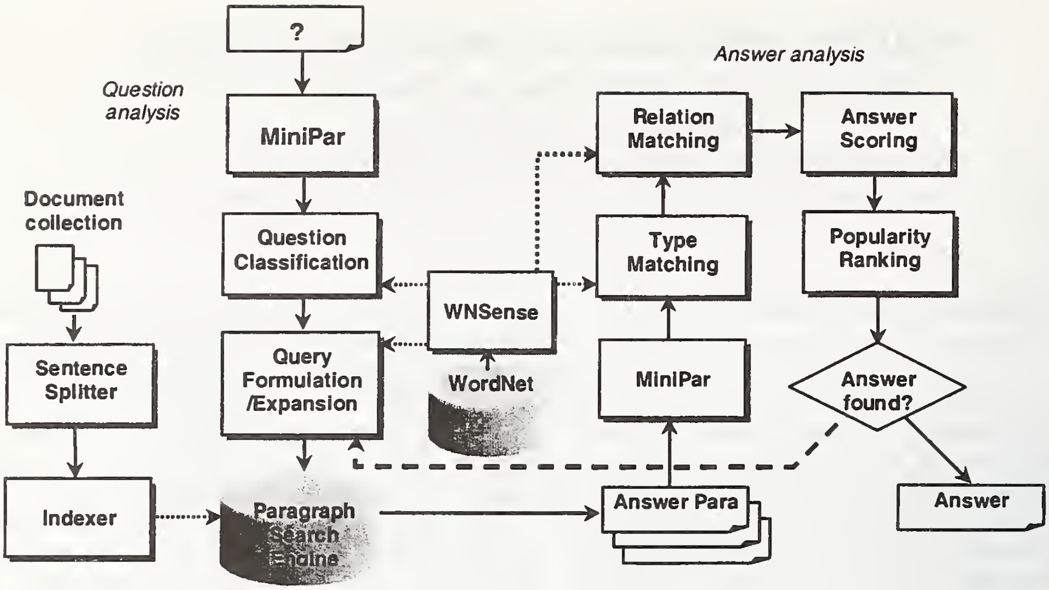


Figure 1. PiQASso Architecture.

combined in AND with the remaining terms. Such queries select documents containing both relevant context for the question and paragraphs where the required words occur. The paragraph engine ranks each paragraph individually and extracts them from the source document exploiting sentence boundary information.

The sentence splitter is based on a maximum entropy learning algorithm as described in [9].

Since sentence splitting is quite time consuming, performing it at indexing time improves significantly PiQASso performance. On a 1 MHz Pentium 3, a paragraph search on the whole Tipster collection takes less than 50 msec. Since documents are stored in compressed form, accessing the individual paragraphs requires an additional amount of time for performing text decompression, which depends on the number of results.

3. Text analysis tools

Our approach to Question Answering relies on Natural Language Processing tools whose quality and accuracy are critical and influence the overall architecture of the system. We performed experiments with various tools and we had to adapt or to extend some of them for achieving our aims.

We briefly sketch the main NLP tools deployed in PiQASso:

- the dependency parser Minipar
- WNSense: an interface to WordNet
- a Named Entity tagger.

3.1. Minipar

Sentences are parsed by means of Minipar [2], producing a dependency tree which represents the dependency relations between words in the sentence. A dependency relationship is an asymmetric binary relationship between a word called *head*, and another word called *modifier*. A word in the sentence may have several modifiers, but each word may modify at most one word. Figure 2 shows an example of a dependency tree for the sentence *John found a solution to the problem*. The links in the diagram represent dependency relationships. The direction of a link is from the head to the modifier in the relationship. Labels associated with the links represent types of dependency relations. Table 1 lists some of the dependency relations.

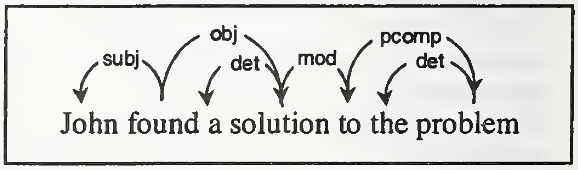


Figure 2. Sample dependency tree.

The root node does not modify any word, and is given an empty node type. Other empty nodes may be present in the tree. For instance, in the parse tree for sentence "*It's the early bird that gets the worm*", the word "*that*" is identified as the subject of the "*gets the worm*" subordinate phrase, and an empty node is inserted to represent the subject of the verb "*gets*", including a reference to the word "*bird*". The parser identifies "*that*" as the subject of "*gets*" and "*the early bird*" as an additional one.

This is an instance of the problem of *coreference resolution*: identifying the entity to which a pronoun refers. From the dependency tree built by Minipar we gather that "*the early bird*" is the subject of "*gets the worm*", enabling us to answer a question like "*who gets the worm*?", even if question and answer are stated in slightly different syntactic forms.

Minipar has some drawbacks: its parsing accuracy is not particularly high and the selection of dependency relations is somewhat arbitrary, so that two similar phrases may have quite different, although correct, parses. We apply several heuristic rules to normalize the dependency tree and to facilitate identifying the most essential and relevant relations for comparing questions and answers.

Relation	Description
i	main verb
subj, s	subject of the verb
obj, obj*n*	object of the verb
pcomp-n	prepositional complement
appo	appositive noun
gen	genitive
inside	location specifier
nn	nominal compound
lex-mod	lexical modifier
det	determiners
mod	Modifiers (adjs, advs, preps)
pred	Predicate
aux	Auxiliary verb
neg	negative particle

Table 1: some relations in Minipar output.

Minipar is also capable of identifying and classifying named entities (NE). Using its own internal dictionary, plus a few rules, it detects word sequences referring to a person, a geographic location or an amount of money.

3.2. WNSense

We built WNSense (WordNetSense) as a tool for classifying word senses, assigning a semantic type to a word, and evaluating semantic distance between words based on hyperonymy and synonymy relations. WNSense exploits information from WordNet [7], for instance to compute the probability of a word sense.

3.2.1. Sense Probability

Senses are organized in WordNet in distinct taxonomies (for instance, the word "crane" has senses in the "animal" taxonomy as well as in the "artifact" one). During sentence analysis PiQASso often needs to determine whether a word belongs to a certain category: e.g. it is of the expected answer type. This can be estimated by computing the probability for the word sense to belong to a WordNet category (e.g., the probability of the sense of the word "cat" to fall within the "animal" category).

WordNet orders word senses by frequency. Given such ordered list of senses $\{s_0, \dots, s_n\}$ for a word w, we compute the probability that the sense for the word belongs to category C as follows:

$$P(w, C) = k\sum_{j=0}^{n}\frac{n-j}{n}\gamma(s_j)$$

where

$$\gamma(s_j) = \begin{cases} 1 & if\ s_j \in C \\ 0 & otherwise \end{cases}$$

and k is a parameter of the heuristics, roughly the probability that the first WordNet sense is the correct one (currently, k is at 0.7).

3.2.2. Word Type

The type for a word w is computed as:

$$\arg\max_{C\in TLC} P(w,C)$$

i.e. the category C among those in *TLC*, to which the word belongs with the highest the probability. The *TLC* categories used by PiQASso are the 23 top-level categories from WordNet corresponding to nouns, from the total of 45 lexical files into which WordNet organizes synsets.

3.2.3. Word Distance

A measure of word distance is used for estimating the distance between two sentences, in particular an answer paragraph and a question.

Word distance for hyperonyms is based on the distance in depths of their senses in the WordNet taxonomy. The depth differences are normalized dividing them by the taxonomy depth, so that a depth difference of 1 in a detailed taxonomy has less influence than a difference of 1 in a coarser one. The depth differences for all pairs of senses of two words are weighted according to the probabilities of both senses and added together.

Word distance for two synonyms is also computed over all their senses, weighted according to their probability.The distance between two words, denoted by $dist(w_1, w_2)$, is defined as either their synonym distance, if they are synonyms, or else their hyperonym distance.

3.2.4. Word Alternatives

Alternatives for a word are required during query expansion. They are computed considering the union W of all synsets containing the word w. The set of alternatives for w is defined as:

$$\{ s \in W \mid dist(w, s) < th \}$$

where th is a fixed ceiling, useful to avoids cases where a synonym of w has meanings not typically related to w (e.g. "*machine*" for "*computer*").

3.3. Named Entity Tagger

The NE tagger in Minipar can achieve high precision in NE recognition, since it is dictionary-based, but it has limitations (for instance it does not handle unknown names). Therefore we integrated it with an external tagger, based on a maximum entropy probabilistic approach [3] that uses both a part-of-speech tagger (TreeTagger [10]) and a gazetteer to determine word features.

The Named Entity extractor identifies person names, organizations, locations, quantities and dates, and assigns to them one of the semantic types as defined in MUC [4].

To maintain uniformity of treatment, the tags produced by the NE tagger are integrated within the same tree produced by Minipar as additional semantic features for the corresponding words.

4. Question analysis

Question analysis extracts or identifies the following information from the question:

- the keywords to be used in the paragraph search;
- the expected answer type;
- the location of the answering entity.

These pieces of information correspond to three successive steps in the process of question answering: keyword based retrieval, paragraph filtering based on the expected answer type, and logical relation matching between questions and answer paragraphs.

4.1. Keyword Extraction

The first step selects words from the question for generating a suitable paragraph query.

PiQASso considers the adjectives, adverbs, nouns and verbs in the question, excluding words from a list, determined experimentally, which includes:

- nouns such as "*type*", "*sort*", "*kind*", "*name*", frequently occurring in questions but unlikely to occur in answers;
- generic verbs like "be", rhetorical ones like "*call*", auxiliary verbs;
- adjectives that qualify "*how*" (as in "*how long*", "*how far*", etc.).

Words to which the parser does not assign the part-of-speech tag are discarded: including them did not have a clear effect on performance, according to our experiments.

4.2. Question Classification

The *expected answer type* is the semantic type of the entity expected as the answer to the question. The expected answer type is helpful for factual questions, but not for questions that require complex explanations.

Irrelevant sentences can be often discarded simply by checking whether they contain an entity of the expected type. The TREC 2001 QA main task requires answers shorter than 50 characters and this entails that only factual questions are asked (no long explanations can be returned as answers).

PiQASso uses a coarse-grained question taxonomy consisting of five basic types corresponding to the entity tags provided by Minipar (`person`, `organization`, `time`, `quantity` and `location`) extended with the 23 WordNet top-level noun categories. The answer type can be a combination of categories, like in the case of "*Who killed John Fitzgerald Kennedy*?", where the answer type is `person` or `organization`. Categories can often be determined directly from a wh-word: "*who*", "*when*", "*where*".

The category for "*how <adjective>*" is determined from the adjective category: "*many*", "*much*" for quantity, "*long*", "*old*" for time, etc.

The type for a "*what <noun>*" question is normally the semantic type of the noun, as determined by WNSense. For instance in "*what king signed the Magna Charta*?", the semantic type for "*king*" is person. When feasible, the WordNet category for a word is mapped to one of the basic question types. The other cases are mapped to one of the top-level WordNet categories by means of WNSense: "*what metal has the highest melting point*" has the semantic type "*substance*".

For "*what <verb>*" questions the answer type is the type of the object of the verb. "*What is*" questions, which expect a definition as an answer ("*what is narcolepsy*?", "*what is molybdenum*?") are dealt specially. The answer type is the answer itself (a disease, a metal). However, it is often not possible to just look up the semantic type of the word, because lack of context does not allow identifying the right sense. Therefore, we treat definition questions as type-less questions: entities of any type are accepted as answers (skipping the semantic type filter), provided they appear as subject in an is-a sentence.

4.3. Proper and Common Names

Questions whose expected answer type is `person` require special treatment. A question like "*who is Zsa Zsa Gabor?*" expects a definition, and therefore a common noun as an answer, while a question like "*who is the king who signed the Magna Charta?*" expects a proper noun. Therefore the rule for the case of a `person` answer type is: if the question contains a proper noun, a common noun is expected and vice versa.

4.4. Relations between Words

Relations between words in the question are determined from the dependency tree built by Minipar. In a question like "*who killed John F. Kennedy*", Minipar identifies the verb "*killed*" as having "*Kennedy*" as object, and a missing subject (represented by an empty node – a node with no corresponding word). In a possible answer sentence the verb "*kill*" may appear with exactly "*Kennedy*" as object. However the same relation could also be stated in a quite different syntactic form, where the dependencies are not so explicit, but require more complex analysis of the tree, as discussed later.

4.4.1. Identifying the Answer Node

In the dependency tree of the question we must identify the node that represents the object of the question. We call this the *answer node*, since it can be considered as a placeholder to be matched with the answer object in the answer paragraph. The answer node will have the answer type as determined above. Often this node exists and is empty: it corresponds to the missing subject of a verb, as in "*who killed John F. Kennedy*". In other cases the node is not empty: for instance in "*What instrument did Glenn Miller play?*", the answer node corresponds to the word "*instrument*". The answer type of the question is "*artifact*" and the semantic type of the word "*instrument*". In a direct answer like "*Glenn Miller played trombone*" the answer entity ("*trombone*") occurs in the place held by the word "*instrument*" in the question.

Head	Relation	Modifier
play	obj	instrument
play	s	Miller
play	s	Glenn
Miller	lex-mod	Glenn

Table 2: relations for the sentence "*what instrument did Glenn Miller play?*"

By experimenting with a number of questions and analyzing Minipar output, we noticed that the answer node often corresponds to the first empty subject node in the question. This is because the main verb in a question is often the first one, and because an empty subject means that the actual subject is missing.

An exception to this rule is when the required entity does not participate in the action as a subject, as in question "*in what year did the Titanic sink?*". In this case, the answer is a complement, and the answer node is still in relation with the verb, but as a complement instead of as a subject. In such cases, there is no such node in the output of the parser, and a new one must be created.

These simple heuristics are effective for simple questions: dealing with more involved expressions will require extending such heuristics, since determining the answer node is a critical issue in our approach.

5. Query Formulation and Expansion

5.1. Query formulation

The first iteration in the question answering process performs keyword extraction and query formulation.

A keyword search is performed for selecting candidate answer sentences from the whole Tipster document collection. Further iterations perform various level of query expansion, each allowing larger recall in the search.

PiQASso only addresses the problem of finding answers that are fully justified within a single paragraph. This simplifies textual analysis, as only one sentence at a time needs to be analyzed.

Although sentence boundaries information is stored in the index, search is performed document-wise, in order to achieve better recall. Consider two sentences like: "*Neil Armstrong walked on the moon. Armstrong was the first man to walk on Earth's satellite*". A question like "*Who was the first man to walk on the moon?*" would yield the keywords "*first*", "*walk*", "*moon*" and "*man*". However, while the two sentences contain all those keywords, none of them does by itself. Instead of looking for keywords within a each individual sentence, PiQASso performs a proximity search, looking for terms within a specified word position distance. For the above example, the query could be:

```
proximity 100 ((first) & (walk*) & (moon))
```

which looks for the term "*first*", for the prefix "*walk*" and for the word "*moon*" within a window of one hundred words. Such window would spans across the two sentences above, which would be both returned, individually, as candidate answers.

Keyword expansion would hardly propose "*satellite*" as an alternative to "*moon*", and therefore the second sentence would not be returned if paragraphs had been indexed separately. When evaluating the second sentence within the last filter, the match between "*moon*" and "*satellite*" will be given a certain distance as hyperonym, allowing the paragraph to pass the filter.

We use the following criterion for choosing the size of the proximity window. In principle question and answer lengths are not strictly related: an answer to a short question may appear within a very long sentence, and vice versa, an answer to a complex question could be much shorter than the question itself. However, it seems reasonable to expect that the keywords in an answer paragraph are not too spread apart. The size of the proximity window is twice the number of nodes in the question parse tree (including irrelevant or empty nodes that may account for complicated sentences).

The heuristics proposed for keyword extraction is adequate for short questions, but returns too many terms for long questions. Thus, we split keywords into two sets: those that must appear within the proximity window and those that must occur anywhere in the document. The generated query consists of a conjunction of those terms that must be found in the document, and of a proximity query.

Terms are put in either of the two sets depending on the distance of the term from the root of the parse tree. Terms closer to the root, and therefore more central to the question, are required to be appear within the proximity window, while the others are accessory: they are only requested to occur in the document, but may be missing from the sentence.

5.2. Query Expansion

The first expansion step tries to cope with morphological variants of words by replacing each keyword with a prefix search, obtained from stemming the original word. Certain prefixes that appear frequently in questions are discarded: for instance "*locate*", "*find*", "*situate*" in questions expecting a location as an answer, "*day*", "*date*", "*year*" in questions expecting a date and so on. We use about a dozen of such exceptions, which correspond to cases in which the type makes these words superfluous.

Stemming is performed using Linh Huynh implementation of *Lovins's stemmer* [6].

In the second expansion cycle we broaden the search by adding (in `or`) the *synonyms* of the search terms. Synonyms are looked up in WordNet by means of WNSense. Synonyms are stemmed as well.

In the third and fourth expansion cycles, we increase recall by dropping some search terms. During the third cycle, adverbs are dropped.

During the last expansion cycle, if the query contains more than three keywords in conjunctive form, verbs are discarded, as well as person's first names when the last name is present. If after such a pruning there are still more than three keywords in `and`, then we also drop those keywords whose parent (as from the dependence tree) is already within the keywords to be searched. This has the effect, if looking for a "black cat", to perform a search for a "cat", black being a modifier of cat, and therefore depending on it.

6. Type Matching

The sentences returned by the query are analyzed and checked for the presence of entities of the proper answer type, as determined by question analysis.

Sentences are parsed and recognized entities are tagged. The tree is then visited, looking for a node tagged with the expected answer type.

We also check whether an entity which occurs in the sentence is already present in the question. A question like "*Who is George Bush's wife?*" expects a proper person name as an answer. The sentence "*George Bush and his wife visited Italy*" contains a proper person name, but does not answer the question. Such sentences occur frequently (the search keywords being "*George*", "*Bush*" and "*wife*"), so it is convenient to discard them as early as possible.

Sentences not verifying this condition are rejected.

7. Relation Matching

Sentences that pass the answer type filter are submitted to the relation matching filter, which performs a more semantic analysis, verifying that the answer sentence contains words that have the same type and relation than corresponding words in the question.

The filter analyzes Minipar output in order to:

- determine a set of relations between nodes present both in the question and in the sentence;
- look for relations in the answer corresponding to those in the question;
- compute the distance of each candidate answer and select the one with the lower distance.

In order to simplify the process, not all the nodes in the question and in the answer are considered. The same criterion used for selecting words as search keywords is applied also in this case: nouns, verbs, adjectives and adverbs are relevant (including dates and words with unknown tag).

During this analysis, the parser tree is flattened into a set of triples (H, r, M): head node, relation, modifier node. This representation is more general and allows us to turn the dependency tree into a graph.

In fact it is often useful to make certain relations explicit by adding links to the parser tree. For instance in the phrase "*Man first walked on the moon in 1969*", "*1969*" depends on "*in*", which in turns depends on "*moon*". According to our criterion, "*in*" is not a relevant node and so it will not be considered. We can however short circuit the node by adding a direct link between "*moon*" and "*1969*". More generally, we follow the rule that whenever two relevant nodes are linked through an irrelevant one, a link is added between them. Similarly, since the NE tagger recognizes "*1969*" as a date, it is convenient to add a link between the main verb ("*walk*") and the date, since the date modifies the action expressed by the verb.

7.1. Extracting Relations

Questions and answers are analyzed in order to determine whether relations present in a question appear in a candidate answer as well.

PiQASso exploits the relations produced by Minipar and infers new relations applying the following rules:

Direct link if B is a child of A in Minipar output, A is related to B according to their relation in the parse tree.

Conjunctions Relations distribute over conjunctive links. For example, in "*Jack and John love Mary*", the relation between John and Mary is distributed over the conj link between John and Jack, i.e. a "*love*" relation between Jack and Mary is inferred.

Predicates A and B are related if they are both child of a "*to be*" verb, the first with the role of subject, the second as predicate (which is the relation between them). This is because a question like "*who is the Pope*?" is often answered by phrases such as "*The Pope, John Paul II, ...*" in which the answer does not go through a "*to be*" verb.

Possession A and B are related with the relation of genitive if A is the subject of a verb "*to have*" and B is the object. The rule enables matching "*John's car*" with "*John has a car*".

Location A and B are in inside relation if there is a subj-in relation between B and A (phrase of the form "*A is in B*"). This allows matching "*Paris is in France*" with "*Paris, France*".

Invertible relations Some relations are invertible, so that A and B are in relation if B and A are in relation as either apposition (a particular case of nominal compound) or by the person relation (a relation between the first and second name of a person).

Dates Minipar links a modifier (e.g. "*in 1986*") to the closest noun: a relation between the main verb (describing the action) and the date is inferred.

Empty nodes Empty nodes represent an implicit element of the sentence. Minipar sometimes can determine the word they refer to: in this case we add a relation between A and B if there is a relation between A and C, and C is an empty node referring to B (and dually for the first node in the relation).

Negation A relation between two nodes is discarded if both nodes depend from a node with a negative modifier. This accounts for negative phrases like "*John is not a policeman*" and avoids inferring a relation between "*John*" and "*policeman*". This rule has precedence over the others.

7.2. Finding a match

Suppose the following relations appear in a question: $qr_1 = (A, r_1, B)$ and $qr_2 = (A, r_3, C)$. Suppose the following relations are present in a candidate answer sentence: $ar_1 = (1, R_1, 2)$, $ar_2 = (2, R_2, 3)$ and $ar_3 = (1, R_3, 3)$. All matches between triples in the question and in the answer are considered, provided that no node is put in correspondence with two different nodes: if we match qr_1 with ar_1, we cannot match qr_2 with ar_2, or node A in the question would have to match both nodes 1 and 2 in the answer.

The match with the smallest distance is selected.

7.3. Matching Distance

An answer paragraph can be considered as a close answer to a question if it contains nodes and relations corresponding to all nodes and relations in the question. For each missing node the distance is increased by an amount that depends on the relevance of the node. To represent this relevance we associate a *mismatch distance*, $mmd(n)$, to each node n in the question. For instance the mismatch distance is small for the node corresponding to the question type (e.g. the node "*instrument*" in a previous example), since it may be missing in the answer. Nodes depending on other relevant nodes have half the mismatch distance of their parents: they may express a specification that a correct answer need not contain.

The overall *matching distance* between a question and a candidate answer is computed by summing, for each node n in the question:

$$\begin{array}{ll} mmd(n) \cdot dist(n, m) & \text{if } n \text{ matches node } m \text{ in the answer} \\ mmd(n) & \text{otherwise} \end{array}$$

and similarly for each relation in the question.

The distance is incremented to account for special situations, e.g. when the answer is too specific: for the question "*Who was the first man in Space*?", "*The first American in space was ...*" is not a proper answer, contrary to a naïve rule that wants a specific answer correct for a general question.

If the answer sentence does not contain an entity of the expected answer type, the distance is set to infinity, to ensure that the paragraph is rejected.

The maximum distance from the question allowed by the filter (distance ceiling) may be set to either: a) very high, so that any sentence matching the answer node will pass the filter (recall-oriented); or b) proportional to the number of nodes in the question. PiQASso implements a simple tightening strategy whereby the ceiling is decreased at each expansion iteration, avoiding too much garbage in the early phases.

8. Answer Popularity

After the TREC 2001 submission we introduced a criterion for selecting answers based on a measure of answer popularity, which proved quite effective.

Answers are grouped according to the value contained in their answer node. A score is assigned to each group proportional to the average of the matching distances in the group and inversely proportional to the cardinality of the group. Groups are sorted by increasing score value and only the answer with the smallest matching distance in each group is returned.

The criterion combines a measure of difference to the question and a measure of likelihood based on how often the same answer was offered. Selecting only one answer per group ensures more variety in the answers.

9. Results

PiQASso achieved the scores summarized in Table 3, expressed as MRR (Mean Reciprocal Rank) of up to five answers per question. The official score, computed from the judgments of NIST assessors at TREC 2001, ranks PiQASSo in 15th overall position. PiQASso achieved the same score for both strict evaluation (answers supported by a document in the collection) and lenient evaluation (answer not supported), since it does not use any external source of information. The unofficial score was computed by our own evaluation of the results on a new run of the system after the addition of the popularity ranking.

Run	MRR Strict	MRR Lenient
TREC 2001, official	0.271	0.271
TREC 2001, unofficial	0.32	

Table 3: Scores in the TREC 2001 QA main task.

A peculiarity of the TREC 2001 questions was the presence of a higher than usual percentage (almost 25%) of definition questions that could have been answered by simple lookup in a dictionary or from other sources (e.g. the Web), as some other systems did. For PiQASso we concentrated in improving the system ability to analyze and extract knowledge from the given document collection.

10. Assessment

In order to assess the effectiveness of the various filters, and how they affect the overall performance we performed some measurements using a subset of 50 questions of the TREC 2001 set. Results are summarized in Table 4. For half of the questions (49%), no paragraph passed all filters. The great majority of answers are obtained from the results of the first IR query.

	%
Questions for which no answer was found	49
Questions answered by first query (over all answers)	92
Questions for which no paragraph was retrieved	2

Table 4: Filter effectiveness.

The benefits of iterating the process after performing query expansion are less than expected.

Overcoming this limit requires improving query expansion to produce more word alternatives or morphological variations. This may however complicate the task of the relation matching filter, which also needs to be refined: the paragraphs retrieved by the initial query (without stemming or synonym expansion) are simpler to match with the question and produce most of the answers. When more complex paragraphs are retrieved by the more complex queries, matching is more difficult and rarely an answer is found.

The current system is not capable, for example, of matching the sentences "*John loves Mary*" and "*John is in love with Mary*", since their main verbs "*love*" and "*to be*" are different. Either deeper semantic knowledge would be required or a collection of phrase variants, that might be built automatically with the method suggested by Lin [10], discovering similarities in paths within the dependency graph of the parser.

11. Conclusions and Future Work

PiQASso is engineered as a vertical application, which combines several libraries into a single application. With the exception of Minipar and WordNet, all the components in the architecture were built by our team, including the special purpose paragraph indexing and search engine, up to the tools for lexical analysis, question analysis and semantic filtering.

PiQASso is based on an approach that relies on linguistic analysis and linguistic tools, except for passage retrieval, where it exploits modern and efficient information retrieval techniques. Linguistic tools provide in principle higher flexibility, but often appear brittle, since implementations must restrict choices to reduce the efffects of combinatorial explosions. One way to improve their performance would be by providing them with large amounts of semantic data in a preprocessed form: for instance generating a large number of variants from the phrases in the document collection and matching them with effective indexing techniques and statistical estimates, rather than performing sophisticated matching algorithms.

PiQASso is heavily dependent on Minipar since it relies on the dependency relations it creates. Such relations are often too tied to the syntactic form of the sentence for our purposes, so we had to add specific processing rules to abstract from such representation and to work around certain of its idiosyncrasies.

Question analysis could be improved by adopting a finer-grained taxonomy for the expected answer type. Such granularity requires support by the named entity tagger.

Keyword extraction/expansion would benefit from a better identification of the sense of a word, so that fewer and more accurate alternatives can be used in the query formulation. Current figures show that present keyword expansion is not effective, for it either does not add results, or it adds too many, returning way too many hits for the system to analyze them all. As for about half of the questions our system did not find any answer at all (which gives us outstanding improvement margins), this seems a necessary step.

Acknowledgements

Alessandro Tommasi is supported by a PhD fellowship from Microsoft Research Cambridge.

Cesare Zavattari contributed to various aspects of the implementation, in particular of the sentence splitter.

12. References

[1] G. Attardi, A. Cisternino, Reflection support by means of template metaprogramming, *Proceedings of Third International Conference on Generative and Component-Based Software Engineering, LNCS*, Springer-Verlag, Berlin, 2001.

[2] D. Lin, LaTaT: Language and Text Analysis Tools, *Proc. Human Language Technology Conference*, San Diego, California, March 2001. http://hlt2001.org.

[3] A. Berger, S. Della Pietra, and M. Della Pietra, A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1), 1996.

[4] R. Grishman and B. Sundheim, Design of the MUC-6 evaluation. In NIST, editor, *The Sixth Message Understanding Conference (MUC-6)*, Columbia, MD. NIST, Morgan-Kauffmann Publisher, 1995.

[5] S. Harabagiu, Moldovan et al., FALCON: Boosting Knowledge for Answering Engines. *TREC 2000 Proceedings*, 2000.

[6] J. B. Lovins, Development of a Stemming Algorithm. *Mechanical Translations and Computational Linguistics*, 11: 22-31, 1968.

[7] G. Miller, Five papers on WordNet. *Special issue of International Lexicography* 3(4), 1990.

[8] J. C. Reyner and A. Ratnaparkhi, A Maximum Entropy Approachg to Identify Sentence Boundaries. *Computational Language*, 1997.

[9] G. Schmid, TreeTagger – a language independent part-of-speech tagger, 1994. Available: http://www.ims.uni-stuttgart.de/Tools/DecisionTreeTagger.html.

[10] D. Lin and P. Pantel, Discovery of Inference Rules for Question Answering. To appear in the *Journal of Natural Language Engineering*, 2001.

Keep It Simple Sheffield – a KISS approach to the Arabic track

Mark Sanderson and Asaad Alberair
m.sanderson@shef.ac.uk, ir.shef.ac.uk

Department of Information Studies, University of Sheffield,
Western Bank, Sheffield, S10 2TN, UK

Abstract

Sheffield's participation in the inaugural Arabic cross language track is described here. Our goal was to examine how well one could achieve retrieval of Arabic text with the minimum of resources and adaptation of existing retrieval systems. To this end the public translators used for query translation and the minimal changes to our retrieval system are described. While the effectiveness of our resulting system is not as high as one might desire, it nevertheless provides reasonable performance particularly in the monolingual track: on average, just under four relevant documents were found in the 10 top ranked documents.

Introduction

One of the truisms (almost a law) of information retrieval is that the more data one searches, the less language processing is required to match on at least some relevant documents. When searching a collection of image captions, for example, one is likely to be keen to locate any 'hits' between query and caption. When searching the Web, however, being overwhelmed with hits is a more likely problem; linguistically adjusting the query to match on more Web pages is not necessary. In Sheffield's first attempt at Arabic retrieval, it was decided (due to a combination of curiosity and lack of linguistic resources) to see how effective retrieval could be when very little linguistic processing of the query or document took place.

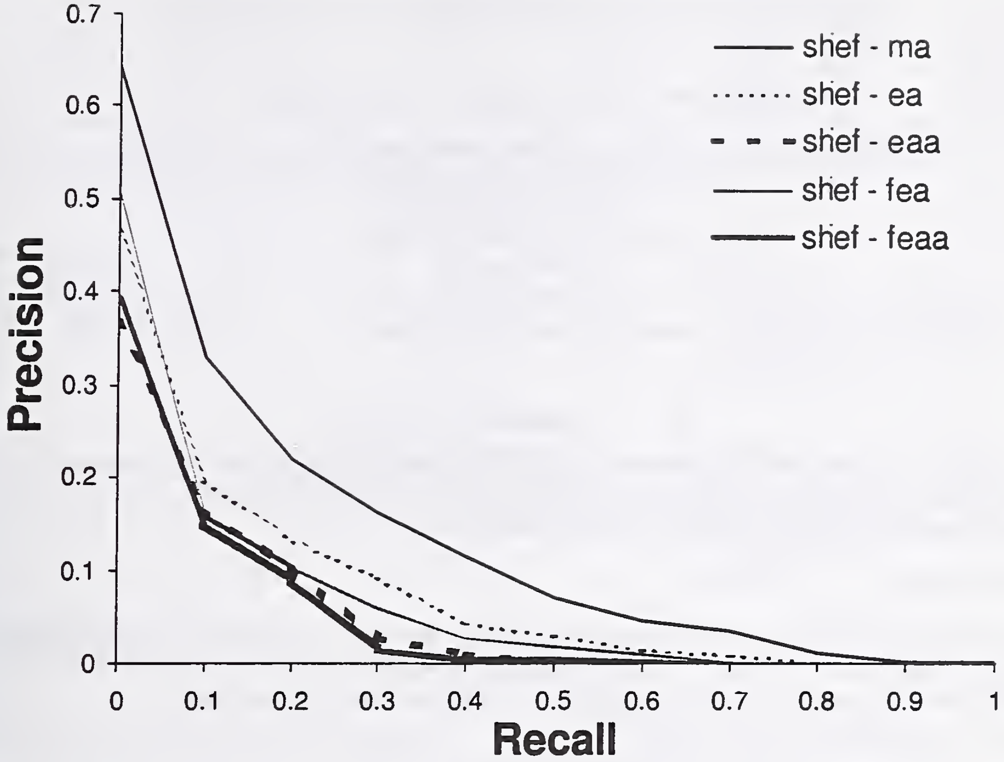
This paper describes the adjustments made and minimal resources exploited to allow an IR system to conduct all aspects of the Arabic track: Arabic monolingual, processing English version of the queries; and finally dealing with French queries. The set up is described first, followed by the runs and results before concluding.

Set up

The retrieval system used in Sheffield's experiments was the GLASS experimental retrieval system. The suite of programs that make up GLASS was written to serve the experiments of the first author's PhD, the system has continued to be used in a range of applications since then (Purves, 1998, Gollins, 2001). The retrieval system has recently been adjusted to use BM25 ranking (Robertson, 1994). In order to be able to handle the Arabic documents, a new GLASS tokeniser was created to deal with the texts' UTF-8 encoding. An Arabic speaker (the second author) manually checked initial word lists generated by the tokeniser and provided an updated list of characters that signify word breaks. No stop word list was used, however ranking optimisations akin to those proposed by Persin (1994) were employed to speed up the retrieval process. The morphological variation of Arabic words is greater than that found in English. Given the relatively large size of the collection being searched (approximately ½Gb), however, it was hoped that a sufficient number of relevant documents would match the unprocessed query words to allow the system to be reasonably effective in the top ranks. A web-based interface to Arabic GLASS was created to enable the Arabic speaker to run a few test queries on the system[1]. This is the full extent of adjustments made to the core retrieval system.

In order to enable cross-language retrieval, the English and French queries were translated using public Web-based translation systems. English to Arabic was conducted using mainly the almisbar[2] and

[1] Arabic display and text entry was an extensible feature of the Web browser used: IE v5.0.
[2] http://www.almisbar.com



occasionally ajeeb[3] public translator web sites. As no public French to Arabic translator was located French was translated into English (a pivot), using Babel Fish on AltaVista[4], before then being translated into Arabic.

All retrievals were conducted using the title part of the query only.

Runs

Sheffield submitted five runs to TREC: a monolingual run; two English cross language runs; and two French cross language runs. They are now described.

- Monolingual
  - shefma - here, the title of the Arabic queries was submitted to GLASS and the retrieval runs noted.
- English cross language
  - shefea - the title of the English queries was translated into Arabic using almisbar.com.
  - shefeaa - here, two separate versions of the Arabic query was created, the first using almisbar and second using another Arabic translation facility, ajeeb.com. The two Arabic queries were simply concatenated. The idea of using both translators was the hope that any failing in one translator (such as lack of vocabulary coverage) would be covered by the success of the other.
- French cross language
  - sheffea - the title of the French queries was translated into English using Babel Fish, and this was as with shefea translated into Arabic using almisbar.
  - sheffeaa - as with shefeaa, once the French query was in English form, it was translated twice into Arabic using ajeeb as well.

[3] http://ajeeb.com
[4] http://www.altavista.com

The Use of External Knowledge in Factoid QA

Eduard Hovy, Ulf Hermjakob, Chin-Yew Lin

Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292-6695
tel: 310-448-8731
fax: 310-823-6714
email: {hovy,ulf,cyl}@isi.edu

Abstract

This paper describes recent development in the Webclopedia QA system, focusing on the use of knowledge resources such as WordNet and a QA typology to improve the basic operations of candidate answer retrieval, ranking, and answer matching.

1. Introduction

The Webclopedia factoid QA system increasingly makes use of syntactic and semantic (world) knowledge to improve the accuracy of its results. Previous TREC QA evaluations made clear the need for using such external knowledge to improve answers. For example, for definition-type questions such as

Q: what is bandwidth?

the system uses WordNet to extract words used in the term definitions before searching for definitions in the answer corpus, and boosts candidate answer scores appropriately. Such definitional WordNet glosses have helped definition answers (10% for definition questions, which translates to about 2% overall score in the TREC-10 QA evaluation, given that as many as a little over 100 out of 500 TREC-10 questions were definition questions).

This knowledge is of one of two principal types: generic knowledge about language, and knowledge about the world. After outlining the general system architecture, this paper describes the use of knowledge to improve the purity of phase 1 of the process (retrieval, segmenting, and ranking candidate segments), and to improve the results of phase 2 (parsing, matching, and ranking answers).

Webclopedia adopts the by now more or less standard QA system architecture, namely question analysis, document / passage retrieval, passage analysis for matching against the question, and ranking of results. Its architecture (Figure 1) contains the following modules, which are described in more detail in (Hovy et al., 2001; Hovy et al., 2000):

- **Question parsing:** Using BBN's IdentiFinder (Bikel et al., 1999), the CONTEX parser produces a syntactic-semantic analysis of the question and determines the QA type.
- **Query formation**: Single- and multi-word units (content words) are extracted from the analysis, and WordNet synsets are used for query expansion. A series of Boolean queries is formed.
- **IR**: The IR engine MG (Witten et al., 1994) returns the top-ranked *N* documents.
- **Selecting and ranking sentences**: For each document, the most promising $K<<N$ sentences are located and scored using a formula that rewards word and phrase overlap with the question and its expanded query words. Results are ranked.
- **Parsing segments**: CONTEX parses the top-ranked 300 sentences.
- **Pinpointing**: Each candidate answer sentence parse tree is matched against the parse of the question; sometimes also the preceding sentence. As a fallback the window method is used.
- **Ranking of answers**: The candidate answers' scores are compared and the winner(s) are output.

Webclopedia classifies desired answers by their semantic type, using the approx. 140 classes developed in earlier work on the project (Hovy et al., 2000). These types include common semantic classes such as PROPER-PERSON, EMAIL-ADDRESS, LOCATION, and PROPER-ORGANIZATION, but also classes particular to QA such as WHY-FAMOUS, YES:NO, and ABBREVIATION-EXPANSION. They have been taxonomized as the Webclopedia QA Typology, of which an older version can be found at http://www.isi.edu/natural-language/projects/webclopedia/Taxonomy/taxonomy_toplevel.html.

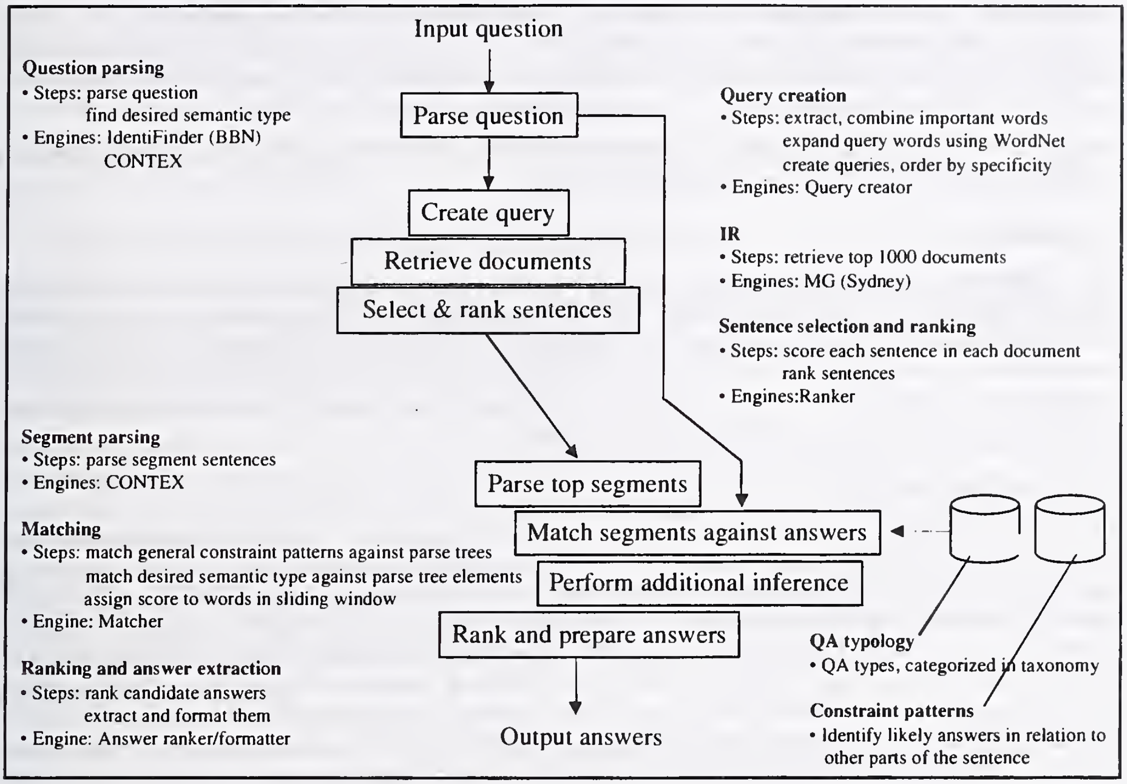

Input question
Question parsing
• Steps: parse question
find desired semantic type
• Engines: IdentiFinder (BBN)
CONTEX
Parse question
Query creation
• Steps: extract, combine important words
expand query words using WordNet
create queries, order by specificity
• Engines: Query creator
Create query
Retrieve documents
Select & rank sentences
IR
• Steps: retrieve top 1000 documents
• Engines: MG (Sydney)
Sentence selection and ranking
• Steps: score each sentence in each document
rank sentences
• Engines:Ranker
Segment parsing
• Steps: parse segment sentences
• Engines: CONTEX
Parse top segments
Matching
• Steps: match general constraint patterns against parse trees
match desired semantic type against parse tree elements
assign score to words in sliding window
• Engine: Matcher
Match segments against answers
Perform additional inference
Rank and prepare answers
Ranking and answer extraction
• Steps: rank candidate answers
extract and format them
• Engine: Answer ranker/formatter
Output answers
QA typology
• QA types, categorized in taxonomy
Constraint patterns
• Identify likely answers in relation to other parts of the sentence

Figure 1. Webclopedia architecture.

2. Parsing

CONTEX is a deterministic machine-learning based grammar learner/parser that was originally built for MT (Hermjakob, 1997). For English, parses of unseen sentences measured 87.6% labeled precision and 88.4% labeled recall, trained on 2048 sentences from the Penn Treebank. Over the past few years it has been extended to Japanese and Korean (Hermjakob, 2000).

For Webclopedia, CONTEX required two extensions. First, its grammar had to be extended to include question forms. The grammar learner portion of CONTEX was trained on approx. 1150 questions and achieved accuracies of approx. 89% labeled precision and labeled recall (Hermjakob, 2001). Second, the grammar had to be augmented to recognize the semantic type of the desired answer (which we call the *qtarget*). Its semantic type ontology was extended to include currently abourt 140 qtarget types, plus some combined types (Hermjakob, 2001). Beside the qtargets that refer to semantic concepts, qtargets can also refer to part of speech labels (e.g., S-PROPER-NAME) and to constituent roles or slots of parse trees (e.g., [ROLE REASON]). For questions with the Qtargets Q-WHY-FAMOUS, Q-WHY-FAMOUS-PERSON, Q-SYNONYM, and others, the parser also provides *qargs*—information helpful for matching:

Who was Betsy Ross? QTARGET: Q-WHY-FAMOUS-PERSON QARGS: ("Betsy Ross")

How is "Pacific Bell" abbreviated? QTARGET: Q-ABBREVIATION QARGS: ("Pacific Bell")

What are geckos? QTARGET: Q-DEFINITION QARGS: (("geckos" "gecko") ("animal"))

These qtargets are determined during parsing using approx. 300 hand-written rules.

3. Document Retrieval and Sentence Ranking

Analyzing the Question to Create a Query

We parse input questions using CONTEX (Section 2) to obtain a semantic representation of the questions. For example, we determine that the question "How far is it from Denver to Aspen?" is asking for a distance quantity. The question analysis module identifies noun phrases, nouns, verb phrases, verbs, adjective phrases, and adjectives embedded in the question. These phrases/words are assigned significance scores according to the frequency of their type in our question corpus (a collection of 27,000+ questions and answers), secondarily by their length, and finally by their significance scores, derived from word frequencies in the question corpus.

We remain indebted to BBN for the use of IdentiFinder (Bikel et al., 1999), which isolates proper names in a text and classifies them as person, organization, or location.

Expanding Queries

Query expansion comes from two sources and used in different stages. In the document retrieval stage, the highly relevant question terms (identified by CONTEX) are expanded in order to boost recall, for example going from "Russian" to "Soviet" or from "capital of the United States" to "Washington". In the sentence ranking stage, we use WordNet 1.6 (Fellbaum, 1998) to match expanded query terms. Although these expanded terms contribute to the final score, their contribution is discounted. This application of expansion strategy aims to achieve high precision and moderate recall.

Retrieving Documents

We use MG (Witten et al., 1994) as our search engine. Although MG is capable of performing ranked query, we only use its Boolean query capability. For the entire TREC-10 test corpus, the size of the inverse index file is about 200 MB and the size of the compressed text database is about 884 MB. The stemming option is turned on. Queries are sent to the MG database, and the retrieved documents are ranked according to their ranking from query analysis. We order queries most specific first, then gradually relax them to more general, until we have retrieved a sufficient number of documents. For example, *(Denver&Aspen)* is sent to the database first. If the number of documents returned is less than a pre-specified threshold, for example, 500, then we retain this set of documents as the basis for further processing, while also submitting the separate queries *(Denver)* and *(Aspen)*.

Ranking Sentences

If the total numbers of sentences contained in the documents returned by MG is N for a given Boolean query, we would like to rank the sentences in the documents to maximize answer recall and precision in the topmost $K << N$, in order to minimize the parsing and subsequent processing. In this stage we set K=300. We assign goodness score to a sentence according to the following criteria:

1. Exact match of proper names such as "Denver" and "Aspen" get 100% bonus score.
2. Upper case term match of length greater than 1 get 60% bonus, otherwise get 30%. For example, match of "United States" is better than just of "United".
3. Lower case matches get the original score.
4. Lower case term match with WordNet expansion stems get 10% discount. If the original term is capital case then it gets 50% discount. For example, when *Cag(e)* matches *cag(e)*, the former may be the last name of some person while the latter is an object; therefore, the case mismatch signals less reliable information.
5. Lower case term matches after Porter stemming get 30% discount. If the original term is capital case then 70% discount. The Porter stemmed match is considered less reliable than a WordNet stem match.
6. Porter stemmer matches of both question word and sentence word get 60% discount. If the original term is capital case then get 80% discount.

7. If CONTEX indicates a term as being QSUBUMED then it gets 90% discount. For example, "Which country manufactures weapons of mass destruction?" where "country" will be marked as *qsubsumed*.

Normally common words are ignored unless they are part of a phrase in question word order. Based on these scores, the total score for a sentence is:

Sentence score = sum of word scores

At the end of the ranking we apply qtarget filtering to promote promising answer sentences. For example, since the question "How far is it from Denver to Aspen?" is asking for a distance quantity, any sentence that contains only "Denver" or "Aspen" but not any distance quantities are thrown out. Only the top 300 remaining sentences are passed to the answer pinpointing module.

The bonus and discount rates given here are heuristics. We are in the process of developing mechanisms to learn these parameters automatically.

4. Answer Matching using Qtarget-Specific Knowledge

Once the candidate answer passages have been identified, their sentences are parsed by CONTEX. The Matcher module then compares their parse trees to the parse tree of the original question. The Matcher performs two independent matches (Hovy et al., 2001; Hovy et al., 2000):

- match qtargets and qargs/qwords in the parse trees,
- match over the answer text using a word window.

Obviously, qtargets and their accompanying qargs play an important role; they enable the matcher to pinpoint within the answer passage the exact, syntactically delimited, answer segment. (In contrast, word window matching techniques, that have no recourse to parse structures, have no accurate way to delimit the exact answer boundaries.)

Unfortunately, there are many questions, for which the qtarget (which can be as generic as NP), syntactic clues and word overlap are insufficient to select a good answer. Over the past year we therefore focused on strategies for dealing with this, and developed the following.

Expected Answer Range

For quantity-targeting questions, humans often have a good sense of reasonable answer ranges and would find it easy to identify the correct answer in the following scenario:

Q: What is the population of New York?

S1. The mayor is held in high regards by the 8 million New Yorkers.

S2. The mayor is held in high regards by the two New Yorkers.

Even without any knowledge about the population of specific cities and countries, a population of 8,000,000 makes more sense than a population of 2. We mirror this 'common sense' knowledge by biasing quantity questions like the one above towards normal value ranges.

Abbreviation Knowledge

Multi-word expressions are not abbreviated arbitrarily:

Q: What does NAFTA stand for?

S1. This range of topics also includes the North American Free Trade Agreement, NAFTA, and the world trade agreement GATT.

S2. The interview now changed to the subject of trade and pending economic issues, such as the issue of opening the rice market, NAFTA, and the issue of Russia repaying economic cooperation funds.

After Webclopedia identifies the qtarget of the question as I-EN-ABBREVIATION-EXPANSION, the system extracts possible answer candidates, including "North American Free Trade Agreement" from S1 and "the rice market" from S2. Based on the perfect match of the initial letters of the first candidate with the acronym NAFTA, an acronym evaluator easily prefers the former over the latter candidate.

Semantic Mark-Up in Parse Trees

Phone numbers, zip codes, email addresses, URLs, and different types of quantities follow patterns that can be exploited to mark them up, even without any explicit mentioning of key words like "phone number". For a question/sentence candidate pair like

Q: What is the zip code for Fremont, CA?

S1. From Everex Systems Inc., 48431 Milmont Drive, Fremont, CA 94538.

Webclopedia identifies the qtarget as C-ZIP-CODE. To match such qtargets, the CONTEX parser marks up (likely) zip codes, based on both structure (e.g., 5 digits) and context (e.g., preceding state code). Two more question/answer pairs that are matched this way:

Q: What's Dianne Feinstein's email address?

Qtarget: C-EMAIL-ADDRESS

S1. Comments on this issue should be directed to Dianne Feinstein at senator@feinstein.senate.gov

Q: How hot is the core of the earth?

Qtarget: I-EN-TEMPERATURE-QUANTITY

S1. The temperature of Earth's inner core may be as high as 9,000 degrees Fahrenheit (5,000 degrees Celsius).

Using External Glosses for Definition Questions

We have found a 10% increase in accuracy in answering definition questions by using external glosses.

Q: What is the Milky Way?

Candidate 1: outer regions

Candidate 2: the galaxy that contains the Earth

For the above question, Webclopedia identified two leading answer candidates. Comparing these answer candidates with the gloss that the system finds in Wordnet:

Wordnet: Milky Way—the galaxy containing the solar system

Webclopedia biases the answer to the candidate with the greater overlap, in this case clearly "the galaxy that contains the Earth".

Finding Support For a Known Answer

It seems against all intuition that a question like

Q1: What is the capital of the United States?

initially poses great difficulties for a question answering system. While a question like

Q2: What is the capital of Kosovo?

can easily be answered from text such as

S2. ... said Mr Panic in Pristina, the capital of Kosovo, after talks with Mr Ibrahim Rugova ...

many American readers would find a newspaper sentence such as

S1. Later in the day, the president returned to Washington, the capital of the United States.

almost insulting. The fact that Washington is the capital of the United States is too basic to be made explicit. In this unexpectedly difficult case, we can fall back on sources like Wordnet:

Wordnet: Washington—the capital of the United States

which, as luck would have it, directly answers our question. Based on this knowledge, Webclopedia produces the answer, represented as a lexical target (LEX "Washington"), which the IR module then uses to focus its search on passages containing "Washington", "capital" and "United States". The matcher then limits the search space to "Washington". The purpose of this exercise is not as ridiculous as it might first appear: even though the system already knows the answer before consulting the document collection, it makes a contribution by identifying documents that support "Washington" as the correct answer.

Semantic Relation Matching in Webclopedia

In question answering, matching words and groups of words is often insufficient to accurately score an answer. As the following examples demonstrate, scoring can benefit from the correct matching of semantic relations in addition:

Question 110: Who killed Lee Harvey Oswald?
Qtargets: I-EN-PROPER-PERSON&S-PROPER-NAME, I-EN-PROPER-ORGANIZATION (0.5)

S1. Belli's clients have included **Jack Ruby**, who killed John F. Kennedy assassin Lee Harvey Oswald, and Jim and Tammy Bakker. [Score: 666.72577; 07/25/90; LA072590-0163]

S2. On Nov. 22, 1963, the building gained national notoriety when Lee Harvey Oswald allegedly shot and killed **President John F. Kennedy** from a sixth floor window as the presidential motorcade passed. [Score: 484.50128; 10/31/88; AP881031-0271]

Note: Answer candidates are bold ("red"), while constituents with corresponding words in the question are underlined ("blue") (http://www.isi.edu/natural-language/projects/webclopedia/sem-rel-examples.html).

Both answer candidates S1 and S2 receive credit for matching "Lee Harvey Oswald" and "kill", as well as for finding an answer (underlined) of the proper type (I-EN-PROPER-PERSON), as determined by the qtarget. However, is the answer "Jack Ruby" or "President John F. Kennedy"? The only way to determine this is to consider the semantic relationship between these candidates and the verb "kill", for which Webclopedia uses the following question and answer parse trees (simplified here):

```
[1] Who killed Lee Harvey Oswald?  [S-SNT]
  (SUBJ) [2] Who  [S-INTERR-NP]
    (PRED) [3] Who  [S-INTERR-PRON]
  (PRED) [4] killed  [S-TR-VERB]
  (OBJ) [5] Lee Harvey Oswald  [S-NP]
    (PRED) [6] Lee Harvey Oswald  [S-PROPER-NAME]
      (MOD) [7] Lee  [S-PROPER-NAME]
      (MOD) [8] Harvey  [S-PROPER-NAME]
      (PRED) [9] Oswald  [S-PROPER-NAME]
  (DUMMY) [10] ?  [D-QUESTION-MARK]
```

```
[1] Jack Ruby, who killed John F. Kennedy assassin Lee Harvey Oswald  [S-NP]
  (PRED) [2] <Jack Ruby>1  [S-NP]
  (DUMMY) [6] ,  [D-COMMA]
  (MOD) [7] who killed John F. Kennedy assassin Lee Harvey Oswald  [S-REL-CLAUSE]
    (SUBJ) [8] who<1>  [S-INTERR-NP]
    (PRED) [10] killed  [S-TR-VERB]
    (OBJ) [11] John F. Kennedy assassin Lee Harvey Oswald  [S-NP]
      (PRED) [12] John F. Kennedy assassin Lee Harvey Oswald  [S-PROPER-NAME]
        (MOD) [13] John F. Kennedy  [S-PROPER-NAME]
        (MOD) [19] assassin  [S-NOUN]
        (PRED) [20] Lee Harvey Oswald  [S-PROPER-NAME]
```

For S1, based on these parse trees, the matcher awards additional credit to node [2] (Jack Ruby) for being the logical subject of the killing (using anaphora resolution) as well as to node [20] (Lee Harvey Oswald) for being the head of the logical object of the killing. Note that—superficially—John F. Kennedy appears to be closer to "killed", but the parse tree correctly records that node [13] is actually not the object of the killing. The candidate in S2 receives no extra credit for semantic relation matching.

Robustness

It is important to note that the Webclopedia matcher awards extra credit for *each* matching semantic relationship between two constituents, not only when everything matches. This results in robustness that comes in handy in cases such as:

Question 268: Who killed Caesar?
Qtargets: I-EN-PROPER-PERSON&S-PROPER-NAME, I-EN-PROPER-ORGANIZATION (0.5)

S1. This version of the plot to kill Julius Caesar is told through the eyes of **Decimus Brutus**, the protege whom Caesar most trusted and who became one of his assassins.
[Score: 284.945; 93/05/15; FT932-8961]

S2. Having failed to prevent Cleopatra's henchwoman Ftatateeta from killing **Pothinus**, Caesar lets Rufius—the new governor of Egypt—murder her, before turning his back on the lot of them in a devastating display of political indifference. [Score: 264.30093; 92/02/06; FT921-10331]

In S1, the matcher gives points to Caesar for being the object of the killing, but (at least as of now) still fails to establish the chain of links that would establish Brutus as his assassin. The predicate-object credit however is enough to make the first answer score higher than in S2, which, while having all agents right next to each other at the surface level, receives no extra credit for semantic relation matching.

Good Generalization

Semantic relation matching applies not only to logical subjects and objects, but also to all other roles such as location, time, reason, etc. It also applies at not only the sentential level, but at all levels:

Question 248: What is the largest snake in the world?
Qtargets: I-EN-ANIMAL

S1. **Reticulated pythons** are the world's largest snakes, reaching lengths of up to 36 feet.
[Score: 384.42365; 12/08/88; AP881208-0148]

S2. The amazing Amazon, the widest, wettest and, so National Geographic now affirms, the longest river in the world (4,007 miles, 51 longer than the Nile), boasts the longest snake **the most venomous viper**, the biggest rat, beetle and ant, along with razor-toothed piranhas that can reduce a Brahman steer to raw bones in minutes and electric eels delivering 640 volts, enough to drive a Metro-North commuter train. [Score: 291.98352; 02/29/88; AP880229-0246]

In the S1, [world] receives credit for modifying snake, even though it is the (semantic) head of a post-modifying prepositional phrase in the question and the head of a pre-modifying determiner phrase in the answer sentence. While the system still of course prefers "in the world" over "the world's" on the constituent matching level, its proper relationship to snake (and the proper relationship between "largest" and "snakes", as well as "pythons" and "snakes") by far outweigh the more literal match of "in the world".

Using a Little Additional Knowledge

Additionally, Webclopedia uses its knowledge of the semantic relationships between concepts like "to invent", "invention" and "inventor", so that in example 209, "Johan Vaaler" gets extra credit for being a likely logical subject of "invention", while "David" actually loses points for being outside of the clausal scope of the inventing process in the second case.

Question 209: Who invented the paper clip?
Qtargets: I-EN-PROPER-PERSON&S-PROPER-NAME, I-EN-PROPER-ORGANIZATION (0.5)

S1. The paper clip, weighing a desk-crushing 1,320 pounds, is a faithful copy of **Norwegian Johan Vaaler**'s 1899 invention, said Per Langaker of the Norwegian School of Management.
[Score: 381.0031; 10/09/89; AP891009-0048]

S2. "Like the guy who invented the safety pin, or the guy who invented the paper clip," **David** added. [Score: 236.47534; 07/20/89; LA072089-0033]

Question 3: What does the Peugeot company manufacture?
Qtargets: S-NP, S-NOUN

S1. Peugeot intends to manufacture **10,000 cars** there each year.
[Score: 360.49545; 10/09/89; AP891009-0048]

S2. These include Coca Cola and Pepsico, the US soft drinks giants, Peugeot the French **car** manufacturer, finance companies GE Capital and Morgan Stanley, Nippon Denro, the Japanese steel manufacturer, and the Scotch whisky maker Seagram and United Distillers, the spirits arm of Guinness. [Score: 323.76758; 93/06/25; FT932-902]

In S2, "car" gets credit as a likely logical object of the manufacturing process, and "Peugeot", being recognized as a "manufacturer", is boosted for playing the proper logical subject role. This example shows that particularly when the qtarget doesn't help much in narrowing down the answer candidate space, semantic relation matching can often make the crucial difference in finding the right answer.

5. Experiments and Results

We entered the TREC-10 QA track, and received an overall Mean Reciprocal Rank (MRR) score of 0.435, which puts Webclopedia among the top performers. The average MRR score for the main task is about 0.234. The answer rank distribution is shown in Figure 2. It indicates that we cannot find answers in the top 5 in about 43% of the cases. Once we find answers we usually rank them at the first place.

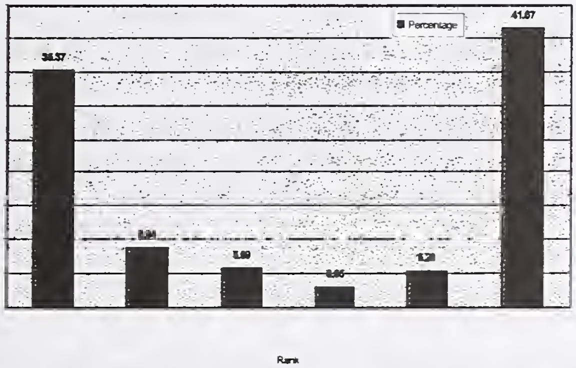


Figure 2. Webclopedia answer rank distribution in TREC-10.

Analysis of the answers returned by the TREC assessors revealed several problems, ranging from outright errors to judgments open to interpretation. One example of an error is a ruling that the answer to "what is cryogenics?" is not "engineering at low temperature" (as defined for example in Webster's Ninth New Collegiate Dictionary, and as appears in the TREC collection), but rather the more colloquial "freezing human being for later resustication" (which also appears in the collection). Although Webclopedia returned both, the correct answer (which it preferred) was marked wrong. While we recognize that it imposes a great administrative burden on the TREC QA administrators and assessors to re-evaluate such judgments, it is also clearly not good R&D methodology to train systems to produce answers that are incorrect but colloquially accepted. (Checking whether their knowledge is correct is precisely one of the reasons people need QA systems!) We therefore propose an appeals procedure by which the appellant must provide to the administrator the question, the correct answer, and proof, drawn from a standard reference work, of correctness. The administrator can provide a list of acceptable reference works beforehand, which should include dictionaries, lists of common knowledge facts (the seven wonder of the world, historical events, etc.), abbreviation lists, etc., but which would presumably not include local telephone books, etc. (thereby ruling out local restaurants as answer to "what is the Taj Mahal?").

6. References

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.

Fellbaum, Ch. (ed). 1998. *WordNet: An Electronic Lexical Database*. Cambridge: MIT Press.

Hermjakob, U. 1997. *Learning Parse and Translation Decisions from Examples with Rich Context.* Ph.D. dissertation, University of Texas at Austin. file://ftp.cs.utexas.edu/pub/ mooney/papers/hermjakob-dissertation-97.ps.gz.

Hermjakob, U. 2000. Rapid Parser Development: A Machine Learning Approach for Korean. In *Proceedings of the North American chapter of the Association for Computational Linguistics* (NAACL-2000). http://www.isi.edu/~ulf/papers/kor_naacl00.ps.gz.

Hermjakob, U. 2001. Parsing and Question Classification for Question Answering. In *Proceedings of the Workshop on Question Answering at the Conference ACL-2001.* Toulouse, France.

Hovy, E.H., L. Gerber, U. Hermjakob, M. Junk, and C.-Y. Lin. 2000. Question Answering in Webclopedia. *Proceedings of the TREC-9 Conference*. NIST. Gaithersburg, MD.

Hovy, E.H., U. Hermjakob, C.-Y. Lin, and D. Ravichandran. 2001. Toward Semantics-Based Answer Pinpointing. *Proceedings of the Human Language Technologies Conference* (HLT). San Diego, CA.

Witten, I.H., A. Moffat, and T.C. Bell. 1994. Managing Gigabytes: Compressing and indexing documents and images. New York: Van Nostrand Reinhold.

Selecting versus Describing:
A Preliminary Analysis of the Efficacy of Categories in Exploring the Web

E.G. Toms[1a], R.W. Kopak[2], J. Bartlett[1], L. Freund[1]

[1]Faculty of Information Studies, University of Toronto,
Toronto, Ontario, Canada
[2]School of Library, Archival and Information Studies, University of British Columbia,
Vancouver, British Columbia, Canada

[a] Person to whom all correspondence should be sent: toms@fis.utoronto.ca

This paper reports the findings of an exploratory study carried out as part of the Interactive Track at the 10th annual Text Retrieval Conference (TREC). Forty-eight, non-expert participants each completed four Web search tasks from among four specified topic areas: shopping, medicine, travel, and research. Participants were given a choice of initiating the search with a query or with a selection of a category from a pre-defined list. Participants were also asked to phrase a selected number of their search queries in the form of a complete statement or question. Results showed that there was little effect of the task domain on the search outcome. Exceptions to this were the problematic nature of the Shopping tasks, and the preference for query over category when the search task was general, i.e. when the semantics of the task did not map directly onto one of the available categories. Participants also evidenced a reluctance/inability to phrase search queries in the form of a complete statement or question. When keywords were used, they were short, averaging around two terms per query statement.

Introduction

We are working toward improved search interfaces. We have observed that to date multiple approaches to the search processes have been suggested [1,3,4,5], but these discuss the search process at a macro level, offering guidance and orientation on how that task may be implemented{2]. At the 'keystroke' level – the point of interaction with the system, the search task is procedural; commands are entered and responses received. Missing from the literature to date is an understanding of component steps used to perform the search task at that micro level. In our work, we are taking a holistic approach to how interfaces might be designed to facilitate information searching, browsing and encountering. As a first step, we are observing how non-experts seek information on the World Wide Web (the 'Web'), noting in particular their mode of interaction with the system.

In this exploratory study we compared how participants used pre-defined categories versus standard search statements. By doing so, we hope to understand the interplay between browsing and searching while in the process of information seeking. In addition, we also examined participant behaviour across three additional factors: the way a search was entered (as question or as keyword), by the source of the task (researcher-specified versus user-personalized), and by task domain (medicine, travel, shopping, and research). We assessed the outcomes among these factors using a series of efficiency, effectiveness, and satisfaction metrics. We added verbal protocol data so that we could better understand the reasoning behind participants' use of category and string searching, and to provide a rich description of strategies used and rationales for observed patterns. In this version of our analysis, only an analysis of quantitative data is included.

Method

Participants

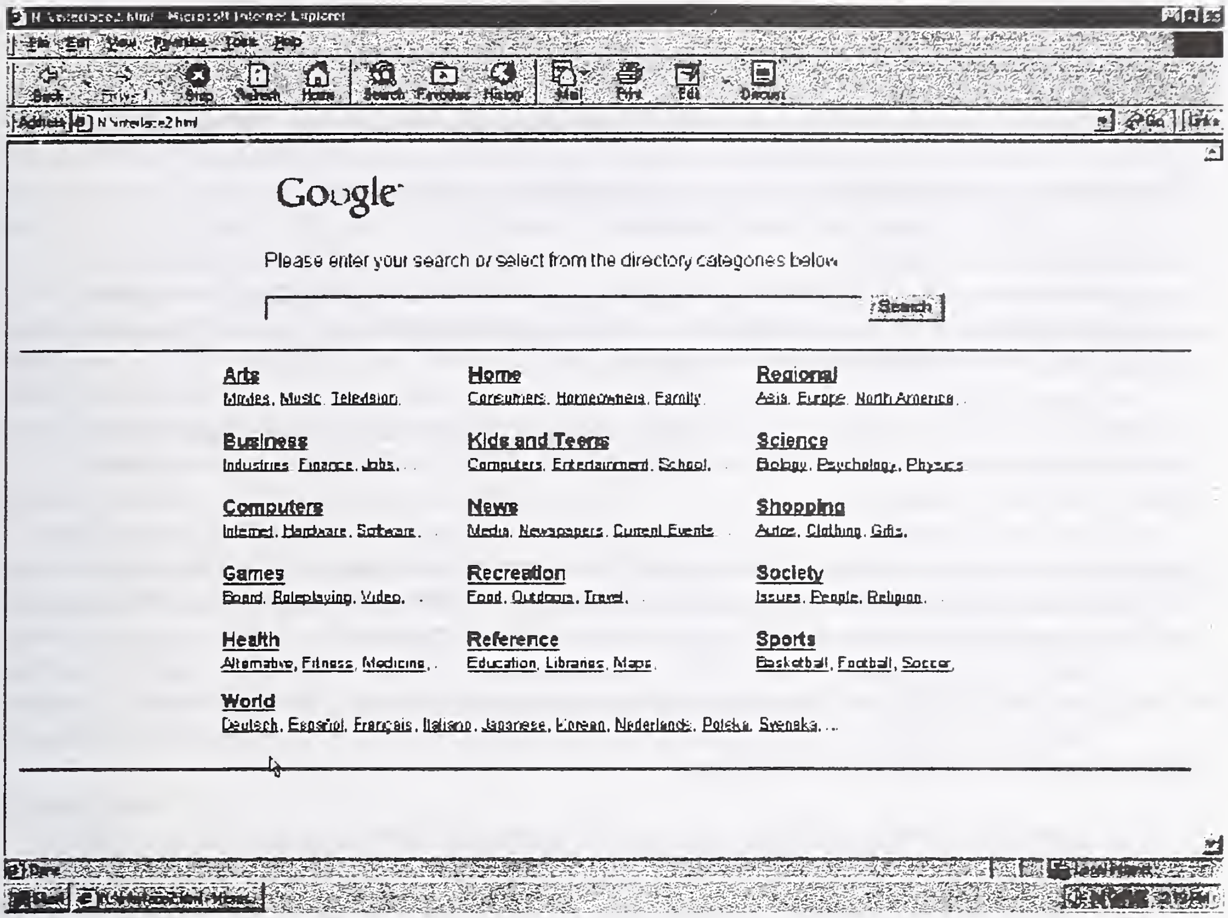
Our criteria for selection specified that participants be adult members of the general public (including but not limited to the university community) who have used the Web, who may have searched the Web previously, and who may have had some training, but who had not taken professional search courses. Information science/studies students were eligible only if they were in first term, and had not yet taken a professional search course. The sample was one of convenience. Participants were recruited by printed posters posted on bulletin boards on campus, or in libraries and coffee shops in the surrounding area, and via e-mail posted on listservs or e-notice boards at the Universities of Toronto and British Columbia.

The 48 participants (29 women and 19 men) ranged in age from 18-20 to over 65 years; 80% were under 35. Most had university level education, mainly at the bachelor (38%) or masters (30%) level, predominantly from the humanities or social sciences. About half were students; the remainder were from a diverse range of occupations. Most (94%) of the participants had been using the Internet for more than two years, frequently using it for 6 or more hours (50%) per week. Email was the most frequently used application, with all but one person using it daily. All but one participant reported searching the web on a daily or weekly basis. Almost all had no search training of any sort. Overall, they were a relatively young, educated group who were experienced in terms of web use.

Search Interface

We used Google as our Web search engine, and modified the standard Google interface to include both the search box/button, and the Google top level category list (directory). The resulting screen retained Google's simplicity. An instruction to either enter a query in the search box or select a category from the directory was added (See Figure 1). Beyond this initial page, the standard Google interface screens were retained.

Choice of Google as the search engine was based on its current status as the most popular search engine (http://www.searchenginewatch.com/reports/perday.html). Like many search engines, Google accepts natural language queries, joining terms with AND by default. Google uses a stop list, and displays to the user terms which were eliminated from the search. Words such as the questions' terms (who, what, where, when, why) and many other common words seem to appear on that stoplist. A query seems to be limited to ten non-stop word terms and to not be stemmed.


Google
Search
Arts
Movies, Music, Television, ...
Home
Consumers, Homeowners, Family, ...
Regional
Asia, Europe, North America, ...
Business
Industries, Finance, Jobs, ...
Kids and Teens
Computers, Entertainment, School, ...
Science
Biology, Psychology, Physics, ...
Computers
Internet, Hardware, Software, ...
News
Media, Newspapers, Current Events, ...
Shopping
Autos, Clothing, Gifts, ...
Games
Board, Roleplaying, Video, ...
Recreation
Food, Outdoors, Travel, ...
Society
Issues, People, Religion, ...
Health
Alternative, Fitness, Medicine, ...
Reference
Education, Libraries, Maps, ...
Sports
Basketball, Football, Soccer, ...
World
Deutsch, Español, Français, Italiano, Japanese, Korean, Nederlands, Polska, Svenska, ...

Please enter your search or select from the directory categories below

Figure 1. Modified Google interface

Tasks

Sixteen tasks (devised by the TREC 10 Interactive Track participants) were used in the study. The questions came from four domains: Medical, Research, Travel and Shopping. Of the 16 tasks, half were fully specified and half were partially specified so that participants could personalize them.

Procedure

The participants were recruited in August and September of 2001 in Toronto and Vancouver. Each participant was given four search tasks, one from each of the four domains. Two of the four tasks contained specific questions or imperatives to which the participant was to respond by finding relevant Web pages. For the remaining two tasks, participants were asked to provide a topic of personal interest, but within the general topic domain pertaining to the task, e.g. Medicine. We used a modified latin squares method to distribute the question variations among the participants.

We also used two different sets of search instructions. For the first two topics, we asked the participant to either enter the query as a list of one or more words or phrases, or to select a category from the directory. For the last two topics, we asked the participant to either enter the query as a *complete* question or sentence, or to select a category from the directory.

Each participant session lasted approximately two hours. During this time, participants

- completed a demographics and web/search experience questionnaire.

- were assigned four search tasks in sequence. For each task the participant completed a pre-search questionnaire to establish their familiarity with the topic, searched for the topic using the Web interface, and responded to a post-search questionnaire about the search process and their satisfaction with the results.

- described their search while a screen-capture video of the search was replaying. During these retrospective interviews, we tried to elicit the decision-making process used at each stage in the search process.

- responded to a series of questions regarding the search process as a whole. This final interview, which lasted about 10 minutes, was intended to get the participant to comment in a personal way on their personal challenges when searching the Web.

This paper reports, primarily, on the results of the data collected in steps 1 and 2 above.

Data was collected using four mechanisms:

1. Questionnaires for demographics, and pre- and post-search evaluations.

2. Audio-tape for all semi-structured interviews.

3. Transaction logs; the *WinWhatWhere* software used captured the titles and URLs of all sites visited, and all keystrokes entered.

4. Screen capture to visually record the user process; Lotus *Screencam* software records in real-time each user session and stores it for playback.

Data Analysis

Data from the pre- and post-search questionnaires and the demographics survey data were combined with data from the transaction logs. Because of the way that WinWhatWhere outputted data, we manually coded the search state, such as query use, category selection, hit list·selection, URL, viewing and so on, by reviewing the ScreenCam files and the WinWhatWhere files together. The additional coding made it possible to identify the path taken in each search, to determine the amount of time spent at each state, and to identify the rank position on a hit list page of each selected URL. In addition, audio-tapes were transcribed and the content is being analyzed (but is not included in this report).

Results and Discussion

Summary of Results

The 48 participants spent about 7 minutes doing each task. They used the search box for about 66% of the tasks and selected from the directory categories for the remainder. On average, they examined about 5 URLs and about 6 links within each of those URLs. They tended to select about the fourth item on a hitlist and on average examined about two pages of hitlists.

Participants reported little familiarity with the topics for each of the assigned tasks, with few having ever done a search on any of the topics prior to the session. On a five-point scale with one being the poorest rating and five being the best rating, they indicated the degree of certainty with which they found their answer, the ease of finding the answer, and their satisfaction with the process of finding their answer at around four.

User-Specified vs. Researcher Specified Task

Half the questions were completely specified and half were fill-in-the-blanks, allowing some user modification toward personalizing the task. There were no significant differences between the two types on any measure. This finding challenges the assumption that information retrieval experimentation with pre-defined queries alters user behaviour in experimental settings. Our participants performed about the same regardless of whether they were assigned a task or allowed to create their own. That said, it is likely that the artificially of the process, e.g., time constraints, lab setting, and so on, may have a greater impact than the nature of the task.

Tactic Used

Participant search paths were analyzed according to the strategy taken in finding information. To start, they could have elected to use a query or a category, and could have changed that tactic to the other technique at any time during the process. Some participants, for example, used a single tactic such as queries only, while some used novel strategies that combined queries and categories as illustrated below:

Strategy	N	Code used in Table 1
used queries only	104	Qry
used categories only	25	Cat
used queries and then selected categories	24	Qry -> Cat
used categories and then selected queries	39	Cat -> Qry

A caveat of this result is the effect of an inherent bias towards the query. While the initial start page contained both categories and a query box, once a query was used, the categories had to be sought out. On the other hand, the query box is an integral part of the second and subsequent category pages, appearing at the top of each, and on each hitlist page.

Twenty efficiency, effectiveness, and satisfactory metrics assessed the strategies used by participants. Most of this data was derived from the transaction logs or self-reported by the participant in pre- and post task questionnaires. Results from analyses of variance for each of the measures appear in Table 1.

Table 1. Results on all measures by approaches used in the search

	Strategies Used				
Metric	**Qry -> Cat**	**Cat -> Qry**	**Qry**	**Cat**	**Statistical Significance**
Average Number of Instances of Each Search State Per Task					
# of Queries	2.3	.72	1.9	.0	F(3,192)=18.758, p<.001
# of Categories	4.2	5.3	.05	5.9	F(3,192)=35.236, p<.001
# of URLs	6.8	5.4	4.4	3.1	F(3,192)=3.922, p=.010
# of Print	1.8	1.9	2.2	1.5	F(3,192)=2.862, p=.038
# of HitLists	7.9	6.6	6.0	1.0	F(3,192)=9.719, p<.001
# of in Site links	5.8	6.6	6.4	6.2	*ns*
Average Time (seconds) Spent at Each Search State Per Task					
Query Time	29.0	36.2	21.3	.88	**
Hit List Time	120.9	145.9	144.4	41.2	**
URL Time	113.3	114.1	98.5	77.0	*ns*
Category Time	123.8	67.2	1.3	203.5	F(3,188)=5.830, p=.001
Print Time	170.9	84.3	207.8	114.5	**
In Site Time	109.3	138.6	116.8	122.1	*ns*
Position of URLs in HitList					
Avg. Rank	4.7	4.2	4.4	4.8	*ns*
Min Rank	1.7	1.9	2.3	3.4	*ns*
Max Rank	8.6	7.1	6.7	6.9	*ns*
Average User Perception Rating Per Task (scale from 1 to 5)					
Familiarity	2.7	2.4	2.3	2.2	*ns*
Certainty	3.7	3.5	4.0	4.1	F(3,190)=2.901, p=.036
Ease	3.4	3.2	3.9	4.0	F(3,190)=4.543, p=.004
Time allotted	3.6	3.1	3.5	3.8	F(3,190)=2.883, p=.037
Satisfaction	3.4	3.3	3.7	4.0	F(3,190)=2.593, p=.054

There were two key differences in the strategy data. There was a key distinction between the two single-tactic strategies and between the single and mixed strategies. Participants who used only Categories looked at significantly fewer hit lists and spent less time looking at those lists than those who used only Queries. The use of categories seems to have led participants to sites that were more specifically related to the topic, while those who used Queries seemed to examine more pages of hitlists and spent more time doing so. But there were no user perception differences between those who used the single tactic strategies. Both query only and category only participants were equally satisfied with the task and with the ease with which the task was completed.

Participants who chose the single tactic strategies i.e., categories- or queries-only, tended to find it easier and more satisfying, and were more certain about their results than when using mixed approaches. In addition, participants who use a single tactic seemed to be more successful.

Those who used mixed tactics felt the task was more difficult and seemed less satisfied than those using the single tactic strategies. Mixed tactic users also, selectedly, scored lower on some of the count and time efficiency measures.

Search as Question vs. Search as Keyword

Participants were required to carry out half the tasks using a question or statement and half using keywords or phrases. Because of the interface, they could, however, choose to use the search box or select from the categories. In general participants used the searchbox 66% of the time, but the selection from categories versus use of the searchbox was clearly related to the way that participants were required to enter the query. When asked to search with a question, the use of categories increased significantly ($?^2$=6.0, p=.014).

Table 2. Participants first tactic by the type of search entry

		Search Entered		
		in question form	as keyword(s)	Total
Start With:	**Directory**	40	24	64
	Searchbox	56	72	128
	Total	96	96	192

Participants examined fewer hitlists (4.3 and 6.3, respectively) when using categories than when using a searchbox (F(1,192)=161.461, p=.017). In addition, when asked to provide a question, they tended to provide keywords or phrases for a significant number of the question-based queries (F(2,189)=3.844, p=.023). Participants also tended to rate the task-as-question as more difficult that the task-as-keyword (F(2,189)=5.986, p=.015). We believe that participants were challenged by this task as it did not represent the way that they normally conceptualize the search process. Thus to avoid asking a question, they opted for categories.

Additionally, those who asked questions tended to create longer queries – from 2.6 to 5.8 words (F(2,165)=421.469, p<.001). But the increase in size was accounted for primarily by stopwords (F(1, 166)=65.663, p<.001). Queries as questions had approximately 3.7 stopwords while those entered as keywords had on average about 2.5 stopwords.

Type of Task

Four different task domains were used in this study: Medicine, Research, Shopping and Travel. Results for various measures across each task domain appear in Table 3. There are few significant differences among the four domains.

There were however differences in post hoc Bonferroni adjusted tests. Participants did more printing in Research than Shopping (p=.035) and spent less time in Categories while responding to a Research task than a Travel task (p=.010). Research was perhaps the most complex and cognitively challenging task. Categories were rarely used for Research tasks, and participants spent little time examining categories while doing them. However, Categories were used almost identically across the other three tasks. We can speculate that the presence of top level categories that were semantically related to the assigned task made it easier to use in Shopping, Medicine and Travel queries, and, additionally, that those tasks were more specific than the Research task.

The type of task had an effect on user perception. In general participants found the Shopping task more difficult and less satisfying than the other tasks, rating these on average between 3.1 and 3.3 on a five-point scale.

Table 3. Various Metrics across Type of Task

		Type of Task				
Metric	**Mean**	**Medical**	**Shopping**	**Travel**	**Research**	**Statistical Significance**
Average Number of Instances of Each Search State Per Task						
# of Queries	1.7	1.3	1.7	1.5	2.0	*ns*
# of Categories	2.4	1.1	2.4	2.8	2.2	*ns*
# of URLs	4.7	4.2	4.7	5.0	5.0	*ns*
# of Print	2.0	2.0	1.6	2.2	2.2	**
# of HitLists	5.7	5.4	5.3	5.2	6.8	*ns*
# of In-Site links	6.4	3.8	8.0	8.2	5.5	*ns*
Average Time Spent at Each Search State Per Task						
Query Time	22.6	15.5	24.1	29.4	21.6	*ns*
Hit List Time	128.3	144.3	107.2	100.8	161.0	*ns*
URL Time	100.7	117.2	90.9	87.7	107.0	*ns*
Category Time	56.3	15.8	51.2	99.5	58.9	*ns*
Print Time	166.0	174.1	129.2	205.0	155.6	*ns*
In site Time	121.0	70.3	156.0	159.5	98.1	*F(3,192)=5.072, p=.002*
Position of URLs in HitList						
Avg. Rank	4.4	4.0	4.9	4.4	4.5	*ns*
Min Rank	2.3	2.0	2.7	2.3	2.1	*ns*
Max Rank	7.0	6.5	7.7	7.0	7.0	*ns*
Average User Perception Rating Per Task						
Familiarity	2.4	2.5	2.2	2.1	2.6	**
Certainty	3.9	4.2	3.7	4.0	3.8	**
Ease	3.7	3.9	3.3	3.8	3.7	**
Time allotted	3.5	3.7	3.2	3.5	3.5	**
Rating	3.6	3.9	3.1	3.7	3.8	*F(3,189)=5.191, p=.002*

** selected results were significant in Bonferroni adjusted post hoc tests

In addition, we mapped strategies by domain as illustrated in Figure 2. The mixed strategies are evenly used across the four domains. But when the directory categories were used as a tactic, it tended to be for travel topics.

Discussion and Conclusions

We discovered that researcher-specified versus participant-personalized queries had no effect on results, suggesting that experimental tasks are as effective in experimental settings as user-defined tasks. The domain of the task, too, appears to have had little effect, although the Shopping tasks tended to be more difficult to complete, and were generally the least satisfying.

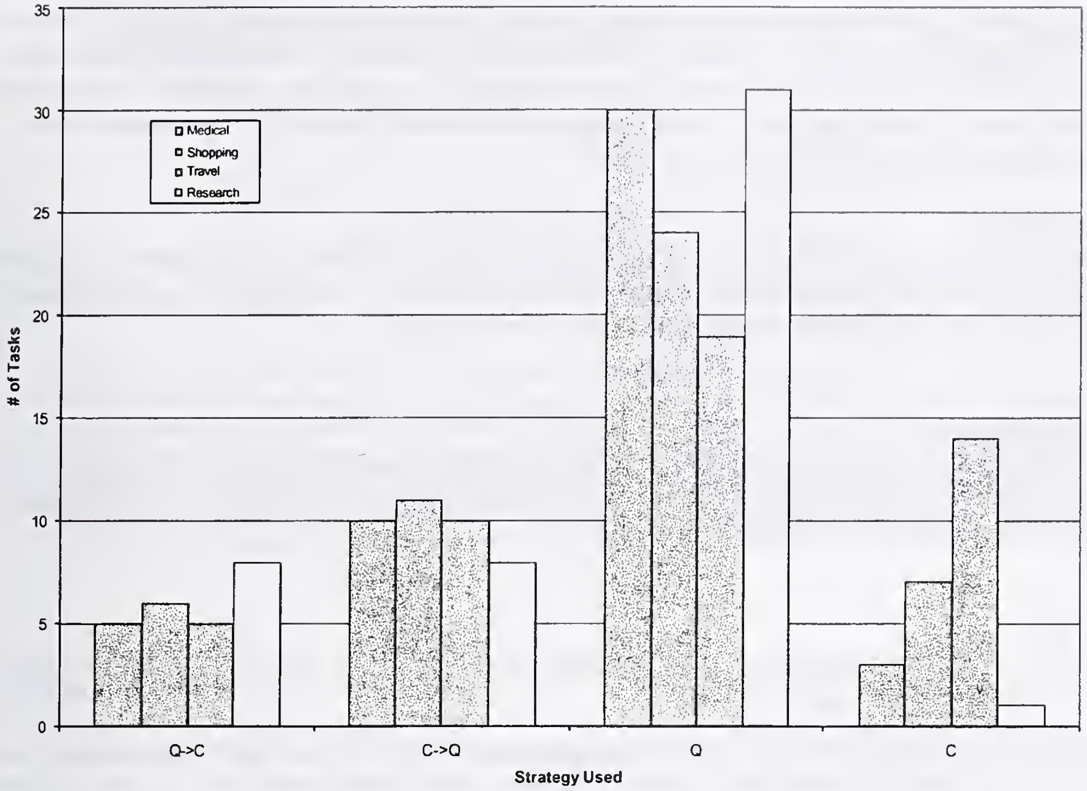


Figure 2. Domain by Strategy Used

The use of categories seems to have influenced the search process itself, where more time was spent contemplating the nature of the search task at the beginning of the process, resulting in fewer items being selected from the hit list, and marginally less navigation within a site once there. Anecdotally, participants indicated a need to develop a broad perspective before focusing on specific results. Once focused, they were able to make clear choices from the hiltlists than those who issued queries.

When participants were asked to express a search statement in the form of a question or statement, they had only modest success. The choice between initiating a search with a search box or with a selection from the categories seems dependent at least partially on the manner in which the query is entered. Participants were more likely to search using the categories when they were requested to create the query as question. It seems the prospect of using a question posed difficulties for participants. When entering keyword queries, the number of keywords used, on average, was quite small. Likely participants have learned one way of conceptualizing the search process and have developed a fixed mental model of that process, which constrained their ability to provide richer search statements.

Future analyses of the data will use the verbal protocol data collected to enhance our interpretation of the current findings. From the examination of this protocol data we hope to gain a richer explanation of not only what was observed in the findings reported here, but of why participants chose the courses of action they did for the various tasks performed. For example, why was the query constructed in that particular manner? What did the participant think it would achieve? Why did they choose to search using categories or queries? How did they select from the results list? And, how did they decide if a site was useful? In addition, we hope to pinpoint the problems within the search process. When participants appeared to be off course, what might have been useful to help get them back on track?

Future Research

Based on this current work, we also hope to carry out two additional studies that will focus on: i) developing a more refined experimental approach to the category and query integrated search, and ii) manipulating how people conceptualize the query process.

Acknowledgments

This work was partially funded by a Natural Sciences and Engineering Research Council of Canada grant to the first author. The authors wish to thank Research Assistants, J. Heaton and A. Olsen in Vancouver and A. Lebowitz in Toronto.

References

[1] Guthrie, J. T. Locating information in documents: examination of a cognitive model. *Reading Research Quarterly,* 23(2 1988, 178-199.

[2] Kuhlthau, C. Inside the search process: information seeking from the user's perspective. *Journal of the American Society for Information Science*, 42(5 1991), 361-71.

[3] Marchionini, G. *Information seeking in electronic environments*. New York: Cambridge University Press, 1995.

[4] Norman, D. A. Cognitive engineering. In D. A. Norman & S. W. Draper (Eds.), *User centered system design: new perspectives on human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1986, 31-61.

[5] Shneiderman, B., Byrd, D & Croft, W.B. Sorting our searching: a user interface framework for text searches. *Communications of the ACM* 41 (4 1998), 95-98.

Retrieving Web Pages using Content, Links, URLs and Anchors

Thijs Westerveld[1], Wessel Kraaij[2], and Djoerd Hiemstra[1]

[1] University of Twente, CTIT, P.O. Box 217, 7500 AE Enschede, The Netherlands
{hiemstra,westerve}@cs.utwente.nl
[2] TNO-TPD, P.O. Box 155, 2600 AD Delft, The Netherlands
kraaij@tpd.tno.nl

Abstract. For this year's web track, we concentrated on the entry page finding task. For the content-only runs, in both the ad-hoc task and the entry page finding task, we used an information retrieval system based on a simple unigram language model. In the Ad hoc task we experimented with alternatieve approaches to smoothing. For the entry page task, we incorporated additional information into the model. The sources of information we used in addition to the document's content are links, URLs and anchors. We found that almost every approach can improve the results of a content only run. In the end, a very basic approach, using the depth of the path of the URL as a prior, yielded by far the largest improvement over the content only results.

1 Introduction

Entry page searching is different from general information searching, not only because entry pages differ from other web documents, but also because the goals of the tasks are different. In a general information search task we're interested in finding as much information as possible, whereas for entry page searches we're looking for one specific document. Therefore, the entry page task is clearly a high precision task. Because of both the differences in the task and in the documents, information sources other then the document's content can be very useful for locating the relevant entry page even though they didn't do much for general information searching.

For the content-only runs, in both the ad-hoc task and the entry page finding task, we used an information retrieval system based on a simple unigram language model. This IR model, which we introduced at the TREC-7 conference [4] and which worked effectively on last year's web task, is presented in section 2. Section 3 describes how we used links, anchors and URLs to improve a content only run and section 4 lists the official results for the submitted runs as well as the results for the additional runs we did. Finally, section 5 lists our conclusions.

2 Basic IR model

All runs were carried out with an information retrieval system based on a simple statistical language model [3]. The basic idea is that documents can be represented by unigram language models. Now, if a query is more probable given a language model based on document d_1, than given e.g. a language model based on document d_2, then we hypothesise that the document d_1 is more relevant to the query than document d_2. Thus the probability of generating a certain query given a document-based language model can serve as a score to rank documents with respect to relevance.

$$P(T_1, T_2, \cdots, T_n|D_k)P(D_k) = P(D_k)\prod_{i=1}^{n}(1-\lambda)P(T_i|C) + \lambda P(T_i|D_k) \qquad (1)$$

Equation 1 shows the basic idea of this approach to information retrieval, where the document-based language model is smoothed by interpolation with a background language model to compensate for sparseness. In the equation, T_i is a random variable for the query term on position i in the query ($1 \leq i \leq n$, where n is the query length), which sample space is the set $\{t^{(0)}, t^{(1)}, \cdots, t^{(m)}\}$ of all terms in the collection. The probability measure $P(T_i|C)$ defines the probability of drawing a term at random from the collection, $P(T_i|D_k)$ defines the probability of drawing a term at random from document k; and λ is the interpolation parameter[1]. The a-priori probability of relevance $P(D_k)$ is usually taken to be a linear function of the document length, modelling the empirical fact that longer documents have a higher probability of relevance.

2.1 Combining external information

The basic ranking model is based on the content of the web pages. There is evidence that other sources of information (link structure, anchor text) play a decisive role in the ranking process of entry pages (e.g. Google[2]). The preferred way to incorporate extra information about web pages is to include this information in the model. A clean method is to incorporate this information in the prior probability of a document. A second manner is to model different types of evidence as different types of ranking models, and combine these methods via interpolation.

$$score_{\text{combi}} = \alpha score_{\text{content}} + (1-\alpha) score_{\text{features}} \tag{2}$$

Equation 2 shows how two ranking functions can be combined by interpolation. The combined score is based on a weighted function of the unigram document model and the posterior probability given the document feature set and a Bayesian classifier trained on the training set. As features we experimented with the number of inlinks and the URL form. However, for interpolation, scores have to be normalised across queries, because the interpolation scheme is query independent. Therefore, for the interpolation method we normalised the content score by the query length, the ranking models based on other document information that we applied are (discriminative) probabilities and thus need no normalisation. The interpolation method has shown to work well in cases where score normalisation is a key factor [6]. For the experiments we describe here, we have applied both methods and they yield similar results. In a context where score normalisation is not necessary, we prefer method one. We determined the document priors (document-content independent prior probabilities) using various techniques, either postulating a relationship, or learning priors from training data conditioning on e.g. the URL form. This process will be described in more detail in the Section 3.

2.2 Smoothing variants

Recent experiments have shown that the particular choice of smoothing technique can have a large influence on the retrieval effectiveness. For title adhoc queries, Zhai and Lafferty [8] found Dirichlet smoothing to be more effective than linear interpolation[3] Both methods start from the idea that the probability estimate for unseen terms: $P_u(T_i|D_k)$ is modelled a constant times the collection based estimate: $P(T_i|C)$. A crucial difference between Dirichlet and Jelinek-Mercer smoothing is that the smoothing constant is dependent on the document length for Dirichlet, reflecting the fact that probability estimates are more reliable for longer documents. Equation (3) shows the weighting formula for Dirichlet smoothing, where $c(T_i|D_k)$ is the term frequency of term T_i in document D_k , $\sum_w c(T_i; D_k)$ is the length of document D_k and μ is a constant. The collection specific smoothing constant is in this case $\frac{\mu}{\sum_w c(T_i; D_k)+\mu}$, whereas the smoothing constant is $(1-\lambda)$ in the Jelinek-Mercer based model.

$$P(T_1, T_2, \cdots, T_n|D_k)P(D_k) \; = \; P(D_k)\prod_{i=1}^{n}\frac{c(T_i; D_k) + \mu P(T_i|C)}{\sum_w c(T_i; D_k) + \mu} \tag{3}$$

3 Entry page Search

To improve the results of a content only run in the entry page finding task, we experimented with various link, URL and anchor based methods. We tested several well-known and novel techniques on the set of 100 training topics provided by NIST and found that each method we tested was more or less beneficial for finding entry pages. This contrasts with last year's findings where link based techniques didn't add anything in an ad hoc search task [7]. In the following subsections, we subsequently discuss link based methods, URL based methods and anchor based methods, along with our findings on the training data.

[1] We apply a simplified version of the model developed in [3], where λ is term specific, denoting the term importance

[2] `http://www.google.com`

[3] Also called Jelinek-Mercer smoothing.

3.1 Links

One of the sources of information one can use in addition to the content is the link structure. This is the structure of hyperlinks connecting the documents on the web. We took two different approaches exploiting this structure, both relying on the fact that entry pages tend to have a different link structure than other documents.

Inlinks The first link-based approach we experimented with is based on the assumption that entry pages tend to have a higher number of inlinks than other documents (i.e. they are referenced more often). A well known example of a commercial search engine which is based on a similar assumption is Google [1]. To check whether this assumption holds, we made a of plot of $P(entrypage|\#inlinks)$ (See Figure 1). The probabilities are estimated on half of the training data. The figure shows that indeed documents with more inlinks tend to have a higher probability of being an entry page. Therefore, for an entry page task, the number of inlinks might be a good prior. In fact, as figure 2 shows, the assumption that longer documents have a higher probability of being relevant does not hold for entry page searches and a prior based on the number of inlinks might be better than one based on the length of the document.

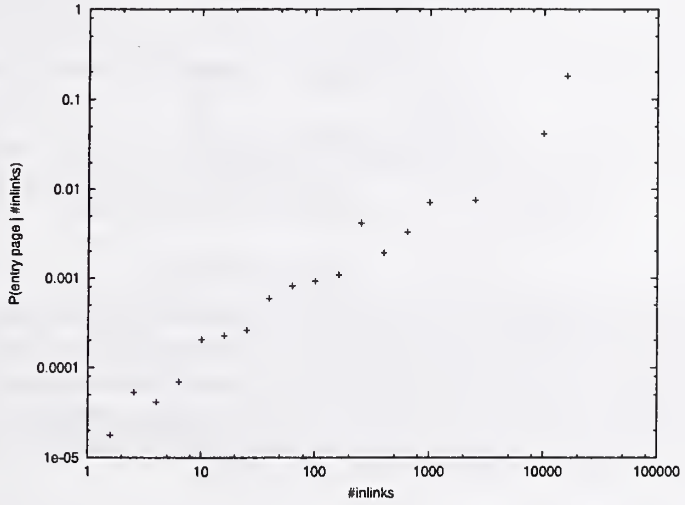


Fig. 1. $P(entrypage \mid \#inlinks)$

As a prior for ad hoc searches, we usually take a document length prior:

$$P(D_k) = \frac{doclen(D_k)}{\sum_{j=1}^{N} doclen(D_j)} \tag{4}$$

We define the inlink prior as:

$$P(D_k) = \frac{\#inlinks(D_k)}{\sum_{j=1}^{N} \#inlinks(D_j)} \tag{5}$$

We compared the two priors of equations 4 and 5 on the training data. We normalised the content score by the query length and interpolated with the inlink prior (cf. eq. 2), the doclen prior is used conform eq. 1. Table 1 shows the mean reciprocal ranks (MRR)[4]. The interpolation parameters used in the table gave the best results. The scores show that indeed, the number of inlinks is a better prior than the length of the document.

[4] The reciprocal of the rank of the relevant entry page averaged over all queries.

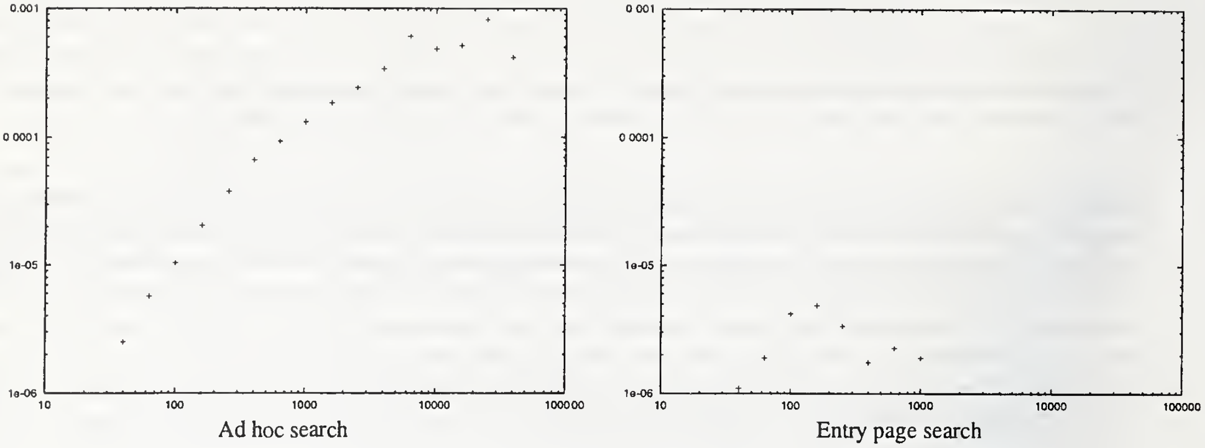


Fig. 2. $P(relevant \mid doclen)$

run	MRR
content	0.26
content + doclen prior	0.21
0.7 * content + 0.3 * inlink	0.38

Table 1. MRRs Inlink and doclen priors on training data

Kleinberg The second link-based approach we experimented with is based on Kleinberg's hub and authority algorithm [5]. This algorithm identifies authorities (important sources of information) and hubs (lists of pointers to authorities) by analysing the structure of hyperlinks. Since entry pages can be seen as authorities on a very specific subject (a certain organisation), Kleinberg's algorithm can be useful for the entry page search task. The algorithm works by iteratively assigning hub and authority scores to documents in such a way that good hubs are pages that refer to many good authorities and good authorities are referenced by many good hubs:

1. Take the top N results from the content run
2. Extend this set S with all documents that are linked to S (either through in or through outlinks)
3. Initialise all hub and authority scores in this set to 1.
4. $hub(D) = \sum_{\{i|link\ D \rightarrow\ i\ exists\}} auth(i)$
5. $auth(D) = \sum_{\{i|link\ i\ \rightarrow\ D\ exists\}} hub(i)$
6. normalise hub and auth scores such that $\sum_{s \in S} hub^2(s) = \sum_{s \in S} auth^2(s) = 1$
7. repeat steps 4 - 6

We computed hubs and authorities for the top N of the content only run and used the resulting authority scores to rank the documents. Table 2 shows the results for different values of N.

As the results show, taking only the top 5 or top 10 ranks from the content run and computing authority scores starting from those, is sufficient to improve the results. Apparently, if an entry page is not in the top 5 from the content run, it *is* often in the set of documents linked to these 5 documents.

3.2 URLs

Apart from content and links, a third source of information are the document's URLs. Entry page URLs often contain the name or acronym of the corresponding organisation. Therefore, an obvious way of exploiting URL information is

N	MRR
content	0.26
1	0.18
5	0.33
10	0.32
50	0.30

Table 2. MRRs Kleinberg@10 results on training data

trying to match query terms and URL terms. Our URL approach however, is based on the observation that entry page URLs tend to be higher in a server's document tree than other web pages, i.e. the number of slashes ('/') in an entry page URL tends to be relatively small.

We define 4 different types of URLs:

- root: a domain name, optionally followed by 'index.html' (e.g. `http://trec.nist.gov`)
- subroot: a domain name, followed by a single directory, optionally followed by 'index.html' name (e.g. `http://trec.nist.gov/pubs/`)
- path: a domain name, followed by an arbitrarily deep path, but not ending in a file name other than 'index.html' (e.g. `http://trec.nist.gov/pubs/trec9/papers/`)
- file: anything ending in a filename other than 'index.html' (e.g. `http://trec.nist.gov/pubs/trec9/t9_proceedings.html`)

We analysed WT10g and the relevant entry pages for half of the training documents to see how entry pages and other documents are distributed over these URL types. Table 3 shows the statistics.

URL type	#entry pages		#WT10g	
root	38	(71.7%)	11680	(0.6%)
subroot	7	(13.2%)	37959	(2.2%)
path	3	(5.7%)	83734	(4.9%)
file	3	(5.7%)	1557719	(92.1%)

Table 3. Distributions of entry pages and WT10g over URL types

From these statistics, we estimated prior probabilities of being an entry page on the basis of the URL type $P(entrypage|URLtype = t)$ for all URL types t. We then interpolated these priors with the normalised content only scores (cf. eq. 2) and tested this on the other 50 entry page search topics of the training data. This gave a major improvement on the content only results (see table 4).

run	MRR
content only	0.26
0.7 * content + 0.3 * URL prior	0.79

Table 4. URL prior results

3.3 Anchors

The fourth source of information is provided by the anchor texts of outlinks. These anchor texts are the underlined and highlighted texts of hyperlinks in web pages. We gathered all anchor texts of the outlinks, combined all texts pointing to the same document to form a new textual representation of that document, and built a separate index on these texts. The texts include the so-called ALT-tags of images as well as the words occurring in the URL.

Note that the score provided by an anchor run is not a document prior. The anchor texts and the body texts ('content-only') provide two very different textual representations of the documents. The information retrieval language models are particularly well-suited for combining several document representations [3]. Our preferred way of combining two representations would be by the following, revised ranking formula.

$$\text{score} = \log(P_{\text{prior}}(D_k)) + \sum_{i=1}^{n} \log((1-\lambda-\mu)P(T_t|C) + \lambda P_{\text{content}}(T_i|D_k) + \mu P_{\text{anchor}}(T_i|D_k)) \tag{6}$$

So, the combination of the anchor run with the content run would be done on a 'query term by query term' basis, whereas the document prior (provided by inlinks or URLs) is added separately. Unfortunately, the current implementation of the retrieval system does not support combining document representations like this. Instead, the anchor runs were done separately from the content runs, their document scores being combined afterwards.

run	MRR
content only	0.26
anchor only	0.29
0.9 * content + 0.1 * anchor	0.36
0.63 * content + 0.07 * anchor + 0.3 * url	0.82

Table 5. MRRs of anchor runs on training data

Table 5 shows the MRRs on half of the training topics. Surprisingly, the anchor-only run slightly outperforms the content-only run. Apparently, search engines do not actually have to see entry pages to provide some useful retrieval functionality. Combining the two approaches leads to improved results. Anchors still seem to provide additional information if they are combined with the successful URL priors.

4 Results

4.1 Ad hoc task

For the Ad Hoc task, we submitted two runs, based on a Jelinek-Mercer smoothing scheme. We did some post hoc runs, based on the Dirichlet smoothing method and were impressed by their superior performance. All runs used only the title field of the topics.

run	description	m.a.p.
tnout10t1	JM smoothing ($\lambda = 0.8$) without doclen prior	0.1652
tnout10t2	JM smoothing ($\lambda = 0.5$) with doclen prior	0.1891
tnout10t3	Dirichlet smoothing ($\mu = 1000$) without doclen prior	0.2039

Table 6. Results of the Ad Hoc runs

Table 4.1 gives the results (mean average precision) for the title only adhoc runs (offical runs in bold font). We know from previous experiments that the Jelinek-Mercer smoothing scheme as such works better on full queries than on title queries. For title queries, the system has too much preference for shorter documents in comparison with the ideal line depicted by $P(rel|dlen)$ (see fig. 3). This can be "compensated" by assuming a prior probability which is linearly dependent on the document length. However, a strict linear relationship will favour long documents too much. An alternative is to use Dirichlet smoothing, such a system yields a $P(ret|dlen)$ curve which has the same shape and

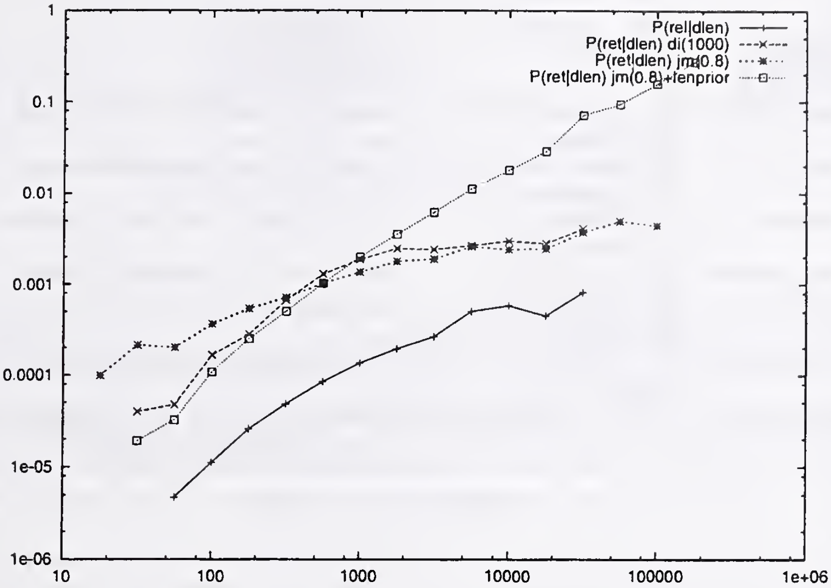


Fig. 3. Probability of relevance and probability of being retrieved as a function of document length

orientation as the ideal $P(rel|dlen)$ curve (fig. 3). The Dirichlet smoothing scheme is less sensitive to query length [8], and the preference for longer documents is inherent, since less smoothing is applied to longer documents.

Figure 3 illustrates the effect. The Dirichlet run follows the shape of the $P(rel|dlen)$ line more closely than the runs based on Jelinek-Mercer smoothing. The JM run based on a document length dependent prior indeed follows the ideal curve better in the lower ranges of document lengths, but overcompensates for the higher document length ranges.

4.2 Entry page task

For the entry page task, we submitted four runs: a content only run, a anchor only run, a content run with URL prior[5] and a run with content, anchors and URL priors. We did some additional runs to have results for all sensible combinations of content, anchors and priors, as well as an inlinkprior run and a Kleinberg run. The mean reciprocal ranks for all runs are shown in table 7 (official runs in bold face). Figure 4 shows the success rate at N for all runs[6] (on a logarithmic scale to emphasise high precision).

The first thing that should be noted from the results is that each combination of content and another source of information outperforms the content only run. The same holds for combinations with the anchor run. However, the improvement when adding URL information is for the anchor run less impressive than for the content run. This is probably due to the differences in the two runs. Although these runs have similar scores (MRR around 0.33), they have different characteristics. The anchor run is a high precision run, whereas the content run also has a reasonable recall. Therefore, it is hard to improve the anchor run since the entry pages that are retrieved are already in the top ranks and the other entry pages are simply not retrieved at all. Figure 4 shows the differences between the two runs: the anchor run has a slightly higher success rate for the lower ranks, but as the ranks get higher, the content run takes over.

As mentioned in section 2.1, our preferred way of combining sources of information when normalisation is not necessary, is to incorporate the additional information in the prior probability of a document. However, in the runs listed in table 7 we interpolated URL priors and inlink priors with the content scores. We did additional runs in which we used the priors exactly as in equation 1; Table 8 shows the results.

[5] We recomputed the priors on the whole set of training data.

[6] The number of entry pages retrieved within the top N documents returned

run	scores	description	MRR
tnout10epC	$contentscore$	Content only run	0.3375
tnout10epA	$anchorscore$	Anchor only run	0.3306
tnout10epCU	$0.7 \cdot contentscore + 0.3 \cdot urlprior$	Content run combined with URL priors	0.7716
tnout10epAU	$0.7 \cdot anchorscore + 0.3 \cdot urlpriors$	Anchor run combined with URL priors	0.4798
tnout10epCA	$0.9 \cdot contentscore + 0.1 \cdot anchorscore$	Interpolation of Content and Anchor runs	0.4500
tnout10epCAU	$0.63 \cdot contentscore + 0.07 \cdot anchorscore + 0.3 \cdot urlpriors$	Interpolation of Content and Anchor runs combined with URL priors	0.7745
tnout10epInlinks	$0.7 \cdot contentscore + 0.3 inlinkprior$	Content run combined with Inlink priors	0.4872
tnout10epKlein10	$Kleinberg's\ auth.score$ @ 10	Authority scores after Kleinberg algorithm on top 10 ranks from Content run	0.3548

Table 7. Entry Page results

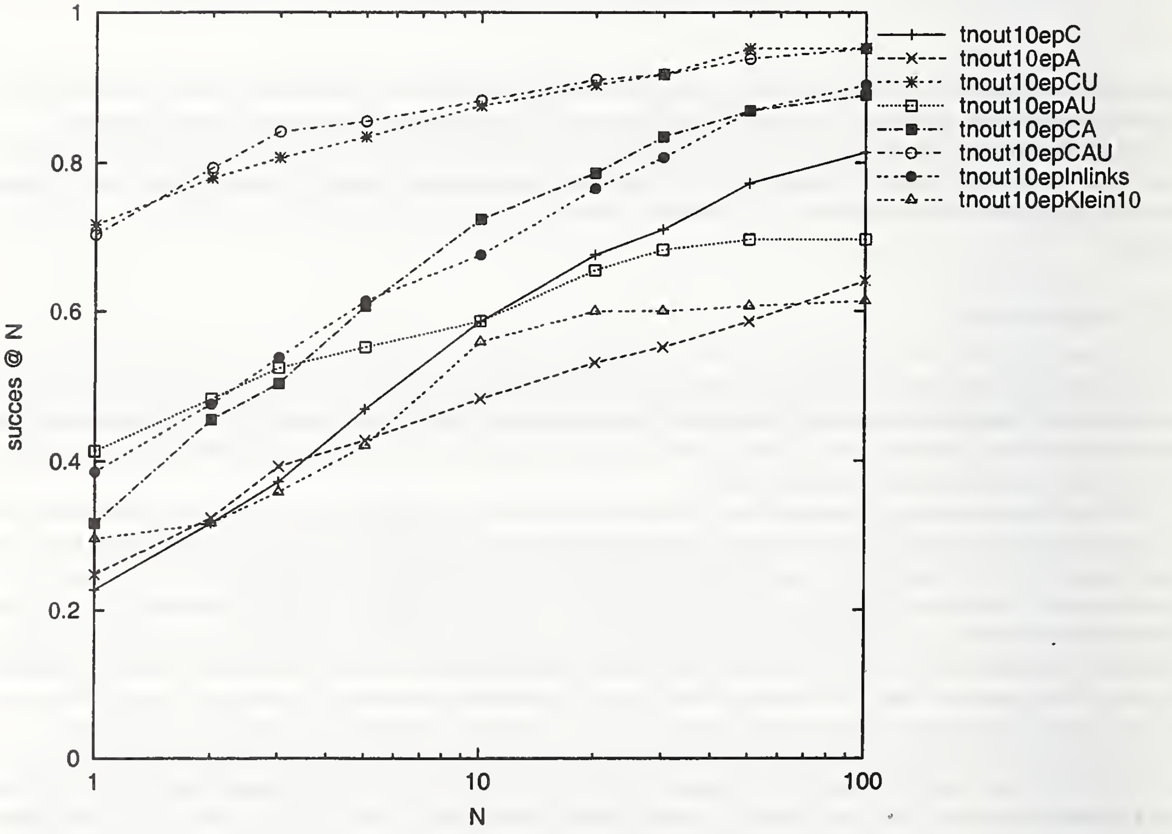


Fig. 4. Entry Page results: success @ N

run	MRR
content only	0.3375
content * URL prior	0.7743
content * inlink prior	0.4251
content * inlink prior * URL prior	0.5440
content * combiprior	0.7746

Table 8. Results with clean (non-interpolated) priors

Table 8 shows that also when we use priors in the clean way(cf. eq 1, they improve our results. Comparing these results to the ones in table 7, we see no difference in performance between the interpolated inlinks and the clean inlinks. The interpolated URL priors are slightly better than the clean ones.

When we take a combination of inlink and URL information as a prior, by simply multiplying the two priors, our results drop (see table 8). This indicates that the two sources of information are not independent. We therefore dropped the independence assumption and had another look at the training data. Just like with the estimation of the URL priors, we subdivided the collection into different categories and estimated prior probabilities of being an entry page given a certain category. As a basis for the categories, we took the 4 URL types defined in section 3.2, then we subdivided the root type into categories on the basis of the number of inlinks. Again we counted the number of entry pages from the training data and the number of documents from WT10g that fell into each category and estimated the prior probabilities from that. We took the categories from the URL types as a starting point and subdivided the root type into 4 subtypes on the basis of the number of inlinks. Table 9 shows the statistics for the different categories.

Document type	#entry pages	#WT10g
root with 1-10 inlinks	39 (36.1%)	8938 (0.5%)
root with 11-100 inlinks	25 (23.1%)	2905 (0.2%)
root with 101-1000 inlinks	11 (10.2%)	377 (0.0%)
root with 1000+ inlinks	4 (3.7%))	38 (0.0%)
subroot	15 (13.9%)	37959 (2.2%)
path	8 (7.4%)	83734 (4.9%)
file	6 (5.6%)	1557719 (92.1%)

Table 9. Distribution entry pages and WT10g over different document types

As can be seen in table 8, this proper combination of URL and inlink information (i.e. without the independence assumption) performs as good as or better than the two separate priors.

5 Conclusion

Post hoc runs show that the Dirichlet smoothing technique yields superior performance for title ad hoc queries on the web collection. This is probably due to the document length dependent smoothing constant, but further investigation is needed.

The Entry page finding task turns out to be very different from an ad hoc task. In previous web tracks link information didn't seem to help for general searches [7] [2]. This year, we found that in addition to content, other sources of information can be very useful for identifying entry pages. We described two different ways of combining different sources of information into our unigram language model: either as a proper prior or by interpolating results from different ranking models. We used both methods successfully when combining content information with other sources as diverse as inlinks, URLs and anchors. URL info gives the best prior info. Adding inlinks yields marginal improvement.

References

1. S. Brin and L. Page *The Anatomy of a Large-Scale Hypertextual Web Search Engine* Proceedings of WWW98, Brisbane, 1998
2. D. Hawking *Overview of the TREC-9 Web Track.* Proceedings of the 9th Text Retrieval Conference (TREC-9), National Institute for Standards and Technology, 2001
3. D. Hiemstra. *Using language models for information retrieval.* PhD thesis, Centre for Telematics and Information Technology, University of Twente, 2001. htpp://www.cs.utwente.nl/~hiemstra/paper/thesis.pdf
4. D. Hiemstra and W. Kraaij. *Twenty-One at TREC7: Ad-hoc and Cross-Language track.* Proceedings of the 7th Text Retrieval Conference (TREC-7), National Institute for Standards and Technology, pages 227-238, 2001
5. J.M. Kleinberg. *Authorative Sources in a Hyperlinked Environment.* Proceedings of 9th ACM-SIAM Symposium on Discrete Algorithms, pages 668-377, 1998
6. W. Kraaij and M. Spitters and M. van der Heijden *Combining a mixture language model and Naive Bayes for multi-document summarisation* Working notes of the DUC2001 workshop (SIGIR2001), New Orleans, 2001
7. W. Kraaij and T. Westerveld. *TNO/UT at TREC-9: How different are Web documents?* Proceedings of the 9th Text Retrieval Conference (TREC-9), National Institute for Standards and Technology, 2001
8. C. Zhai and J. Lafferty *A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval* Proceedings of the 2001 ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01), 2001

Web Reinforced Question Answering (MultiText Experiments for TREC 2001)

C. L. A. Clarke G. V. Cormack T. R. Lynam C. M. Li G. L. McLearn

Computer Science, University of Waterloo, Waterloo, Ontario, Canada
mt@plg.uwaterloo.ca

1 Introduction

For TREC 2001, the MultiText Project concentrated on the QA track. Over the past year, we made substantial enhancements to our QA system in three general areas. First, we explored a number of methods for taking advantage of external resources (including encyclopedias, dictionaries and Web data) as sources for answer validation, improving our ability to identify correct answers in the target corpus. Of the methods explored, the use of Web data to reinforce answer selection proved to be particular value. Second, we made a large number of incremental improvements to the existing system components. For example, in our parsing component, the query generation and answer category identification algorithms were extended and tuned, as were the named entity identification algorithms used in our answer extraction component. Finally, we made a careful analysis of the problem of *null questions*, those that have no answer in the target corpus, and developed a general approach to the problem. A basic method for handling null questions, based on the analysis, was added to our system.

We submitted three runs for the main task of the QA track. The first run (uwmta1) was based on the enhanced system described above, including the full use of Web resources for answer validation. For the second run (uwmta2) the Web resources were not used for validation, but the system was otherwise identical. A comparison between these runs represents a major goal of our TREC experimental work and the major concern of this paper. The final run (uwmta0) tests a last-minute enhancement. For this run a feedback loop was added to the system, in which candidate answer terms were merged back into the query used for passage retrieval. While answer feedback was not an area of significant effort for TREC 2001, and the intial results were disappointing, it represents an area in which future work is planned.

Our other TREC 2001 runs are related to the QA track. Along with the QA runs submitted for the main task, we also submitted exploratory runs for the list (uwmtal0 and uwmtal1) and context (uwmtac0) tasks. These runs were generated through minor modifications to the existing system, and represent preliminary attempts at participation rather than serious attempts at high performance. Our runs for the Web track (uwmtaw0, uwmtaw1, and uwmtaw2) are related to our QA runs. These runs were generated by our QA system by treating the topic title as a question and using the ranked list of documents containing the best answers as the result. Finally, the runs submitted by Sun Microsystems (mtsuna0 and mtsuna1) were generated using our system as the backend and the Sun parser as the frontend. However, the integration between Sun and MultiText was performed in a short period of time, and these runs should also be viewed as preliminary experiments that point toward future work.

In the remainder of the paper we focus on our primary runs for the main task of the QA track. In the next section we provide an overview of the QA system used for our TREC 2001 experiments, including a discussion of our technique for Web reinforcement. In section 3 we present our approach to the problem of null questions. Section 4 details our experimental results.

2 The MultiText QA System

The MultiText QA system was introduced in our TREC-9 paper [1]. and described in further detail in a related SIGIR paper [2] This section presents an updated summary of the system.

Figure 1 provides an overview the version of our QA system used in our TREC 2001 experiments. In our approach, question answering consists of three major processing steps: question parsing, passage retrieval and answer selection.

The MultiText QA parser has two main functions: 1) to generate queries so that the retrieval engine can extract the best candidate passages, and 2) to generate answer selection rules so that the post-processor can select the best 50-byte answer fragment from the passages. The answer selection rules generated by the parser include a category for the question (`<name>`, `<place>`, etc.) and patterns that may be matched in the extracted passages to identify possible answer locations.

Queries generated by the parser are fed to the passage retrieval engine. For question answering, we have developed a passage retrieval technique that can identify small excerpts that cover as many question concepts as possible. Unlike most other passage retrieval techniques, our technique does not require predefined passages, such as paragraphs, sentences or n-word segments, but can retrieve any document substring in the target corpus. The score of substring depends on its length, the number of question concepts it contains and the relative weight assigned to each of these concepts. Once the k highest-scoring substrings from distinct documents are identified, the centerpoint of each substring is computed and a 1000-word passage centered at this point is retrieved from the corpus. These 1000-word passages were then used by the answer selection component to determine the final answers fragments.

The answer selection component identifies possible answers ("candidates") from the passages and then ranks these candidates using a variety of heuristics. These heuristics take into account the number of times each candidate appears in the retrieved passages, the location of the candidate in the retrieved passages, the rank of the passages in which the candidate appears, the likelihood that the candidate matches the assigned answer category, and other special-case information provided by the selection rules.

Since the goal of the TREC 2001 QA experiments was to select 50-byte answer fragments from the retrieved passages, the answer selection technique used to generate our experimental runs does not attempt to identify candidates that are exact or complete answers. Instead, candidates are single terms, where the nature of these terms depends on the category of the question. For example, if a question asks for a proper noun, the candidates consist of those terms that match a simple syntactic pattern for proper nouns; if a question asks for a length, the candidates consist of those numeric values that precede appropriate units; and if a question cannot be classified, the candidates simply consist of all non-query and non-stopword terms appearing in the retrieved passages.

After identification, each candidate term t is assigned a weight that takes into account the number of distinct passages in which the term appears, as well as the relative frequency of the term in the database:

$$w_t = c_t \log(N/f_t),$$

where N is sum of the lengths of all documents in the database, f_t is the number of occurrences

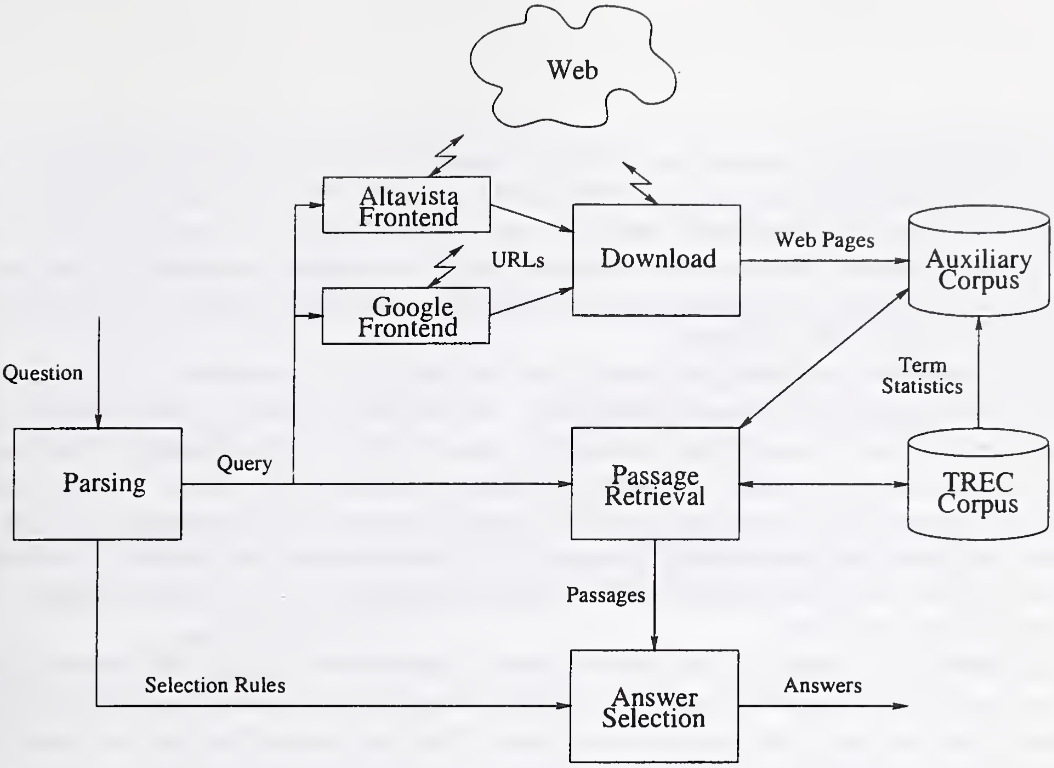


Figure 1: QA System overview.

of t in the database, and $1 \leq c_t \leq k$ is the number of distinct passages in which t appears. The value c_t, which represents the "redundancy" associated with the candidate, is a critical element of the answer selection process [2].

The weights of the candidates are used to select 50-byte answer fragments from the retrieved passages. Each 50 byte substring of the retrieved passages that starts or ends on a word boundary is considered to be a potential answer fragment. A score for each of these fragments is computed by summing the weights of the candidate terms that appear within it. Given a text fragment F and a set of candidates K, each term that appears in both F and K has its weight w_t temporarily modified to a position-specific weight w'_t using heuristics that take into account the rank of the passage in which the fragment appears, the location of the fragment relative to the centerpoint of the passage, and the selection rules generated by the parser. The resulting score for a fragment is

$$\sum_{t \in F \wedge t \in K} (w'_t)^{\alpha}$$

where a value of $\alpha = 3$ was used for all our TREC 2001 experiments.

Once the highest-scoring fragment is selected, the weights of the candidates appearing in that fragment are reduced to zero. All fragments are re-scored and the highest-scoring fragment is again selected. This process is repeated until five answer fragments have been selected.

At the level of detail given above, our QA system is little changed from TREC-9. However, for TREC 2001 we expanded and enhanced many of the heuristics in the parser and answer selection components. The number of question categories was increased from eight to 22, and these categories were arranged hierarchically. We extended and improved the pattern matching process for recognizing candidates corresponding to these categories. This extended matching process relies heavily on external resources, such as large lists of countries and cites, dictionaries, and the WordNet lexical database. If insufficient candidates are identified using the question category assigned by the parser, the selecion component considers patterns matching categories farther up the hierarchy. Finally, position- and rank-specific adjustments to candidate weights were modified to take the question category into account.

Another addition for TREC 2001 was the use of Web data to reinforce the scores of promising candidates by providing additional redundancy. As shown in figure 1, each question is used to generate appropriate queries for two commercial search engines. The contents of the top 200 documents returned by each engine were used to create an auxiliary database. The passage retrieval component then extracts 20 passages from the target database and 40 passages from the auxiliary database, recording the source of each passage. Since the contents of the auxiliary database is heavily biased by the query, term statistics from the target corpus are used during passages extraction from the auxiliary database.

All 60 passages are passed to the answer selection component. The answer selection component then proceeds to select answer fragments as usual, except that fragments cannot be selected from passages extracted out of the auxiliary database. The Web data influences the answer selection process only by increasing the redundancy factor c_t for particular candidates.

3 Null Questions

Null questions are those questions which have no answer in the target corpus. For TREC 2001, a corresponding null ("no answer") response was treated as a legitimate answer that could be included at any rank. If a question was judged to have no answer in the target corpus, null responses were marked as correct.

Given a question Q, a small number of parameters must be estimated to determine the best rank (if any) to place a null response. The first of these parameters is $p_0(Q)$, the probability that Q has no answer. In addition, for any question, our system will produce five ranked answer fragments. For each question, we have $p_i(Q)$, the probability that the highest-ranked fragment containing the correct answer is located at rank i, with $1 \leq i \leq 5$.

Given these parameters we can compute the expected mean reciprocal rank (MRR) for Q ($E_i(Q)$) under the assumption that the null response is placed at a particular rank i, pushing down the answer fragments appearing at the same rank and lower. For example,

$$E_3(Q) = \frac{p_0(Q)}{3} + p_1(Q) + \frac{p_2(Q)}{2} + \frac{p_3(Q)}{4} + \frac{p_4(Q)}{5}.$$

In addition we define $E_0(Q)$ as the expected MRR if a null response is not included:

$$E_0(Q) = \sum_{i=1}^{n} \frac{p_i(Q)}{i}.$$

The optimal location for a null response is then simply the value of i for which $E_i(Q)$ has maximum value:

$$\arg\max_{0 \leq i \leq 5} E_i(Q).$$

Estimating $p_i(Q)$ for a specific Q proved to be a difficult problem, and for TREC 2001 we simply used fixed estimates $p_i(Q) = p_i$ $(1 \leq i \leq 5)$ for all questions. These fixed estimates were derived

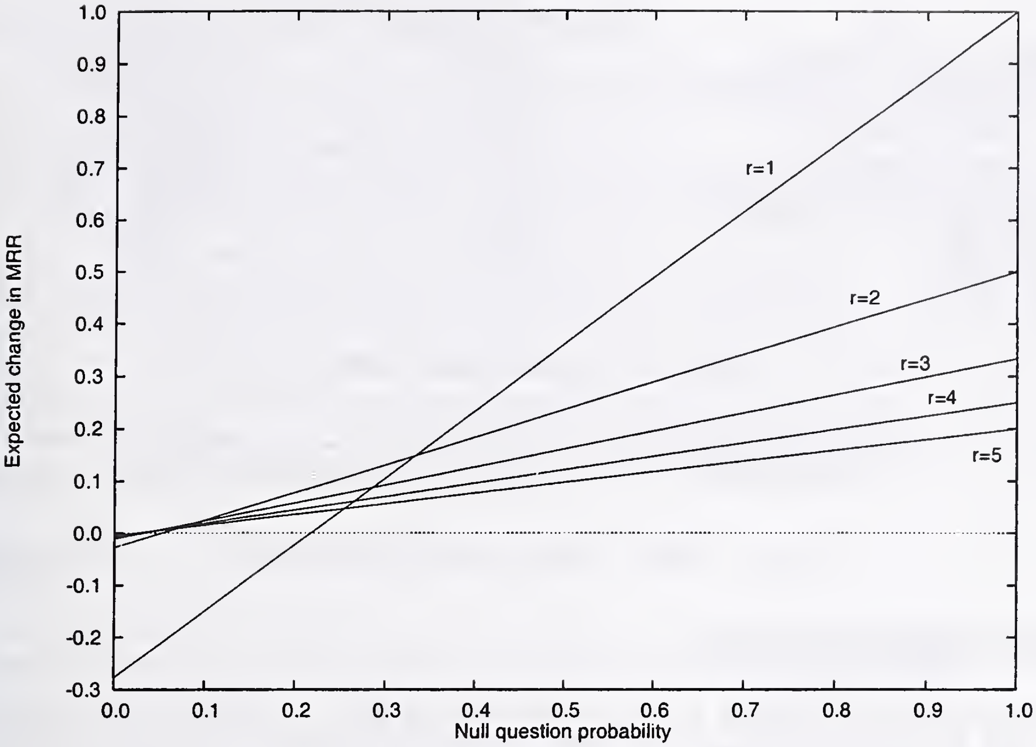


Figure 2: Effect of placing a null response at various ranks (r).

from the actual performance of our TREC 2001 system on the TREC-9 questions as judged by the NIST-supplied judging script. The values of the estimates used for TREC 2001 submissions are:

p_1	p_2	p_3	p_4	p_5
0.500	0.100	0.050	0.033	0.025

In a similar fashion we used a fixed estimate $p_0(Q) = p_0$ for the probability that a question has no answer. Fixing the value of the $p_i(Q)$ for all questions fixes the values for estimated MRR $E_i(Q) = E_i$ $(0 \leq i \leq 5)$ and implies that a null response should always be placed at fixed rank r or always omitted. The precise action taken depends on the values of the estimates.

Deriving a estimate for p_0 that was anything but a guess proved to be impossible. The value of p_0 represents the ratio of questions in the test set that have no known answers in the target corpus. The selection of this value was entirely the choice of NIST, and was not released to participants in advance.

Since a meaningful estimate of p_0 could not be obtained, we treated the value as a free parameter and examined the impact of its value on the expected change in MRR $(E_r - E_0)$ for the possible values of r. The results are shown in figure 2.

After some discussion between members of the group, we agreed that the proportion of questions with no known answer was unlikely to fall below 10% and unlikely to be greater than 20%; our best guess was 10%. A small minority felt that the value would be very small (1–2%). For p_0 in the range $(0.10, 0.20)$ values of $r = 2$ and $r = 3$ both produce small positive improvements to MRR. In the end, a value of $r = 3$ was selected to minimize the consequences of an extremely small p_0.

Run	MultiText Judgment	NIST Judgment (strict/lenient)
MultiText baseline (uwmta2)	0.379	0.346/0.365
+ Web re-inforcement (uwmta1)	0.483	0.434/0.457
+ feedback (uwmta0)	0.482	0.404/0.450
Sun baseline (mtsuna1)	N/A	0.307/0.322
+ Web re-inforcement (mtsuna0)	0.416	0.405/0.418
Web data only	0.608	N/A
TREC-9 method	0.317	N/A

Figure 3: QA main task results

null response rank	1	2	3	4	5
adjusted MRR	0.311	0.440	0.434	0.426	0.424

Figure 4: Effect of relocating the null response

4 Experimental Results

Our main task question answering results are presented in figure 3. The figure includes both the results of our own submissions and the results of the Sun submissions, which used our system backend, including its passage retrieval and answer selection components. The third column lists official NIST judgments or results derived from them. The second column lists unofficial judgments made by one of the authors (Clarke) immediately after the runs were submitted to NIST. Although creation of these unofficial judgments required less than two hours of total effort, their relative values appear to correlate well with the official judgments, with slightly higher absolute values. In this discussion below, we use these unofficial numbers to support comments that cannot be supported by the official numbers.

The use of Web reinforcement produced a 25% improvement on our own results (uwmta2 vs. uwmta1) and a 30% improvement on the Sun results (mtsuna1 vs. mtsuna0). Considering that the Web data can influence the answer selection process only through modifications to the candidate redundancy parameter c_t, the magnitude of the improvement is surprising and provides substantial support for our view that candidate redundancy is a key factor in question answering [2]. To provide an additional comparison, the top five answers were selected from the Web data used to reinforce the MultiText runs and were judged by Clarke ("Web data only").

For the TREC 2001 questions, p_0 was 10%. As a result, the decision to always place a null response at rank 3 had a small but positive impact. In reality, our performance estimates for our system (p_i, $1 \leq i \leq 5$) were somewhat optimistic. Nonetheless, rank 3 proved to be a good choice. Figure 4 shows the change to the strict MRR for our best run (uwmta10) if other ranks were chosen for the null response. Rank 2 would have been a slightly better location, but the potential improvement is less than 2%. If the null response had been omitted, we estimate that the MRR for uwmta10 would have been 0.421. Thus, our choice to always place a null response at rank 3 gave a performance improvement of roughly 3%.

As a final experiment, we executed our TREC-9 system on this year's questions. Based on the judgments made by Clarke, the total effect of our efforts this year was an overall performance improvement of more than 50%.

5 Conclusion and Future Work

We are continuing to enhance and extend our question answering system. The performance of all aspects of the system is currently under review and many of the components will be heavily modified or replaced over the coming year.

If the approach taken to null questions in TREC 2001 is continued in future TREC conferences we plan to improve our technique by taking question-specific information into account. For example, we intend to consider question category when estimating values for p_i $(1 \leq i \leq 5)$. Also we hope that NIST will release a prior probability p_0 that a question will have no answer, since this information is critical for placing null responses and in practice could be readily estimated from query logs.

Finally, we are actively experimenting with Web-based question answering, both as a method of reinforcing question answering from closed collections and as an end in itself. We are presently in the process of creating a $> 1TB$ collection of Web explicitly to support question answering, replacing the commercial search engines used in our TREC 2001 experiments

References

[1] C. L. A. Clarke, G. V. Cormack, D. I. E. Kisman, and T. R. Lynam. Question answering by passage selection. In *9th Text REtrieval Conference*, Gaithersburg, MD, 2000.

[2] 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.

A prototype Question Answering system using syntactic and semantic information for answer retrieval

Enrique Alfonseca, Marco De Boni, José-Luis Jara-Valencia, Suresh Manandhar

Department of Computer Science
The University of York
Heslington
YORK
YO10 5DD
United Kingdom

Introduction

This was our first entry at TREC and the system we presented was, due to time constraints, an incomplete prototype. Our main aims were to verify the usefulness of syntactic analysis for QA and to experiment with different semantic distance metrics in view of a more complete and fully integrated future system. To this end we made use of a part-of-speech tagger and NP chunker in conjunction with entity recognition and semantic distance metrics. We also envisaged experimenting with a shallow best first parser but time factors meant integration with the rest of the system was not achieved. Unfortunately due to time constraints no testing and no parameter tuning was carried out prior TREC. This in turn meant that a number of small bugs negatively influenced our results. Moreover it was not possible to carry out experiments in parameter tuning, meaning our system did not achieve optimal performance. Nevertheless we obtained reasonable results, the best score being 18.1% of the questions correct (with lenient judgements).

Question-Answering algorithm

The YorkQA question answering system takes inspiration from the generic question answering algorithm presented in [Simmons 1973]. Although unacknowledged by more recent research its general structure is very similar to the basic algorithms used in the Question Answering systems built for previous TREC conferences

The algorithm proceeds as follows:

I) Accumulate a database of semantic structures representing sentence meanings.

II) Select a set of structures that appears relevant to the question. Relevance is measured by the number of lexical concepts in common between the proposed answer and the question. This is done by ordering the candidates according to the number of Token values they have in common with the questions.

III) Match the question structure against each candidate. This is done by:

- Determining if the head verb of the question matches the head verb of the candidate. If there is no direct match, a paraphrase rule is applied to see if the question structure can be transformed into the structure of the answer. Paraphrase rules are stored as part of the lexicon and an examination of the lexical structures of two words will be able to determine if there is a rule (path) connecting the two. If there is not, the set of words that the first transforms into is recursively examined to see if can be transformed into the second word. If this fails, the transformation rules are recursively applied to the second word to see if a match can be found. This procedure continues until either a match is found or an arbitrarily set depth is reached.
- Applying the same procedure to the other words in the question and the candidate answer question in order to transform the question structure into the form of the candidate answer.
- Examining quantifiers and modalities to see if quantificational, tense and negation relationships are matched.
- Examining the question's semantic structure to determine if the question word type (the wh-word) is present and satisfied in the answer

Our algorithm proceeded as follows:

1. Index the documents using a standard Information Retrieval engine
2. Read in the next question, and repeat the following until there are no more questions:
3. Analyse each question to determine the question category, i.e. the type of entity the answer should contain
4. Decide the query to send to the IR engine and send the query
5. Pass the retrieved documents to the sentence splitter, tagger, chunker and named entity recogniser
6. Analyse the sentences to determine if they contain the correct entity type.
7. Rank the sentences containing the correct entity type according to their semantic distance from the question.
8. Use a parser to determine the answer by finding a match between the question and the ranked sentences
9. If matching through the parser fails look for the best entity in the top ranked sentences or, if an entity of the appropriate type is not found, an appropriate 50 byte window.

Step 1 of our algorithm corresponds to point I in Simmons' algorithm, 2-7 correspond to II and 8-9 correspond to III. Due to time constraints we were unable to implement 8.

What follows is a more detailed description of the main parts of our system.

Question Type Identification

The question analyser used pattern matching based on wh-words and simple part-of-speech information combined with the use of semantic information provided by WordNet [Miller 1995] to determine question types. Unlike other question analysers constructed for previous TREC QA tracks (e.g. [Harabagiu 2001]) our question type analyser only recognised a small number of different types. Question types were limited to time, place, currency, organisation, length, quantity, reason, place, constitution, person, length, mode, recommendation, truth and a fallback category of thing. A number of these, for example recommendation ("Should I do X?"), reason ("Why did X occur") and truth ("Is X Y?") revealed themselves not to be needed for the track as no questions of that type were present. An initial evaluation estimated an accuracy of 83% for the question recogniser.

Information Retrieval

Steps 1 and 3 of our algorithm were carried out by a) using the SMART Information Retrieval system (for an introduction to SMART, see [Paijmans 1999]) to index the documents and retrieve a set of at most 50 documents using the question as the query; and b) taking the first 50 documents supplied by the Prise Information Retrieval engine, as provided by the TREC organisers. Unfortunately due to time constraints we were unable to tune the SMART IR system for the task at hand and in fact the documents retrieved by the SMART engine were much worse (very low in both precision and recall) than the documents provided by NIST.

The YorkQA system then transformed the output of the retrieval engine into appropriate XML documents and used a number of tools for linguistically processing them, in particular a tokeniser, sentence splitter, part-of-speech tagger, morphological analyser, entity recogniser, QP- and NP-chunker.

Tagging and chunking

The texts were processed using the TnT part-of-speech tagger [Brants, 2000] and a Noun Phrase chunk parser based on Transformation Lists [Ramshaw and Marcus, 1995] to find non-recursive noun phrases. To improve the accuracy of the chunker, we trained it on a copy of the Wall Street Journal corpus that was manually corrected by a human annotator, so as to improve the quality of the training data. This increased the initial accuracy more than 3% to 95%.

The sentence splitter was based on [Mikheev, 1999]. It is divided in two steps of processing dot-ending words. Firstly, the system decides whether they are or not abbreviations. Non-abbreviations ended with dots all indicate sentential endings; and abbreviations followed by a non-capitalised word are never sentence ends. The second step addresses the difficult case, when an abbreviation is followed by a capitalised word, and several heuristics are used to decide whether a sentence ends there or not.

Named Entity Recognition

This was an important step in the processing of the text as the YorkQA system initially tried to find sentences containing an appropriate entity that might answer a determined question. Nevertheless this module was, for this version, the weakest link in the processing pipeline and should, and will, be subject to serious improvement in the future. At present, this tool aims to recognise six types of entities:

- *Currency expressions*, such as "$10,000", "2.5 million dollars", etc. The evaluation shows a very high accuracy for this sub-module (4/4), but all the expressions found were in dollars and it is not clear that this performance could be obtained for other types of currencies. Its precision however should remain very high in any stricter evaluation.
- *Locations*, such as "Chicago", "Thames", "Mount Kilimanjaro", etc. This sub-module is programmed to recognise locations by context words, so that it recognises "New York City" but not "New York". This limitation is clearly reflected in its performance as this module missed all location expressions (0/32) in the sentences selected for the manual evaluation.
- *Organisations*, such as "Climbax Corp.", "Nobel Prize Committee", "The National Library of Medicine", etc. The approach used by this sub-module is to look for a clear organisational word at the end of a sequence of capitalised words. Therefore, it recognises "National Cancer Institute" but not "Massachusetts Institute of Technology". Consequently, we determined that only 35% (8/23) of the organisations mentioned in the evaluated text were correctly tagged and that most of the error were due to poor recall.
- *People names*, such as "Reagan", "Marilyn Monroe", "G. Garcia Marquez", etc. Again precision is preferred to recall in this sub-module. Therefore, people's names are marked when they are surrounded by clear context words such as personal titles ("Mr.", "Dr.", etc.) and common pre-name positions ("President ...", "Sen. ...", etc.) or both the forename(s) and the surname(s) are found in a gazetteer of common names in several languages. The evaluation is consistent with this bias as most of the 69% of incorrect answer (13/42) from this module are consequence of poor recall.
- *Quantity expressions*, such as "twelve", "11/2 billion", etc. Because of the relative regularity in this kind of expressions, this sub-module gets a fairly good accuracy (60/73) and most errors are misinterpretations of the words "a" and "one".
- *Time expressions*, such as "June, 28 1993", "Today", "late 2000", etc. The accuracy of this sub-tool is around 50%, mainly because it misses many relative time expressions (such as "the day after the great storm") which are difficult to capture by regular expressions only.

Clearly these expressions help the system to answer questions about people, organisations, money, etc. but we are very aware that much more must be done on this. For example, the system would get better performance if a broader range of entities were recognised, in particular speeds, weights, lengths, duration expression, etc. will certainly increase the focus of the search which at present can only rely on quantity entities. This and other improvements are planned in the future and we hope to have in hand a much better entity recogniser for the next version of YorkQA.

Measuring Semantic distance

The central part of our system was the semantic distance measuring module, which analysed the tagged and chunked sentences in the documents selected by the Information Retrieval engine, comparing them with the question in order to find the sentence which was most similar.

To calculate similarity was necessary to calculate the semantic (or conceptual) distance between a sentence and a question. A number of algorithms have been presented to measure semantic (or conceptual) distance (see for example [Miller and Teibel 1991, Rada et al. 1989]). WordNet [Miller 1995] has been shown to be useful in this respect, as it explicitly defines semantic relationships between words (see, for example, [Mihalcea and Moldovan 1999] and [Harabagiu et al. 1990]). Our system, however, differs from previous approaches in that it does not limit itself to considering the WordNet relationships of synomymity and hyperonymy, but makes use of all semantic relationships available in WordNet (is_a, satellite, similar, pertains, meronym, entails, etc.) as well as information provided by the noun phrase chunker.

Answer identification

The question types were used to identify a candidate answer within the retrieved sentences in the case of questions of type quantity, person, place, time and currency; in the case of questions of type reason, recommendation, mode and difference, appropriate keywords were searched for (e.g. a part of sentence following the word "because" is probably a reason); finally, if the question analyser failed to recognise any question type (i.e. the answer type was the catch-all category "thing"), the system looked for a portion of text 50 bytes long within a sentence which was closest to the words contained in the question, but contained the lowest number of question words.

Results, Conclusions and further work

Given that our system was a simple prototype which had not been tested, nor tuned, the results we got were encouraging, the best score being 18.1% of the questions correct (with lenient judgements), proving that the use of syntactic and semantic information can be fruitfully used for the QA task.

The current system will have to be fully tested and debugged and a full evaluation will have to be carried out on each separate module to evaluate performance and identify the source of errors.

Future work will include: generic question type identification (needed since handing coding for each type is cumbersome and time consuming); an improved named-entity recogniser; improved semantic distance metrics; a parser to transform sentences into a simple logical form which can then be manipulated in order to find an answer.

Bibliography

Brants, T.,"TnT - A Statistical Part-of-Speech Tagger", User manual, 2000

Harabagiu, S. A., Miller, A. G., Moldovan, D. I., "WordNet2 - a morphologically and semantically enhanced resource", In Proceedings of SIGLEX-99, University of Maryland, 1999.

Harabagiu, S.,et al., FALCON - Boosting Knowledge for Answer Engines. In Proceedings of TREC-9, NIST, 2001.

Mikheev, Andrei, "Periods, capitalized words, etc." Submitted to Computational Linguistics, 1999

Miller, G, and Teibel, D., "A proposal for lexical disambiguation", in Proceedings of DARPA Speech and natural Language Workshop, California, 1991.

Miller, G. A., "WordNet: A Lexical Database", Communications of the ACM, 38 (11), 1995.

Paijmans, Hans, "SMART Tutorial for beginners.", http://pi0959.kub.nl:2080/Paai/Onderw/Smart/hands.html, 1999

Rada, R., Mili, H., Bicknell, E. and Blettner, M., "Development and application of a metric on semantic nets", in “IEEE Transactions on Systems, Man and Cybernetics”, vol.19, n.1, 1989.

Rada Mihalcea and Dan Moldovan, A Method for Word Sense Disambiguation of Unrestricted Text, in Proceedings of ACL '99, Maryland, NY, June 1999.

Ramshaw, Lance A. and Marcus, Mitchell P. , "Text Chunking Using Transformation-Based Learning", Proceedings of the Third ACL Workshop on Very Large Corpora", 82--94, Kluwer, 1995.

Simmons, R. F., "Semantic Networks: computation and use for understanding English sentences", in Schank, R. C. and Colby, K. M., "Computer Models of Thought and Language", San Francisco, 1973.

Machine Learning Approach for Homepage Finding Task

Wensi Xi and Edward A. Fox
Department of Computer Science
Virginia Polytechnic Institute and State University

ABSTRACT

This paper describes new machine learning approaches to predict the correct homepage in response to a user's homepage finding query. This involves two phases. In the first phase, a decision tree is generated to predict whether a URL is a homepage URL or not. The decision tree then is used to filter out non-homepages from the webpages returned by a standard vector space IR system. In the second phase, a logistic regression analysis is used to combine multiple sources of evidence on the remaining webpages to predict which homepage is most relevant to a user's query. 100 queries are used to train the logistic regression model and another 145 testing queries are used to evaluate the model derived. Our results show that about 84% of the testing queries had the correct homepage returned within the top 10 pages. This shows that our machine learning approaches are effective since without any machine learning approaches, only 59% of the testing queries had their correct answers returned within the top 10 hits.

1. INTRODUCTION

With the fast development of the internet and World Wide Web, information from the Web has become one of the primary sources of knowledge for human beings. Although traditional information retrieval techniques have provided many methods to seek relevant information from the internet in response to a user's need, they are still far from sufficient in some cases, such as when a user is seeking information that is too broadly or vaguely specified for a traditional IR system to give a precise result. On the other hand the linking structure and various tagged fields of a Web page can be rich sources of information about the content of that page. Making use of this information can be very helpful in solving those information seeking problems that can not be satisfactorily solved by traditional IR techniques. Among this kind of user's information need are special information seeking tasks like "homepage finding" which involves trying to find the entry page to a website. This paper describes new methods of using machine learning approaches to consider extensive tagged field, URL, and other information to best predict the relevant homepage in response to a user's homepage finding query. The rest of the paper will be organized as follows: related work will be introduced in Section 2; research direction will be described in Section 3; the baseline IR system will be explained in Section 4; machine learning models and results will be reported in Sections 5 and 6; and research will be summarized and discussed in Section 7.

2. RELATED WORK

There are two major methods to make use of link information to identify the correct webpage in response to a user's query: the page rank algorithm and the HITS algorithm.

The page rank algorithm was first introduced by Page and Brin [1]. This algorithm was developed because using in-degree as the predictor of quality is weak. First, not all the back pages are of the same importance. Second, in-degree is spammable. In their page rank algorithm each page was first evaluated as to quality. Then each page allows all the page links to it to distribute their "value" of quality to it. The quality value of each page was divided by the out-degree before they could distribute their "authority" to other pages. The algorithm can be summarized as:

$$\text{PageRank}(P) = \beta/N + (1-\beta)\Sigma\text{PageRank}(B)/\text{outdegree}(B)$$

where β is the probability of a random jump to P and N is the total number of pages on the web.

The HITS algorithm was first introduced by Kleinberg [4]. He assumes that a topic can be roughly divided into pages with good coverage of the topic, called authorities, and directory-like pages with many hyperlinks to pages on the topic, called hubs. The algorithm aims to find good authorities and hubs for a topic. For a topic, the HITS algorithm first creates a neighborhood graph. The neighborhood contains the top 200 matched webpages retrieved from a content based web search engine; it also contains all the pages these 200 webpages link to and pages that linked to these 200 top pages. Then, an iterative calculation is performed on the value of authority and value of hub. Iteration proceeds on the neighborhood graph until the values converge. Kleinberg claimed that the small number of pages with the converged value should be the pages that had the best authorities for the topic. And the experimental results support the concept. Kleinberg also pointed out that there might be topic diffusion problems (with the answer shifting to a broader topic related to the query). There also might be multi-communities for a query, where each community is focused on one meaning of the topic. Sometimes the first-principal community is too broad for the topic and the 2nd and 3rd community might contain the right answer to the user's query.

Combining multiple sources of evidence from different IR systems to improve the retrieval results is a method applied by many researchers (e.g. [7] [9]), and had been proved to be effective. Using regression analysis to improve retrieval also had been studied, e.g. in [2].

Recently, Craswell and Hawking [3] used anchor text to retrieve documents in response to a homepage finding task, and compared their result with full-text retrieval. They found anchor text retrieval is far more effective than full-text retrieval.

3. RESEARCH DIRECTION

Our research makes use of the WT10g web collection provided by the TREC staff. The WT10g collection is about 10GByte in size and contains 1,692,096 webpages crawled in 1997. The average size of a webpage in the collection is 6.3 KBytes.

The TREC Conference provided 100 sample homepage finding queries and their corresponding correct answers (homepages). These sample queries can be used to train the homepage finding system developed. TREC also provided another 145 testing queries without corresponding answers. These queries can be used to evaluate the system developed.

The 100 sample homepage finding queries are very short queries. Most of them only contain 2 to 3 words. They include the name of an institute (e.g., UVA English department), organization (e.g., Chicago Computer Society), or a person's name (e.g., Jim Edwards). Some of the queries also contain descriptive information (e.g., Unofficial Memphis Home Page). After a close analysis of the 100 training queries and URLs of their corresponding homepages, we found these clues:

- A homepage usually ends with a "/"
- Most homepages contain at most 2 other "/", beyond the 2 in http://
- The last word in the homepage URL (if the URL is not ending with a "/") is usually: index.html; index1.html; homepage.html; home.html; main.html; etc.

Most of the 100 sample homepages confirm these rules. However there are exceptions, for example:

McSportlight Media This Week ->
http://www.mcspotlight.org:80/media/thisweek/
LAB MOVIE REVIEW SITE –>
http://www.ucls.uchicago.edu:80/projects/MovieMetropolis/
The Boats and Planes Store ->
http://www.psrc.usm.edu:80/macrog/boats.html

The basic rationale for UR analysis is to filter out non-homepages that rank at the top of the rank list returned by the content based information retrieval system, so that the correct hompages can be ranked higher.

The TREC also provided two mapping files:

in_links: which maps the incoming links to each collection page

out_links: which maps outgoing links from each collection page

4. BASELINE IR SYSTEM

At the beginning of this research, a vector space model IR system was developed to retrieve relevant webpages for each of the 100 training homepage finding queries. The vector space model IR system uses a stop word list to filter out high frequency words. Each word left is stemmed using Porter's algorithm [4]. The IR system uses the *ntf*idf* [6] weighting scheme with cosine normalization to construct the query vectors and the *tf*idf* weighting scheme with cosine normalization to construct the document vectors. *ntf* refers to *normalized term frequency* and is given by the formula:

$$ntf = 0.5 + 0.5 * tf / max\_tf$$

where *max_tf* is the highest term frequency obtained by terms in the vector. The retrieval score for the document is calculated by taking the inner product of the document and query vectors.

The WT10g Web collection contains 3,353,427 unique keywords (after filtering out stopwords, and stemming). The inverted file developed from this collection is about 3 Gbytes in size.

Tagged fields:

In order to investigate the importance of tagged fields in HTML files during the retrieval, several tagged fields were extracted from the WT10g collection. The tagged fields extracted were <title>, <meta>, and <h1>.

Anchor texts:

Anchor texts are the text description of a hyperlink in a webpage. Previous research [3] had found that anchor text retrieval could help improve retrieval performance. In this research work, we extracted and combined the anchor texts with the destination webpage it links to and built a separate anchor text collection, in which each page only contains all the anchor text of other pages describing it.

Abstracts:

Some researchers [5] had found that text summary and abstract retrieval can yield comparable or even better retrieval results than full-text retrieval. Retrieval using abstracts also can save substantial time and space. In this research work, we extracted text to approximate an abstract for each webpage. The abstract contains the URL of the webpage, the <title> tagged field of that page, and the first 40 words following that field in that page. The extracted abstract collection is about 7% of the size of the WT10g collection. The rationale for the abstract collection is that we believe a homepage is very likely to repeat its name in its URL, title, or at the beginning of its homepage, and so this is more likely to achieve better results than full-text retrieval. On the other hand, the abstract contains more information than would the title field, and is not likely to lose the correct answer to queries; thus we should obtain higher recall.

The statistical facts of the full-text, tagged field, anchor, and abstract collections are listed in Table 1 below:

Table 1. Statistical facts of the various collections

Name	Size (Mbytes)	No. of Docs	Avg Doc Length (Kbytes)	Inverted File Size (Mbytes)	No. of Unique Terms
Full text	10000	1692096	6.3	3000	3353427
Title tag	100	1602137	62.5	59	158254
Meta tag	50	306930	167	28	59122
H1 tag	29	517132	56	15	82597
Anchor	180	1296548	138	53	219213
Abstract	710	1692096	420	400	646371

Retrieval results

Table2 and Figure1 report the retrieval result for the 100 testing queries on different collections. From the table we find that the <meta> tag and <h1> tag each performs poorly. This shows that the text in these fields is not a good indication of the main topic of the webpage. Full text retrieval doesn't work very well either. Abstract retrieval works much better than the full-text retrieval as we expected. Anchor text retrieval performs slightly better than abstract retrieval in terms of MRR (Mean Reciprocal Rank). Title tag retrieval performs best of all.

Table 2. Baseline system retrieval results for training queries

Relevant doc found in	full-text	title tag	meta tag	h1 tag	anchor text	abstract
Top1	5	34	4	7	22	23
Top5	18	62	8	11	47	40
Top10	25	68	11	14	54	49
Top20	34	73	14	14	57	59
Top100	48	85	18	15	65	73
Not in list	0	5	73	84	18	2
MRR	0.12	0.46	0.06	0.09	0.33	0.31

$MRR = \Sigma(1/rank)/N$

N: Number of queries

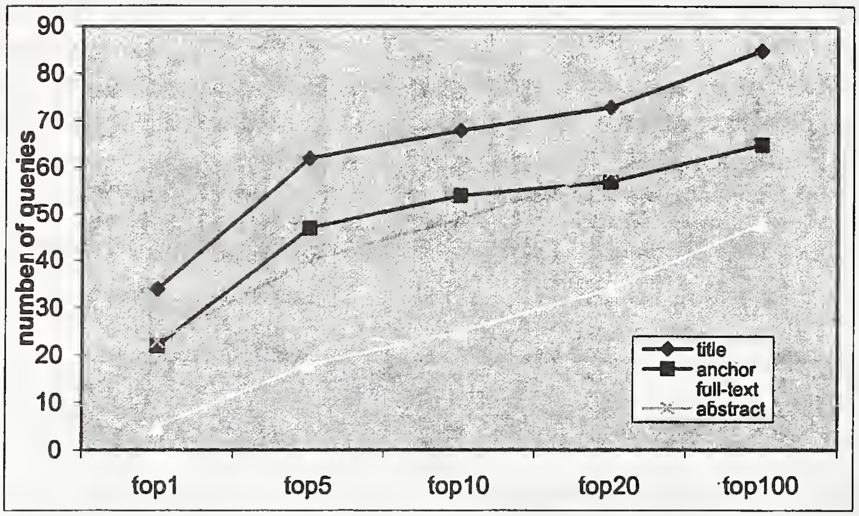


Figure 1. Baseline retrieval results comparison chart for training queries

5. DECISION TREE MODEL

In the second phase of our research work, a decision tree was generated to predict whether a URL of webpage is a homepage URL or not. The detailed steps are:

1. Manually select 91 non-homepages from the WT10g collection. These pages are identified not only by the content but also by the in-links and out-links of the pages and by the structure of the URL.

2. Develop attribute vectors for the 198 cases (107 positive cases provided from TREC and 91 negative cases derived manually); the attribute vectors contain these factors:

- URL length: the number of slashes in the URL;
- In link: the total number of in links;
- In link normalized by homepage: total number of in links divided by the length of the webpage;
- In link from outer domain: the number of in links from outer domains;
- In link from same domain: the number of in links from the same domain;
- Out link: total number of out links of a webpage;
- Out link normalized by homepage: the total number of out links divided by the length of the Web page.
- Out link to outer domain: the number of out links pointing to other domains,
- Out link to same domain: the number of out links pointing to the same domain;
- Keyword: whether the URL ends with a keyword: these keywords are "home". "homepage", "index", "default", "main";
- Slash: whether the URL ends with "/";

- Result: whether it is a homepage or not.

3. The 198 training vectors were provided to the data mining tool C5 or See5 (available at http://www.rulequest.com/see5-info.html). A decision tree was developed by the rule generator based on these training vectors. It can be seen in Figure 2. The correctness of the decision tree against the training cases is 97%.

4 Another 102 test Web pages were manually selected from the TREC collection. Among them, 27 are homepages. The decision tree was evaluated on the test cases and the results were 92% correct. This indicates that the decision tree model is fairly reliable.

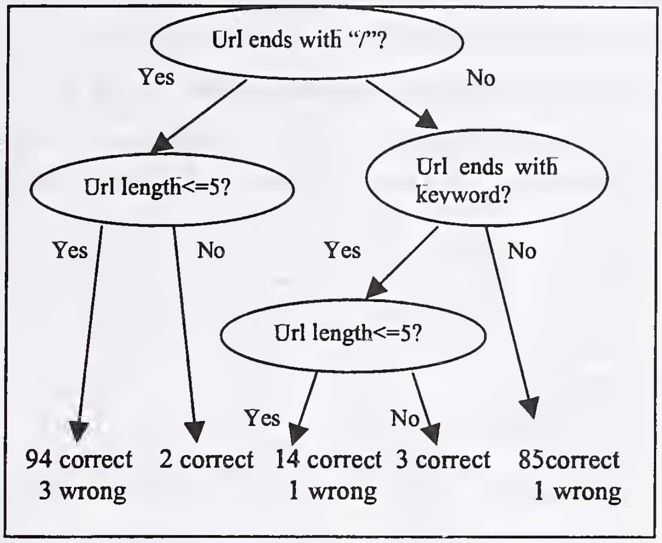


Figure 2. Decision Tree Model

5. The decision tree then was applied to the results returned by the baseline IR system, in hopes that we can filter out most of the non-webpages in these returned webpage lists. The decision tree model was only applied to anchor, title field, and abstract retrieval. Results of the decision tree applied on the title and anchor text retrieval can be found in Table 3.

6. LOGISTIC REGRESSION MODEL

In the third stage of this research work, a logistic regression analysis model was developed to combine link information with the various scores returned by the standard IR system, in order to improve the rank of the correct homepages in response to the query. The detailed steps are:

1. Two training queries (No. 5 and No. 51) were taken out of consideration because their correct answer was already filtered out as non-homepages by the decision tree model. The top 1000 pages for each rank-list file of the remaining 98 training queries were taken into the logistic regression analysis. Thus, there were 67855 pages in the training set; among them 104 pages were relevant to a specific query.

2. A logistic regression analysis was made using SAS software, version 8.02. The evidence thrown into the logistic regression analysis included IR scores from title, anchor, and abstract retrieval. (All scores are pre-normalized by the maximum score for each query, thus, the score ranges from 0 to 1.) Various types of linking information and the URL length information also were

considered. The logs of all these factors were thrown into the logistic regression analysis. The predicted factor is whether a page is relevant to a query (1) or not (0). The system showed that the log of title retrieval score, title retrieval score, anchor retrieval score, abstract retrieval score, and the reciprocal of the URL length can be used to predict the relevance of a webpage to a query. The correlation is 98%.

3. The formula derived from the logistic regression analysis was then applied to the 98 training queries. 70 queries found the correct answer on top of the list, 96 queries found the correct answer within the top10. The MRR is 0.802, which is 13% better than the title retrieval after non-homepage removal by using the decision tree model (the best of all the runs in the previous stage).

Results of the model can be found in Table 3 and Figure 3.

Table 3. Machine learning model results for training queries

Relevant Found in	Anchor + Tree	Title + Tree	Abstract + Tree	Logistic Regression
Top1	43	62	50	70
Top5	61	83	67	94
Top10	63	84	75	96
Top20	65	86	79	96
Top100	72	92	92	97
Not in list	19	7	4	3
MRR	0.504	0.710	0.597	0.802
Improvement	50% over Anchor	55.7% over Title	90.7% over Abstract	13% over Title + Tree

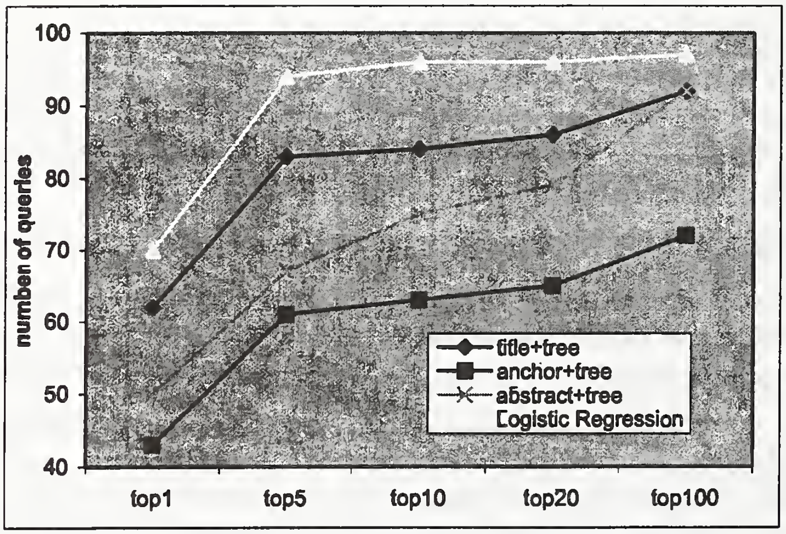


Figure 3. Machine learning model retrieval results comparison chart for training queries

7. TESTING RESULTS AND DISCUSSION

Finally, 145 testing queries provided by TREC were used to evaluate our system. Table 4 and Figure 4 report the retrieval results for the testing queries on title field retrieval with the baseline IR system. From the table we find that testing queries perform substantially worse than training queries. However, on anchor retrieval they perform much better than training queries.

Table 4. Baseline system retrieval results for training queries

Relevant Found in	Title Tag	Anchor Text	Abstract
Top1	38	46	30
Top5	74	71	58
Top10	85	76	66
Top20	92	79	70
Top100	109	90	97
Not in list	17	33	2
MRR	0.378	0.401	0.295

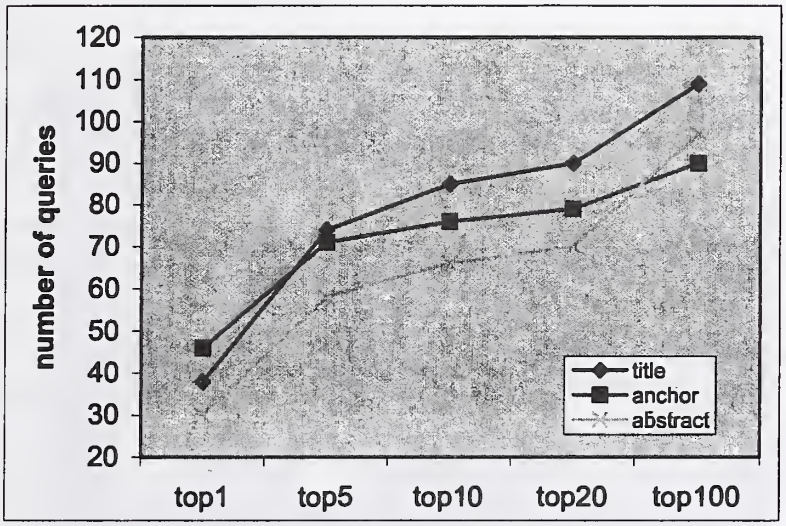


Figure 4. Baseline system retrieval results comparison chart for testing queries

Then, the decision tree model and logistic regression model were applied to the rank lists of the 145 testing queries from the baseline IR system. The results are shown in Table 5 and Figure 5.

Table 5. Machine learning models results for testing queries

Relevant Found in	Anchor + Tree	Title + Tree	Abstract + Tree	Logistic Regression
Top1	68	77	61	96
Top5	81	97	84	118
Top10	84	102	96	121
Top20	88	108	105	128
Top100	98	114	124	130
Not in list	38	27	13	15
MRR	0.511	0.595	0.501	0.727
Improvement	27.4% over Anchor	57.4% over Title	69.8% over Abstract	22.2% over Title + Tree

From Table 5 and Figure 5 we find that the overall performance of the testing queries is much worse than the training queries. This is mainly because 11 testing queries' corresponding correct homepages do not confirm the decision tree model. Thus the correct homepage was filtered out of the rank list by the decision tree step. This greatly affects the final performance.

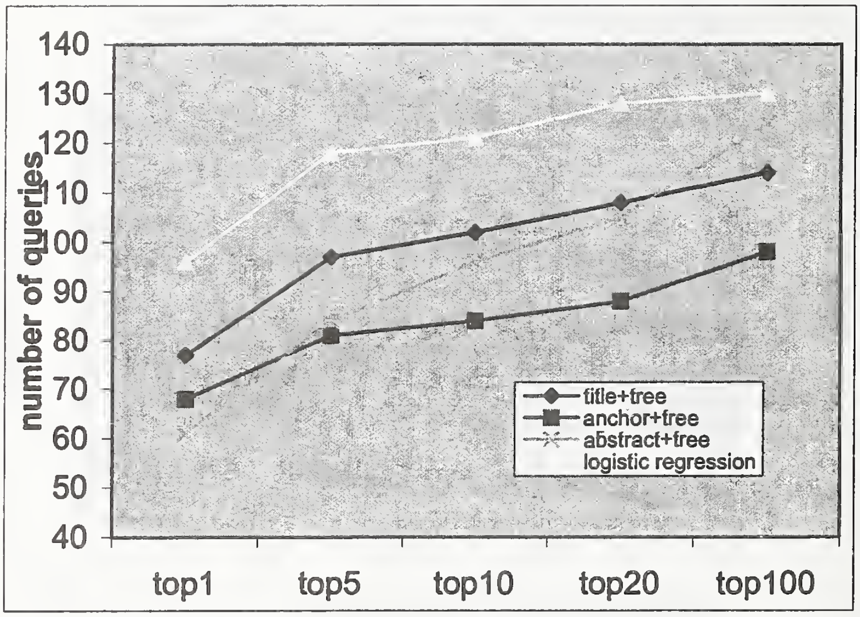


Figure 5. Machine Learning models retrieval results comparison chart for testing queries

After a close examination of these 11 queries, we find that in 3 cases, an argument could be made regarding what should be classified as homepages. For example for query No. 14 "Wah Yew Hotel", the correct answer provided by TREC is

http://www.fastnet.com.au:80/hotels/zone4/my/my00198.htm

Another example: query No.16 "Hotel Grand, Thailand", has correct answer:

http://www.fastnet.com.au:80/hotels/zone4/th/th00635.htm

When we go to the above locations we find each is only an introductory page to Wah Yew Hotel and Hotel Grand, Thailand, in an online hotel index website. It had no links to any other information about these hotels at all. Although this might be the only information about the two hotels on the internet, this may not guarantee itself to be the homepage of these hotels. Actually, common sense would suggest these two pages are not homepages at all.

One more example: query No. 134 "Kaye Bassman International" has correct answer provided by TREC:

http://www.kbic.com:80/toc.htm

However, when you look at the actual page, you will find this is only a table of contents. The homepage of Kaye Bassman International is clearly

http://www.kbic.com:80/index.htm, pointed to by the hyperlink at the table of contents page. These queries lead us to 2 basic questions: What is the definition of a homepage? Can a table of contents also be regarded as a homepage? However, these questions are not easily answered without further research on user behavior on the internet.

8. CONCLUSION AND FUTURE WORK

The conclusions from this research work are:

1. <Title> tagged field retrieval, Anchor text retrieval, and Abstract retrieval all perform substantially better than the full-text retrieval in the context of the homepage finding task. <Title> tagged field text retrieval performs best among these.

2. The decision tree model is an effective machine learning method to filter out homepages. This method can improve the retrieval performance by an average of 50% in terms of MRR.

3. Logistic regression analysis is another effective machine learning approach to combine multiple sources of evidences to improve the retrieval result. Our research results show this method can improve retrieval performance by 13% on training queries and 22% on testing queries.

4. By applying machine learning technologies to our system, our final testing results show 66% of the queries find the correct homepage on top of the return list and 84% of the queries find the correct homepage within the top 10 of the return list.

Future research may include:

1. Further looking into the homepages, finding more attributes that might indicate a homepage. For example, some homepages contain words such as: "welcome", "homepage", "website", "home", "page", in the initial few lines of the text. Incorporating these new factors might help indicate whether a page is a homepage or not.

2. Making use of relevance feedback. The relevance feedback technique is found to be very successful at improving precision for very short queries. Since they are short, homepage finding queries might benefit from this approach.

3. Using a probabilistic rather than a binary decision tree, so likelihood of being a homepage becomes a factor in the logistic regression.

4. Experimenting with large collections to give more thorough and realistic testing of the methods, such as with the 1 terabyte crawling of text recently completed in collaboration with University of Waterloo.

ACKNOWLEDGMENTS

This research work was funded in part by America OnLine, Inc. Thanks go to Dr. A. Chowdhury from AOL for his kind help. Thanks also go to Fernando Das Neves, Hussein Suleman, and other colleagues in the Digital Library Research Laboratory at Virginia Tech for their very useful and important suggestions.

REFERENCES

[1] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Proceeding of the 7th International WWW Conference*, pp.107-117. 1998.
http://www7.scu.edu.au/programme/fullpapers/1921/com1921.htm

[2] A. Chen. "A comparison of regression, neural net, and pattern recognition approaches to IR," in *Proceedings of the 1998 ACM 7th International Conference on Information and Knowledge Management (CIKM '98)* (pp.140-147). New York: ACM, 1998.

[3] N. Craswell; D. Hawking and S. Robertson. Effective Site Finding using Link Anchor Information. *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* pp.250-257. 2001.

[4] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms,* pp.668-677. 1998.
http://**www.cs.cornell.edu/home/kleinber/auth.ps**

[5] M. F. Porter. An algorithm for suffix stripping. Program
14, 130-137. 1980.

[6] T. Sakai and K. Sparck-Jones. Generic Summaries for Indexing
in Information Retrieval. *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* pp.190-198. 2001.

[7] J.A. Shaw & E.A. Fox, "Combination of multiple searches", in *Proceedings of the 3rd Text Retrieval Conference (TREC-3)* (p.105). Gaithersburg, MD: National Institute of Standards and Technology, 1995.

[8] G. Salton. and M. J. McGill. *Introduction to Modern Information Retrieval.* New York: McGraw Hill. 1983.

[9] C.C. Vogt & 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.

[10] E. Voorhees and D.K. Harman. Overview of the Ninth Text Retrieval Conference (TREC-9). In *Proceedings of the Ninth Text Retrieval Conference (TREC-9)*, 1-28. NIST Special Publication pp.1-14. 2000.

Yonsei/ETRI at TREC-10: Utilizing Web Document Properties

Dong-Yul Ra, Eui-Kyu Park and Joong-Sik Jang
Computer Science Dept., Yonsei University
{dyra, ekpark, lunch}@dragon.yonsei.ac.kr

Myung-Gil Jang and Chung Hee Lee
Electronics and Telecommunications Research Institute
{mgjang, forever}@etri.re.kr

Abstract

As our first TREC participation, four runs were submitted for the ad hoc task and two runs for the home page finding task in the web track. For the ad hoc task we experimented on the usefulness of anchor texts. However, no significant gain in retrieval effectiveness was observed. The substring relationship between URL's was found to be effective in the home page finding task.

1. Introduction

This is the first time that our group, Yonsei University and ETRI, participated in the TREC conference. We participated in the web track for both the ad hoc and home page finding tasks. We developed an IR system based on natural language processing according to our original aim. But we could not carry out enough experimentation to draw any conclusion on a NLP-based system. In this paper we will talk about two aspects of a web document retrieval system: taking advantage of the anchor texts of the hyper links and using the substring relationship of URL's in home page finding.

Many reports in TREC-8 and 9 said that the link connectivity itself did not help much to improve the retrieval effectiveness[5,6,8,9]. There have been some suggestions of using the anchor texts of the links[1,2,7]. We thought that a link's anchor text may give some hint on what the document that the link points to is about. As an ad hoc task we developed a system to pursue this issue. The experimental result showed that even the use of anchor texts does not improve the retrieval effectiveness significantly.

We also produced runs related to the home page finding task. What we experimented with this task is the usefulness of the URL substring relationship in finding the home page, i.e. the web site entry page. We have found that if the URL of a document is a prefix of that of another document (where both documents seems to have some relevancy to the topic) the former document is more likely to be the home page than the latter. The experimental observation indicates that the substring relationship of URL's is a good source of information to raise the reciprocal rank (RR) for the home page finding task.

2. Overview of the system

Our system uses a natural language analysis component as the front end for both indexing and retrieval. It consists of morphological analysis, part of speech tagging and the context-free parsing modules. A two-level model is used for morphological analysis. Part of speech tagging is based on the Hidden Markov Model. The bottom-up chart parsing technique is used in the parsing module. It is a shallow parser whose major objective is to find verbs and arguments associated with them. The result of parsing is used to produce the head-modifier index terms whenever it is possible.

The vector space model forms the basis of our system. The index terms can be either key words or a pair of words in head-modifier relationship. Having head-modifier index terms made the number of total index terms huge, which slowed down the speed of the system. The size of the inverted file has grown up to the level that the file system could not handle. This problem was solved by storing the inverted file in several files. This is different from the approach of distributed IR. Our system was developed on the PC of server level with 1GB of memory and 60 GB of disk space. The major amount of time was spent in storing index terms in the indexing storage rather than doing natural language analysis.

This is the first time that we participated in the TREC. We experienced much difficulty in producing the result on time and made some mistakes in the creation of the runs that were submitted for assessment. One non-trivial mistake is that no relevance feedback was done. This might be one of the reasons for coming up with rather low average precision. We hope that we can have better systems by not making mistakes.

3. Experiments in the ad hoc task on usefulness of anchor texts

In this section what we did for the ad hoc task in the web track is explained. We used the typical vector space model for indexing and retrieval. But we tried to make use of information that only web documents can provide. The results of experiments done in the previous TREC conferences pointed out that the use of hyper links does not lead to a noticeable improvement in retrieval effectiveness. But most of the approaches so far just tried to use the information given by the connectivity among documents.

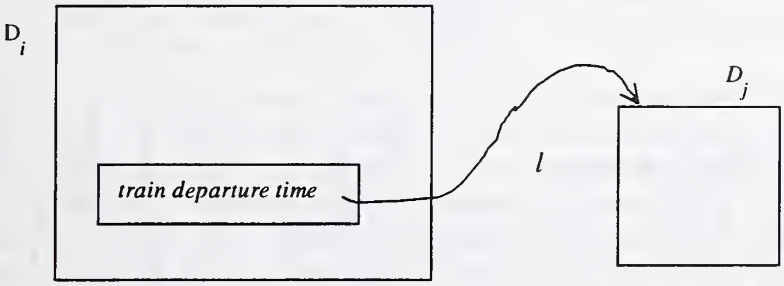


Fig. 1: An anchor text

We thought that the anchor text on which the link is set might be a good source of information.[1] In Fig. 1, the hyper link l connects document D_i and D_j. The anchor text of the link is "*train departure time*". What this anchor text says is that one needs to consult the document D_j to know about *train departure time*; one can find some information about *train departure time* by following the link and reading the document D_j.

Even though the document D_j does not contain any key words indicating relevancy to the topic of train departure time it is likely that the document is relevant to the topic.

Thus the content of a document is reflected in some degree in the anchor texts of the incoming links of the document. But this document receives no contribution to the indication of its content from its outgoing links in our approach. We do not use any information from connectivity such as Kleinberg's scheme except the anchor texts[3].

We cannot consider the links of all documents in the collection because it takes too much time. Let C be the whole collection of the documents. The consideration of links is confined to the documents retrieved for the query by

[1] After we started development with this aim. we found later that several organizations pursued this issue independently[1.2.7].

Table 1: Performance of our system in the ad hoc task

Run id: yeaht01	Run description: automatic, title-only, link(anchor text)		No. of topics: 50
Total number of documents over all topics			
Retrieved: 44922	Relevant: 3363		Relevants retrieved: 1337
Recall level precision averages		Document level precision averages	
Recall	Precision	Recall	Precision
0.0	0.6152	At 5 docs	0.3880
0.1	0.3619	At 10 docs	0.3240
0.2	0.2511	At 15 docs	0.2800
0.3	0.1820	At 20 docs	0.2520
0.4	0.0998	At 30 docs	0.2180
0.5	0.0616	At 100 docs	0.1282
0.6	0.0286	At 200 docs	0.0830
0.7	0.0225	At 500 docs	0.0473
0.8	0.0200	At 1000 docs	0.0267
0.9	0.0200		
1.0	0.0200		
Average precision (non-interpolated) : 0.1286		R-precision (exact) : 0.1796	

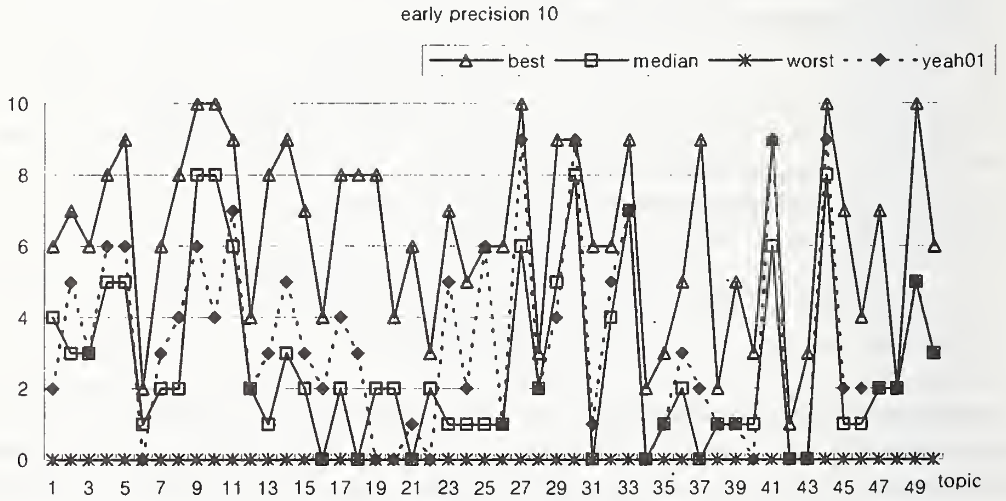


Fig. 2: Performance comparison with others

the typical retrieval engine based on the vector space model[4]. (Let us call it the base set B of the retrieved documents.) The extended set E of the retrieved documents is obtained as follows. Here, $\Phi(l)$ indicates the anchor text of link l; Q denotes the query; sim(Q, $\Phi(l)$) is the similarity[2] between Q and $\Phi(l)$ returned by the retrieval

[2] The score sim(A,B) stands for the cosine similarity value between the vectors of two texts A and B returned by the vector space model.

engine:

```
(i)   $E \leftarrow B$;
(ii)  For each document $d_i$ in $E$,
          add $d_j$ to $E$ if there is a link $l$ out of $d_i$ pointng to $d_j$ and sim($Q$, $\Phi(l)$) > 0 ;
```

Then the score of each document in E is computed again by the following two methods that are different in some way.

- Method link1: The new relevancy score of each document in E is computed as follows:

$$RSV(d) = sim(Q,d) + \alpha \sum_{l \in inlink(d)} sim(Q, \Phi(l))$$

where *inlink(d)* is the set of incoming links to document d. The parameter α is the weight given to the contribution of the anchor texts. It is determined by experiments.

- Method link2: In this method the anchor text is regarded as the part of the text of the document.
  (i) We add all anchor texts of incoming links of every document in the extended set E as a part of the document.
  (ii) We do indexing on a document including the anchor texts. (However a special scheme is used to include only the anchor texts of the incoming links from the documents in the base set B.)
  (iii) The similarity score returned by the vector space model is used for obtaining the final ranked list.

The final ranked list of retrieved documents is obtained by ordering the documents in E based on the *RSV* of each document. We could not submit the official runs using Method link2 because of the tight schedule.

One can see the performance of our system in the ad hoc task of the web track in Table 1. Early precision seems to be important in IR systems. The comparison with other systems in this measure can be seen in Fig. 2. This shows that our system is near median. Table 2 shows the difference made by the use of anchor texts. The run yeaht01 (automatic, title only, use of anchor texts) does not have any significant improvement from the run yeahtb01 (automatic, title only, no use of anchor texts).

Table 2: Effectiveness of the use of anchor texts

Recall	Average precision	
	yeahtb01 (no use of links)	yeaht01 (use of links)
0.0	0.6086	0.6152
0.1	0.3618	0.3619
0.2	0.2534	0.2511
0.3	0.1796	0.1820
0.4	0.1002	0.0998
0.5	0.0618	0.0616
0.6	0.0286	0.0286
0.7	0.0225	0.0225
0.8	0.0200	0.0200
0.9	0.0200	0.0200
1.0	0.0200	0.0200

4. The use of substring relationships of URL's for home page finding

A document of a home page (the entry page) of a web site has the same format as other web pages. There is no information or marks attached to the web pages indicating whether it is a home page or not. Thus it is not easy to locate a home page for a web site search query.

We use a heuristic to cope with this problem. There is a tendency that if a home page D_h has an outgoing link to a page D_i and D_i is stored physically in the same server as the home page then the URL string of D_h is a substring (actually a prefix) of D_i's URL.

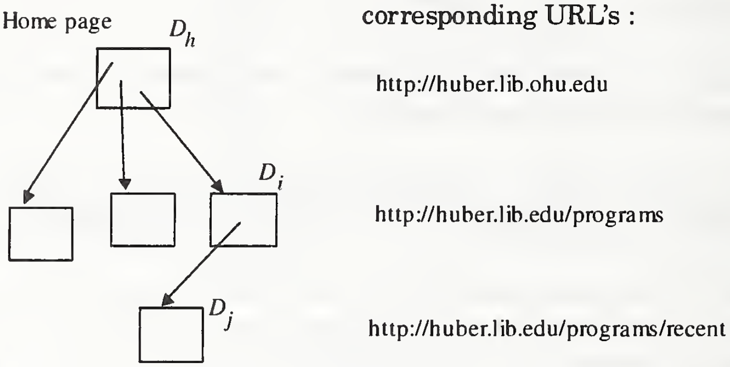


Fig. 3 : Web pages and their URL's

Those pages that are descendants of the entry page D_h will have a tendency that their URL's contain the URL of D_h as shown in Fig. 3. As an example let us assume that the home page finding query is "Huber Library" and retrieval process explained in the previous section produces the following ranked list of documents:

D_j	(http://huber.lib.edu/programs/recent) :	17.5
D_i	(http://huber.lib.edu/programs/) :	14.3
D_h	(http://huber.lib.edu) :	11.8

It is likely that the bottom-most document D_j contains the words "Huber" and/or "Library" more number of times than its ancestors D_i and D_h. Thus the score of D_j is highest. Since URL of D_h is a substring of that of the document D_j in the retrieval list it is given some bonus point , say 4. D_h gets the bonus once more because of D_i by the same reason. Thus the score of D_h will be increased to 19.8. Similarly D_i gets the bonus point of 4 because URL of D_j subsumes that of D_i. But D_j gets no bonus because there is no document whose URL string contains that of D_j. As a result the final score and the ranked list is as follows:

D_h :	(http://huber.lib.edu) :	19.8
D_i :	(http://huber.lib.edu/programs/) :	18.3
D_j :	(http://huber.lib.edu/programs/recent) :	17.5

We take advantage of this observation explained so far to move a home page up in the ranked list (which was called *E*) of the retrieval result. We apply the following heuristic for all the pages in the final ranked list *E* (produced in the ad-hoc processing explained in the previous section):

(i) Every document *d* in *E* gets extra bonus point (added to the existing score) whenever there is a document *b* in *E* such that URL of *d* is a part (substring) of URL of *b*;
(ii) After this processing is done for all the documents in *E*, they are reordered by the new scores.

We submitted two runs for the home page finding task. The assessment for the run yehp01 (that is automatic, uses anchor texts, and uses URL substring heuristic) is as follows:

Average reciprocal rank over 145 topics	0.669
Number of topics for which entry page found in top 10	111 (76.6%)
Number of topics for which no entry page was found	32 (22.1%)

The graph showing the reciprocal ranks (RR's) of the home pages for all 145 topics is shown in Fig. 4. Most of the answers are at rank 1 for the queries for which the home pages are included in the ranked list of 100 documents. The result of subtracting median's RR from the RR of our system for each query is plotted in Fig. 5. It can be said that our system belongs to a class of systems which show high performance in home page finding.

Table 3 is given to illustrate the effectiveness of the heuristic based on URL substring relationship. (The mark "url" in the table indicates that the run used the URL heuristic; "Base" is used to indicate a run not using link information; "Link1" for a run using method link1; "Link2" for using method link2.) We can notice that the performance of runs with the URL heuristic is 3 to 4 times better than the corresponding runs without the heuristic when only the document at rank 1 is considered. It can be seen that the use of link information (actually anchor texts in our method) along with the URL heuristic improves the performance when documents of rank 10 or more are included for consideration. However, the data says that using anchor texts only for home page finding did not result in any performance improvement, which does not agree with the suggestion given in [1].

5. Summary

We participated in the web track of TREC-10. We submitted runs for both ad hoc and home page finding tasks. For the ad hoc task we investigated the effectiveness of utilizing the anchor texts. However, we obtained the same result on this issue as the reports in TREC-9 stating that the anchor texts does not enable the systems to achieve significant improvement in retrieval effectiveness. A heuristic called the URL substring relationship was studied in the home page finding task. It is based on the observation that the URL of a home page is a substring of the URL's of web pages in the same site. The use of this heuristic was found to be effective in making the system to be able to move the home page toward the topmost rank.

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.

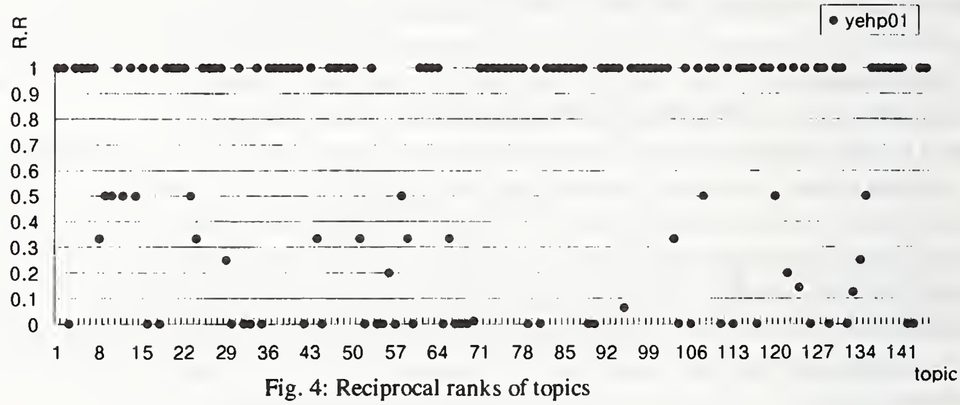


Fig. 4: Reciprocal ranks of topics

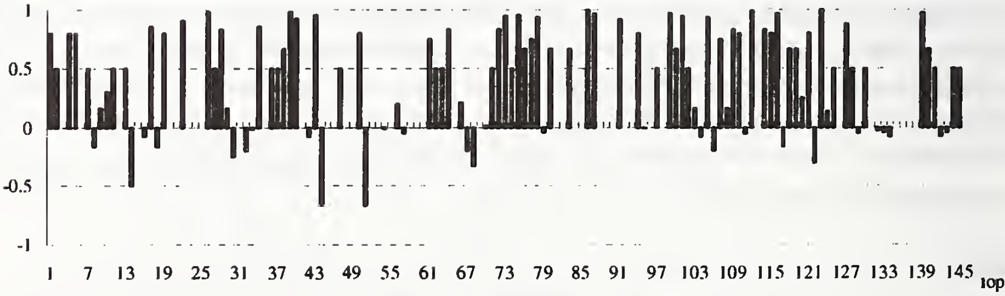


Fig. 5: Difference from median in reciprocal rank

Table 3: Runs using the heuristic of URL substring relationship

Rank	Number of queries with answer within the rank					
	Base	Base/url	Link1	Link1/url	Link2	Link2/url
1	29	80	32	69	30	70
5	65	104	66	106	63	104
10	76	107	76	115	75	115
50	105	112	105	124	103	123
100	111	115	112	125	112	124

TREC 2001 Results

APPENDIX A

This appendix contains the evaluation results for TREC 2001 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
BBN10MON	BBN Technologies	Arabic	Automatic, title+desc
BBN10XLA	BBN Technologies	English	Automatic, title+desc
BBN10XLB	BBN Technologies	English	Automatic, title+desc
BBN10XLC	BBN Technologies	English	Automatic, title+desc
BBN10XLD	BBN Technologies	English	Automatic, title+desc+narr
humAR01t	Hummingbird	Arabic	Automatic, title
humAR01td	Hummingbird	Arabic	Automatic, title+desc
humAR01tdm	Hummingbird	Arabic	Automatic, title+desc
humAR01tdn	Hummingbird	Arabic	Automatic, title+desc+narr
humAR01tdx	Hummingbird	Arabic	Automatic, title+desc
iit01md	IIT	Arabic	Automatic, title+desc
iit01ml	IIT	Arabic	Automatic, title+desc
iit01mlr	IIT	Arabic	Automatic, title+desc
iit01xdi	IIT	English	Automatic, title+desc
iit01xma	IIT	English	Automatic, title+desc
apl10ca1	Johns Hopkins Univ., APL	Arabic	Automatic, title+desc+narr
apl10ce1	Johns Hopkins Univ., APL	English	Automatic, title+desc+narr
apl10ce2	Johns Hopkins Univ., APL	English	Automatic, title+desc
apl10ce3	Johns Hopkins Univ., APL	English	Automatic, title+desc
apl10cf1	Johns Hopkins Univ., APL	French	Automatic, title+desc+narr
NMMLTN	New Mexico State Univ.	Arabic	Automatic, title
NMCLDS	New Mexico State Univ.	English	Automatic, desc
NMCLMN	New Mexico State Univ.	English	Manual
NMCLTNv1	New Mexico State Univ.	English	Automatic, title
NMCLTS	New Mexico State Univ.	English	Automatic, title
pir1XAtd	Queens College, CUNY	Arabic	Automatic, title+desc
pir1XAtdn	Queens College, CUNY	Arabic	Automatic, title+desc+narr
pir1XEtd	Queens College, CUNY	English	Automatic, title+desc
pir1XEtdn	Queens College, CUNY	English	Automatic, title+desc+narr
BKYAAA1	Univ. of Calif./Berkeley	Arabic	Automatic, title+desc+narr
BKYEAA1	Univ. of Calif./Berkeley	English	Automatic, title+desc+narr
BKYEAA2	Univ. of Calif./Berkeley	English	Automatic, title+desc+narr
BKYEAA3	Univ. of Calif./Berkeley	English	Automatic, title+desc+narr
BKYEAA4	Univ. of Calif./Berkeley	English	Automatic, title+desc
UMmanual	Univ. of Maryland	Arabic	Manual
UMmonoAuto	Univ. of Maryland	Arabic	Automatic, title+desc
UMclirAutoFL	Univ. of Maryland	English	Automatic, title+desc
UMclirAutoTJ	Univ. of Maryland	English	Automatic, title+desc
UMclirAutoXP	Univ. of Maryland	English	Automatic, title+desc
UMass1	Univ. of Massachusetts	Arabic	Automatic, title+desc
UMass2	Univ. of Massachusetts	Arabic	Automatic, title+desc
UMass3	Univ. of Massachusetts	English	Automatic, title+desc
UMass4	Univ. of Massachusetts	Arabic	Automatic, title+desc
shefma	Univ. of Sheffield	Arabic	Automatic, title
shefea	Univ. of Sheffield	English	Automatic, title
shefeaa	Univ. of Sheffield	English	Automatic, title
sheffea	Univ. of Sheffield	French	Automatic, title
sheffeaa	Univ. of Sheffield	French	Automatic, title

Filtering Track, Adaptive Task

Tag	Organization	Resourses Used	Optimized For
ICTAdaFT10Fa	Chinese Academy of Sciences	None	T10U
ICTAdaFT10Ua	Chinese Academy of Sciences	None	T10U
ICTAdaFT10Ub	Chinese Academy of Sciences	None	T10U
ICTAdaFT10Uc	Chinese Academy of Sciences	None	T10U
CL01afa	Clairvoyance Corp.	None	T10U
CL01afb	Clairvoyance Corp.	None	T10U
CL01afc	Clairvoyance Corp.	None	T10U
CL01afd	Clairvoyance Corp.	None	T10U
CMUCATmr10	Carnegie Mellon Univ.	Other TREC	T10U
CMUCATmrf5	Carnegie Mellon Univ.	Other TREC	F-beta
CMUCATsr10	Carnegie Mellon Univ.	Other TREC	T10U
CMUCATsrf5	Carnegie Mellon Univ.	Other TREC	F-beta
CMUDIRFLM	Carnegie Mellon Univ.	None	F-beta
CMUDIRFRM	Carnegie Mellon Univ.	None	F-beta
CMUDIRULM	Carnegie Mellon Univ.	None	T10U
CMUDIRURM	Carnegie Mellon Univ.	None	T10U
FDUT10AF1	Fudan Univ.	None	T10U
FDUT10AF4	Fudan Univ.	None	F-beta
krdlT10af01	Kent Ridge Digital Labs	Other TREC, other data	Neither
ok10f2br	Microsoft Research Ltd.	None	F-beta
ok10f2ur	Microsoft Research Ltd.	None	T10U
KUNaF	Katholieke Univ. Nijmegen	None	F-beta
KUNaU	Katholieke Univ. Nijmegen	None	T10U
oraAU082201	Oracle	None	T10U
RUA301	Rutgers Univ.	None	Neither
RUA501	Rutgers Univ.	None	Neither
serASSAT10ad	SER Tech. Deutschland GmbH	None	Neither
serCLST10af	SER Tech. Deutschland GmbH	None	Neither
UIowa01AF01	Univ. of Iowa	Other TREC, other data	T10U
UIowa01AF02	Univ. of Iowa	Other TREC, other data	T10U

Filtering Track, Batch Task

Tag	Organization	Resourses Used	Optimized For
apl10fbsvml	Johns Hopkins Univ., APL	None	T10U
apl10fbsvmr	Johns Hopkins Univ., APL	None	T10U
CLT10BFA	Clairvoyance Corp.	None	T10U
CLT10BFB	Clairvoyance Corp.	None	T10U
CMUCATa210	Carnegie Mellon Univ.	Other TREC	T10U
CMUCATa2f5	Carnegie Mellon Univ.	Other TREC	F-beta
FDUT10BF1	Fudan Univ.	None	T10U
FDUT10BF2	Fudan Univ.	None	F-beta
mer10b	IRIT/SIG	Other TREC	T10U
KAIST10bf1	KAIST	None	neither
KAIST10bf2	KAIST	None	neither
DLewis01bfFa	David Lewis, Consultant	None	F-beta
DLewis01bfUa	David Lewis, Consultant	None	T10U
KUNbF	Katholieke Univ. Nijmegen	None	F-beta
KUNbU	Katholieke Univ. Nijmegen	None	T10U
oraBU082701	Oracle	None	T10U
serASSAT10ba	SER Tech. Deutschland GmbH	None	Neither
serCLST10ba	SER Tech. Deutschland GmbH	None	Neither

Filtering Track, Routing Task

Tag	Organization	Resourses Used	Optimized For
apl10frn	Johns Hopkins Univ., APL	None	Neither
apl10frs	Johns Hopkins Univ., APL	None	Neither
clT10rta	Clairvoyance Corp.	None	Neither
clT10rtb	Clairvoyance Corp.	None	Neither
mer10r1	IRIT/SIG	Other TREC	Neither
jscbtafr1	Justsystem Corp.	Other TREC	Neither
jscbtafr2	Justsystem Corp.	Other TREC	Neither
DLewis01rFA	David Lewis, Consultant	None	F-beta
DLewis01rUa	David Lewis, Consultant	None	T10U
MMAT10rout	Moscow Medical Academy	Other data	Neither
KUNr1	Katholieke Univ. Nijmegen	None	Neither
KUNr2	Katholieke Univ. Nijmegen	None	Neither
oraRO082801	Oracle	None	neither
RUGR01	Rutgers Univ.	None	T10U
serASSAT10ro	SER Tech. Deutschland GmbH	None	Neither
serCLST10ro	SER Tech. Deutschland GmbH	None	Neither
VisaSent1T10	Tampere Univ. of Tech.	None	Neither
VisaWordT10	Tampere Univ. of Tech.	None	Neither

Question Answering Track, Context

Tag	Organization
clr01c1	CL Research
KAISTQACTX	KAIST
LCC3	Language Computer Corp.
qntuac1	National Taiwan Univ.
pir1Qctx2	Queens College-Cuny
pir1Qctx3	Queens College-Cuny
uwmtac0	University of Waterloo

Question Answering Track, List

Tag	Organization
clr0111	CL Research
clr0112	CL Research
KAISTQALIST1	KAIST
KAISTQALIST2	KAIST
LCC2	Language Computer Corp.
qntual1	National Taiwan Univ.
qntual2	National Taiwan Univ.
pirlQli1	Queens Collge-Cuny
pirlQli2	Queens Collge-Cuny
SUT10DOCLT	Syracuse Univ.
SUT10PARLT	Syracuse Univ.
UAmsT10qaL1	Univ. of Amsterdam/ILLC
UAmsT10qaL2	Univ. of Amsterdam/ILLC
UdeMlistB	Univ. of Montreal
UdeMlistP	Univ. of Montreal
isi1l50	Univ. of Southern California/ISI
uwmta10	Univ. of Waterloo
uwmta11	Univ. of Waterloo

Question Answering Track, Main

Tag	Organization
ALICO1M1	Alicante
ALICO1M2	Alicante
ICTQA10a	Chinese Academy of Sciences
ICTQA10b	Chinese Academy of Sciences
ICTQA10c	Chinese Academy of Sciences
clr01b1	CL Research
clr01b2	CL Research
cnxdepg1	Conexor-Oy
ecw1csx	EC Wise, Inc.
ecw1ps	EC Wise, Inc.
FDUT10Q2	Fudan Univ.
HITIRGEQA01M	Harbin Institute of Technology
ibmsqa01a	IBM-Franz
ibmsqa01b	IBM-Franz
ibmsqa01c	IBM-Franz
IBMKS1M1	IBM-Prager
IBMKS1M2	IBM-Prager
IBMKS1M3	IBM-Prager
insight	InsightSoft
irstqa01	ITC-irst
KAISTQAMAIN1	KAIST
KAISTQAMAIN2	KAIST
KCSL10QA1	KCSL
kuqa1	Korea Univ.
kuqa2	Korea Univ.
LCC1	Language Computer Corp.
QALIR1	Limsi
QALIR2	Limsi
QALIR3	Limsi
askmsr2	Microsoft
askmsr	Microsoft
MITRE01A	The Mitre Corp.
nttcs10main	NTT Comm. Science Labs (Kazawa)
qntuam1	National Taiwan Univ.
qntuam2	National Taiwan Univ.
orcl1	Oracle
posqa10a	Postech
pir1Qqa1	Queens College-Cuny
pir1Qqa2	Queens College-Cuny
pir1Qqa3	Queens College-Cuny
mtsuna0	Sun Microsystems Labs
mtsuna1	Sun Microsystems Labs
SUT10DOCMT	Syracuse Univ.
SUT10PARMT	Syracuse Univ.
TilburgILKs	Tilburg Univ.
TilburgILK	Tilburg Univ.
gazoo	Univ. of Alberta
UAmsT10qaM1	Univ. of Amsterdam/ILLC
UAmsT10qaM2	Univ. of Amsterdam/ILLC
UAmsT10qaM3	Univ. of Amsterdam/ILLC

Question Answering Track, Main (continued)

Tag	Organization
UIUC	Univ. of Illinois/Champaign
UIowaQA011	Univ. of Iowa
UIowaQA012	Univ. of Iowa
UdeMmainOk80	Univ. of Montreal
UdeMmainOk60	Univ. of Montreal
UdeMmainQt80	Univ. of Montreal
P50x	Univ. of Pennsylvania
prun001	Univ. of Pisa
prun002	Univ. of Pisa
prun003	Univ. of Pisa
isi1a50	Univ. of Southern California/ISI
ISI1b50	Univ. of Southern California/ISI
uwmta0	Univ. of Waterloo
uwmta1	Univ. of Waterloo
uwmta2	Univ. of Waterloo
yorkqa01	Univ. of York
yorkqa02	Univ. of York

TREC-2001 Video Track - Key to Results Pages

The following pages provide a overview of the Video Track participants, the topics, and the performance of the systems on the shot boundary determination and search tasks. Here is how things are arranged:

1. Shot boundary determination - groups, system ids, and systems
2. Shot boundary determination results
3. Tabular overview of topics
4. General and known-item search - groups, system ids, and systems
5. General search results
6. Known-item search results

Shot boundary determination - groups, system ids, and systems

- CLIPS IMAGE, France
  - Sys01 - CLIPS-1
  - Sys02 - CLIPS-2
- Fudan University, China
  - Sys03 - FudanSys1
  - Sys04 - FudanSys2
- Dublin City University, Ireland
  - Sys05 - FísChlár1
  - Sys06 - FísChlár2
- IBM Almaden Research Center, USA
  - Sys07 - IBM_Alm1
  - Sys08 - IBM_Alm2
- Imperial College - London, UK
  - Sys09 - ICKM1
- Johns Hopkins University, USA
  - Sys10 - JHUAPL_1
- Glasgow University, UK
  - Sys11 - MB_Frequency
- Microsoft Research, China
  - Sys12 - MSSD
- University of Amsterdam and TNO, the Netherlands
  - Sys13 - MediaMill1
  - Sys14 - MediaMill2
- University of Maryland, USA
  - Sys15 - UMDLAMP

Web Track, ad hoc Task

Tag	Organization	Run-Type	Document Structure	URL Text	Link Structure
ajouai0101	Ajou Univ.	Automatic, title only	No	No	No
ajouai0103	Ajou Univ.	Automatic, title only	No	No	Yes
Lemur	Carnegie Mellon Univ.	Automatic, title only	No	No	No
ictweb10n	Chinese Academy of Sci.	Automatic, title only	No	No	No
ictweb10nf	Chinese Academy of Sci.	Automatic, title only	No	No	No
ictweb10nfl	Chinese Academy of Sci.	Automatic, title only	No	No	Yes
ictweb10nl	Chinese Academy of Sci.	Automatic, title only	No	No	Yes
csiro0mwa1	CSIRO	Manual	Yes	Yes	Yes
csiro0awa1	CSIRO	Automatic, title only	Yes	Yes	Yes
csiro0awa2	CSIRO	Automatic, title only	Yes	Yes	Yes
csiro0awa3	CSIRO	Automatic, title only	Yes	Yes	Yes
fub01be2	Fondazione Ugo Bordoni	Automatic, title only	No	No	No
fub01idf	Fondazione Ugo Bordoni	Automatic, title only	No	No	No
fub01ne	Fondazione Ugo Bordoni	Automatic, title only	No	No	No
fub01ne2	Fondazione Ugo Bordoni	Automatic, title only	No	No	No
fdut10wac01	Fudan Univ.	Automatic, adhoc, title+desc+narr	No	No	No
fdut10wal01	Fudan Univ.	Automatic, adhoc, title+desc+narr	No	No	Yes
fdut10wtc01	Fudan Univ.	Automatic, title only	No	No	No
fdut10wtl01	Fudan Univ.	Automatic, title only	No	No	Yes
flabxtd	Fujitsu	Automatic, adhoc, title+desc	No	No	No
flabxtdn	Fujitsu	Automatic, adhoc, title+desc+narr	No	No	No
Flabxt	Fujitsu	Automatic, title only	No	No	No
flabxtl	Fujitsu	Automatic, title only	No	No	Yes
hum01tdlx	Hummingbird	Automatic, adhoc, title+desc	Yes	No	No
hum01t	Hummingbird	Automatic, title only	Yes	No	No
hum01tl	Hummingbird	Automatic, title only	Yes	No	No
hum01tlx	Hummingbird	Automatic, title only	Yes	No	No
ARCJ0	IBM Almaden Res. Ctr.	Automatic, title only	Yes	No	No
ARCJ5	IBM Almaden Res. Ctr.	Automatic, title only	Yes	No	Yes
JuruFull	IBM (Haifa)	Automatic, title only	Yes	Yes	No
JuruFullQE	IBM (Haifa)	Automatic, title only	Yes	Yes	No
JuruPrune005	IBM (Haifa)	Automatic, title only	Yes	Yes	No
JuruPruned01	IBM (Haifa)	Automatic, title only	Yes	Yes	No
iit01m	IIT	Manual	No	No	No
iit01tde	IIT	Automatic, adhoc, title+desc	No	No	No
iit01t	IIT	Automatic, title only	No	No	No
iit01tfc	IIT	Automatic, title only	No	No	No
icadhoc1	Imperial College	Automatic, title only	No	No	Yes
icadhoc2	Imperial College	Automatic, title only	No	No	Yes
icadhoc3	Imperial College	Automatic, title only	No	No	No
Merxtd	IRIT/SIG	Automatic, adhoc, title+desc	No	No	No
Merxt	IRIT/SIG	Automatic, title only	No	No	No
apl10wd	Johns Hopkins Univ., APL	Automatic, adhoc, title+desc+narr	No	No	No
apl10wa	Johns Hopkins Univ., APL	Automatic, title only	No	No	No
apl10wb	Johns Hopkins Univ., APL	Automatic, title only	No	No	No
apl10wc	Johns Hopkins Univ., APL	Automatic, title only	No	No	No
jscbtawtl1	Justsystem Corp.	Automatic, title only	Yes	No	Yes
jscbtawl2	Justsystem Corp.	Automatic, title only	Yes	No	Yes

Web Track, ad hoc Task (continued)

Tag	Organization	Run-Type	Document Structure	URL Text	Link Structur
jscbtawt13	Justsystem Corp.	Automatic, title only	Yes	No	Yes
jscbtawt14	Justsystem Corp.	Automatic, title only	Yes	No	Yes
kuadhoc2001	Kasetsart Univ.	Automatic, adhoc, title+desc+narr	No	No	No
msrcn1	Microsoft Research China	Automatic, title only	Yes	Yes	No
msrcn2	Microsoft Research China	Automatic, title only	Yes	Yes	Yes
msrcn3	Microsoft Research China	Automatic, title only	Yes	Yes	No
msrcn4	Microsoft Research China	Automatic, title only	Yes	Yes	No
ok10wtnd0	Microsoft Research Ltd.	Automatic, adhoc, title+desc+narr	No	Yes	No
ok10wtnd1	Microsoft Research Ltd.	Automatic, adhoc, title+desc+narr	No	Yes	No
ok10wt1	Microsoft Research Ltd.	Automatic, title only	No	Yes	No
ok10wt3	Microsoft Research Ltd.	Automatic, title only	No	Yes	No
Ntvenx1	NexTrieve BV	Automatic, title only	Yes	No	No
Ntvenx2	NexTrieve BV	Automatic, title only	Yes	No	No
Ntvfnx3	NexTrieve BV	Automatic, title only	Yes	No	No
Ntvfnx4	NexTrieve BV	Automatic, title only	Yes	No	No
posnir01ptd	POSTECH	Automatic, adhoc, title+desc	Yes	No	No
posnir01pt	POSTECH	Automatic, title only	Yes	No	No
posnir01rpt	POSTECH	Automatic, title only	Yes	No	No
posnir01st	POSTECH	Automatic, title only	Yes	No	No
pir1Wa	Queens College, CUNY	Automatic, adhoc, title+desc+narr	No	No	No
pir1Wt1	Queens College, CUNY	Automatic, title only	No	No	No
pir1Wt2	Queens College, CUNY	Automatic, title only	No	No	No
ricAP	RICOH Co., Ltd.	Automatic, title only	No	No	No
ricMM	RICOH Co., Ltd.	Automatic, title only	No	No	No
ricMS	RICOH Co., Ltd.	Automatic, title only	No	No	No
ricST	RICOH Co., Ltd.	Automatic, title only	No	No	No
tnout10t1	TNO-TPD & Univ. of Twente	Automatic, title only	No	No	No
tnout10t2	TNO-TPD & Univ. of Twente	Automatic, title only	No	No	No
UniNEn7d	Univ. of Neuchatel	Automatic, adhoc, title+desc+narr	No	No	No
UniNEt7dL	Univ. of Neuchatel	Automatic, title only	No	No	No
UniNEtd	Univ. of Neuchatel	Automatic, title only	No	No	No
UniNEtdL	Univ. of Neuchatel	Automatic, title only	No	No	No
irtLnua	Univ. NC/Chapel Hill-Newby	Automatic, title only	Yes	No	No
irtLnut	Univ. NC/Chapel Hill-Newby	Automatic, title only	Yes	No	No
uncfslm	Univ. NC/Chapel Hill-Yang	Automatic, adhoc, title+desc	No	No	Yes
uncvsmm	Univ. NC/Chapel Hill-Yang	Automatic, adhoc, title+desc	No	No	No
uncfsls	Univ. NC/Chapel Hill-Yang	Automatic, title only	No	No	Yes
uncvsms	Univ. NC/Chapel Hill-Yang	Automatic, title only	No	No	No
PDWTAHDR	Univ. of Padova	Automatic, title only	No	No	No
PDWTAHPR	Univ. of Padova	Automatic, title only	No	No	No
PDWTAHTL	Univ. of Padova	Automatic, title only	No	No	Yes
PDWTAHWL	Univ. of Padova	Automatic, title only	No	No	Yes
uwmtaw0	Univ. of Waterloo	Automatic, title only	No	No	No
uwmtaw1	Univ. of Waterloo	Automatic, title only	No	No	No
uwmtaw2	Univ. of Waterloo	Automatic, title only	No	No	No
yeahdb01	Yonsei Univ.-Ra	Automatic, adhoc, title+desc	Yes	No	No
yeahtd01	Yonsei Univ.-Ra	Automatic, adhoc, title+desc	Yes	No	Yes
yeaht01	Yonsei Univ.-Ra	Automatic, title only	Yes	No	Yes
yeahtb01	Yonsei Univ.-Ra	Automatic, title only	Yes	No	No

Web Track, Entry Page Finding Task

Tag	Organization	Document Structure	URL Text	Link Structure
ajouai0102	Ajou Univ.	No	No	No
ajouai0104	Ajou Univ.	No	No	Yes
csiro0awh1	CSIRO	Yes	Yes	Yes
csiro0awh2	CSIRO	No	No	Yes
flabxe75a	Fujitsu	Yes	Yes	Yes
flabxeall	Fujitsu	No	No	Yes
flabxemerge	Fujitsu	Yes	Yes	Yes
flabxet256	Fujitsu	Yes	No	Yes
IBMHOMENR	IBM Alamaden Res. Ctr.	Yes	No	No
IBMHOMER	IBM Alamaden Res. Ctr.	Yes	No	Yes
ichp1	Imperial College	No	No	Yes
ichp2	Imperial College	No	No	No
iit01st	IIT	Yes	Yes	No
iit01stb	IIT	Yes	Yes	Yes
apl10ha	Johns Hopkins Univ., APL	No	No	No
apl10hb	Johns Hopkins Univ., APL	No	No	No
jscbtawep1	Justsystem Corp.	Yes	Yes	Yes
jscbtawep2	Justsystem Corp.	Yes	Yes	Yes
jscbtawep3	Justsystem Corp.	Yes	Yes	Yes
jscbtawep4	Justsystem Corp.	Yes	Yes	Yes
kuhpf2001	Kasetsart Univ.	No	No	No
msrcnp1	Microsoft Research China	Yes	Yes	No
msrcnp2	Microsoft Research China	Yes	Yes	Yes
ok10wahd0	Microsoft Research Ltd.	No	Yes	Yes
ok10wahd1	Microsoft Research Ltd.	No	Yes	Yes
ok10whd0	Microsoft Research Ltd.	No	Yes	No
ok10whd1	Microsoft Research Ltd.	No	Yes	No
tnout10epA	TNO-TPD & Univ. of Twente	No	No	Yes
tnout10epC	TNO-TPD & Univ. of Twente	No	No	No
tnout10epCAU	TNO-TPD & Univ. of Twente	No	Yes	Yes
tnout10epCU	TNO-TPD & Univ. of Twente	No	Yes	No
UniNEep1	Univ. of Neuchatel	No	Yes	No
UniNEep2	Univ. of Neuchatel	No	Yes	No
UniNEep3	Univ. of Neuchatel	No	No	No
UniNEep4	Univ. of Neuchatel	No	Yes	No
PDWTEPDR	Univ. of Padova	No	No	No
PDWTEPPR	Univ. of Padova	No	No	No
PDWTEPTL	Univ. of Padova	No	No	Yes
PDWTEPWL	Univ. of Padova	No	No	Yes
VTBASE	Virginia Tech	No	No	No
VTEP	Virginia Tech	No	Yes	No
yehp01	Yonsei Univ.	Yes	Yes	Yes
yehpb01	Yonsei Univ.	Yes	Yes	No

1 Common Evaluation Measures

- Recall
  A measure of the ability of a system to present all relevant items.

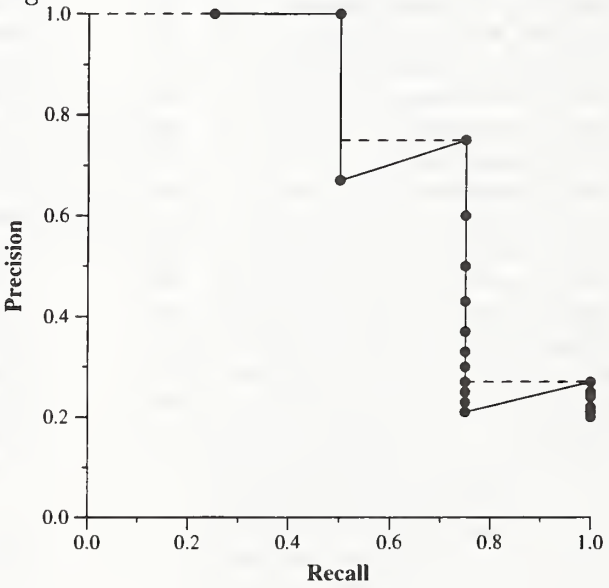
$$\text{recall} = \frac{\text{number of relevant items retrieved}}{\text{number of relevant items in collection}}$$

- 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 ad hoc 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.

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

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

recall level λ) and then dividing by the number of topics.

$$\frac{\sum_{i=1}^{NUM} P_\lambda}{NUM} \qquad \lambda = \{0.0, 0.1, 0.2, 0.3, \ldots, 1.0\}$$

- 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.

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

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

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.

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.

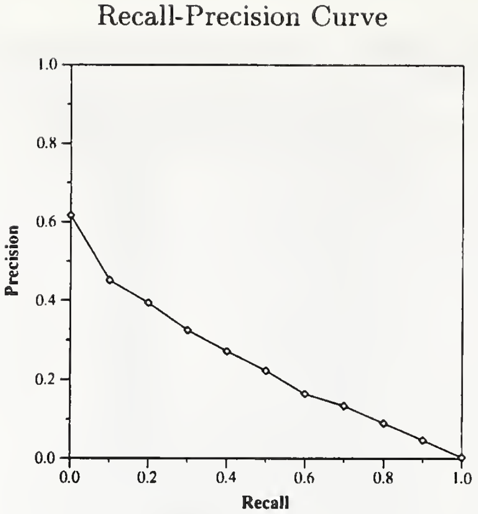


Figure 1: Sample Recall-Precision Graph.

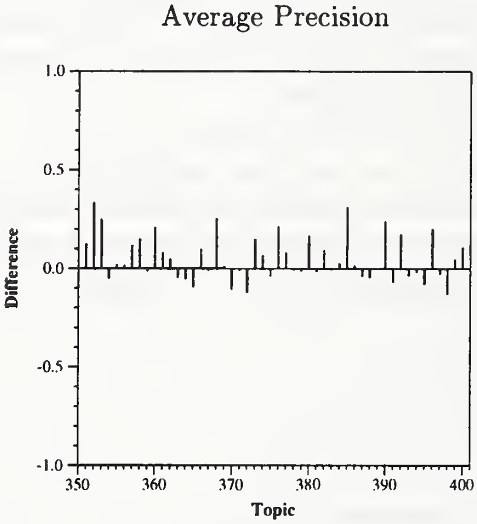


Figure 2: Sample Average Precision Histogram.

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.

3 Question Answering Evaluation Report

The different tasks in the question answering track each used different evaluation metrics and have different evaluation reports.

3.1 Main task

The basic evaluation measure used in the main task is the reciprocal rank: the score for an individual question is the reciprocal of the rank at which the first correct response was found, or zero if no correct response was found in the first five responses. The score for a run as a whole is the mean of the reciprocal rank over the test set of questions. The judging for the question answering track distinguished between correct answers that were supported by the document returned and correct answers that were not supported. In strict evaluation, unsupported responses were counted as incorrect; in lenient evaluation unsupported answers were counted as correct.

The evaluation report for the main task consists of a table giving detailed evaluation scores for the run and a graph that compares the run to a hypothetical median run. An example of the table is shown in Table 4

Table 4: Sample QA Main Task Table.

Summary Statistics	
Run ID	insight
Num questions	492
Mean reciprocal rank (strict)	0.676
Mean reciprocal rank (lenient)	0.686
Num answers not found (strict)	152 (30.9%)
Num answers not found (lenient)	147 (29.9%)
Number of times NIL returned	120
Number of times NIL correctly returned	38
Percentage of answers system confident about	75%
Percentage of confident answers that were correct	77%

The scores given include:

- The mean reciprocal rank for both strict and lenient evaluation.
- The number and percentage of questions for which the correct response was not returned in the top five responses for both strict and lenient evaluation.
- The number of questions for which 'NIL' was returned as a a response. (NIL indicates the system's belief that no correct response is contained in the document collection.)
- The number of questions for which NIL was returned as a response and it was the correct answer.
- The percentage of questions for which the system was confident it had correctly determined the answer.

- The percentage of questions the system was confident about that were actually correct. (For this computation, the system was judged on its selection of one final answer, not on the list of five responses.)

A sample median graph is shown in Figure 3. The graph is a histogram of the number of questions for which the correct response was returned at a given rank. Plotted is both the run's results and the results for a hypothetical run that retrieved the correct response at the median rank for each question. The median is computed over the entire set of runs submitted to the main task.

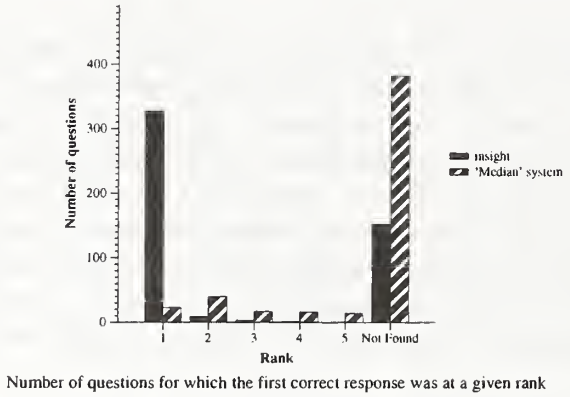


Figure 3: Sample QA Main Task Median Graph.

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.

3.3 Context task

The context task was a pilot study to investigate how well systems can track discourse objects through a series of questions. Since answering later questions in a series requires correctly answering earlier questions in the series, a mean reciprocal rank score over all questions confounds important variables. Since there were only seven runs submitted to the task, median results are also uninformative. The evaluation report for the context task therefore consists simply of the rank at which the first correct response was returned for each of the 42 questions in the test set. Questions are numbered by series and then given a letter for the individual questions within the series. Thus question 3b is the second question of the third series.

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 2001 filtering track were a linear utility function (scaled when averaged) 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

$$\text{T10F} = \begin{cases} 0 & \text{if } R^+ = N+ = 0 \\ \dfrac{1.25R^+}{.25R^- + N^+ + 1.25R^+} & \text{otherwise} \end{cases}$$

and the utility function as

$$\text{T10U} = 2R^+ - N^+.$$

To compute the average utility over a set of topics, the T10U score for the individual topics was scaled between a maximum score of twice the number of relevant documents and a minimum score of −100.

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.

A sample characteristics table is given in Table 5. The characteristics of the run include whether the run was an adaptive or batch run, whether external resources were used in the run, and the measure the run was optimized for (F-beta, T10U, or neither). The scores reported are the recall of the retrieved sets averaged over all topics, the precision of the retrieved sets averaged over all topics, the mean utility, the mean F-beta score, and the number of topics for which no documents were retrieved.

Table 5: Sample Filtering Table.

Summary Statistics	
Run ID	CMUCATsr10
Subtask	adaptive
TREC data used in training?	yes
Reuters data used in training?	no
Other data used in training?	no
Optimized for	T10U
Number of Topics	84
Total retrieved	342552
Relevant retrieved	245386
Macro average recall	0.248
Macro average precision	0.603
Mean T10SU	0.228
Mean F-Beta	0.415
Zero returns	10

A sample median graph is shown in Figure 4. The graph shows the difference between the run's

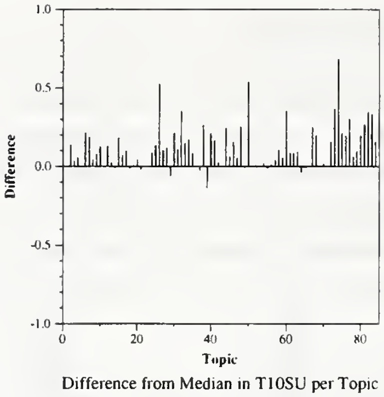


Figure 4: A Sample Filtering Median Graph.

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. A sample time graph is shown in Figure 5.

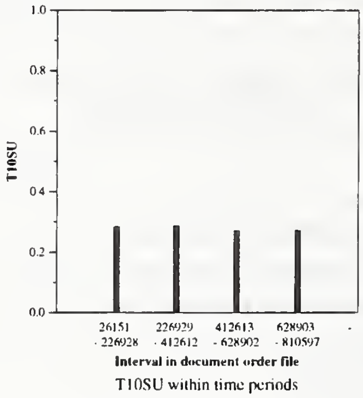


Figure 5: A Sample Filtering Median Graph.

5 Homepage Finding Evaluation Report

The result of a homepage finding task is a ranked list of documents, but the homepage finding task is a known-item search and thus us not evaluated using `trec_eval`. Instead, the runs were evaluated using the rank at which the first correct homepage 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 6. The table contains a description of the run that specifies whether document structure was exploited in the run (docstruct-used or docstruct-notused), whether URL text was exploited in the run (urltext-used or urltext-notused), and whether link structure was exploited in the run (links-used or links-notused). The evaluation scores reported include:

Table 6: Sample Homepage Finding Task Table.

Summary Statistics	
Run ID	tnout10epCAU
Run Description	docstruct-notused, urltext-used, links-used
Num topics	145
Mean reciprocal rank	0.774
Num found at rank 1	102 (70.3%)
Num found in top 10	128 (88.3%)
Num not found in top 100	7 (4.8%)

- The mean reciprocal rank for the run (see the question answering track description for a definition of mean reciprocal rank).
- The number and percentage of topics for which a correct homepage was retrieved in the first rank.
- The number and percentage of topics for which a correct homepage was retrieved in the top ten ranks (includes those topics for which the homepage was returned at rank one).
- The number and percentage of topics for which no correct homepage was returned in the top 100 ranks.

A sample median graph is shown in Figure 6. The graph plots the cumulative percentage of topics for which a correct homepage 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 homepage at the median rank for each topic.

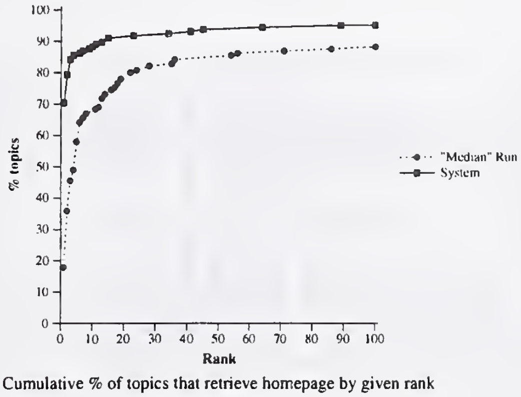


Figure 6: Sample Median Graph for the Homepage Finding Task.

Summary Statistics	
Run ID:	BBN10MON
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	25

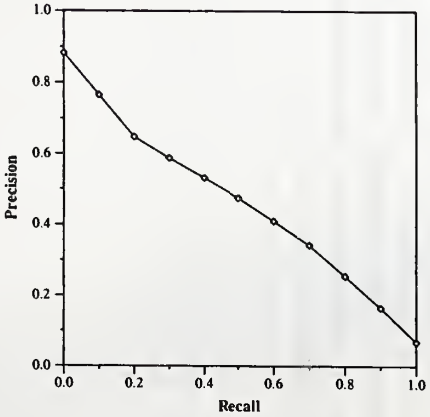
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	3320

Recall Level Averages	
Recall	Precision
0.00	0.8830
0.10	0.7657
0.20	0.6486
0.30	0.5876
0.40	0.5318
0.50	0.4746
0.60	0.4102
0.70	0.3426
0.80	0.2555
0.90	0.1654
1.00	0.0680

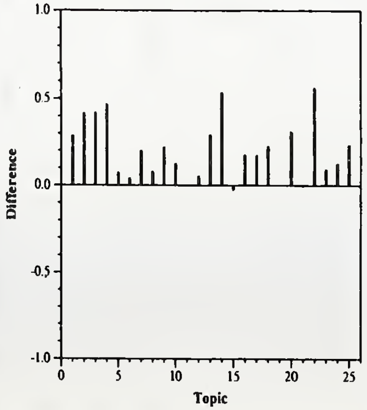
Mean average precision	
non-interpolated	0.4537

Document Level Averages	
	Precision
At 5 docs	0.7120
At 10 docs	0.7160
At 15 docs	0.6933
At 20 docs	0.7020
At 30 docs	0.6533
At 100 docs	0.4860
At 200 docs	0.3732
At 500 docs	0.2230
At 1000 docs	0.1328

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.4528



Recall-Precision Curve



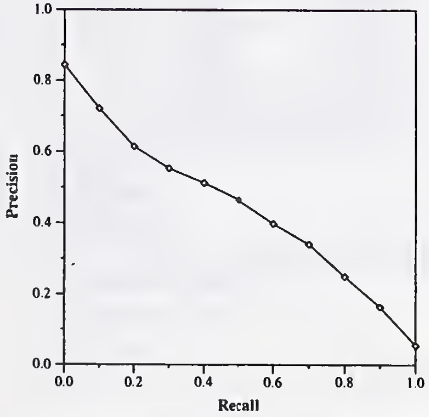
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	BBN10XLA
Run Description	English topics, automatic, title+desc
Number of Topics:	25

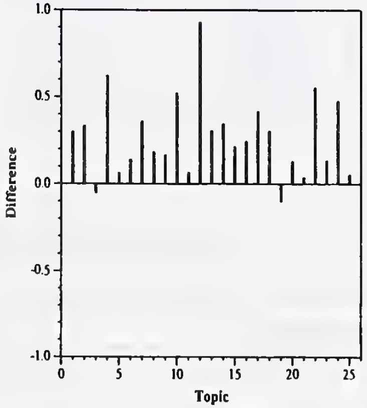
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	3137

Recall Level Averages	
Recall	Precision
0.00	0.8441
0.10	0.7211
0.20	0.6144
0.30	0.5533
0.40	0.5124
0.50	0.4653
0.60	0.3984
0.70	0.3412
0.80	0.2508
0.90	0.1646
1.00	0.0541
Mean average precision	
non-interpolated	0.4382

Document Level Averages	
	Precision
At 5 docs	0.7040
At 10 docs	0.6720
At 15 docs	0.6560
At 20 docs	0.6480
At 30 docs	0.6187
At 100 docs	0.4860
At 200 docs	0.3608
At 500 docs	0.2121
At 1000 docs	0.1255
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.4446



Recall-Precision Curve



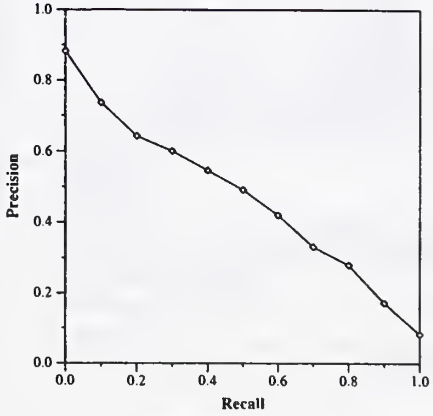
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	BBN10XLB
Run Description	English topics, automatic, title+desc
Number of Topics:	25

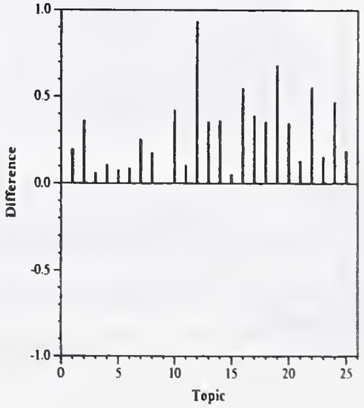
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	3160

Recall Level Averages	
Recall	Precision
0.00	0.8825
0.10	0.7366
0.20	0.6429
0.30	0.6003
0.40	0.5465
0.50	0.4916
0.60	0.4201
0.70	0.3314
0.80	0.2805
0.90	0.1725
1.00	0.0823
Mean average precision	
non-interpolated	0.4639

Document Level Averages	
	Precision
At 5 docs	0.7440
At 10 docs	0.7120
At 15 docs	0.6720
At 20 docs	0.6540
At 30 docs	0.6200
At 100 docs	0.4752
At 200 docs	0.3626
At 500 docs	0.2146
At 1000 docs	0.1264
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.4646



Recall-Precision Curve



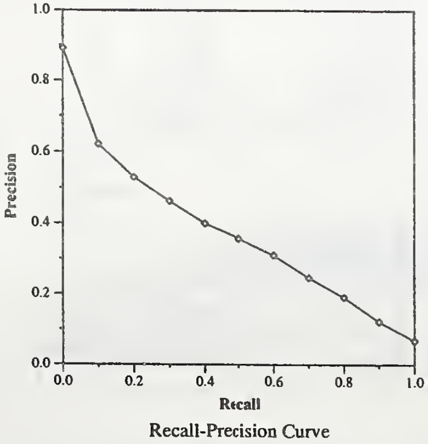
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	BBN10XLC
Run Description	English topics, automatic, title+desc
Number of Topics:	25

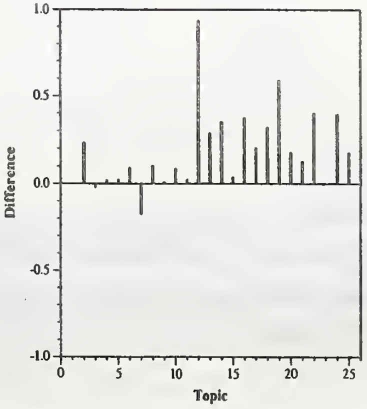
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2751

Recall Level Averages	
Recall	Precision
0.00	0.8911
0.10	0.6213
0.20	0.5292
0.30	0.4622
0.40	0.3993
0.50	0.3562
0.60	0.3098
0.70	0.2465
0.80	0.1908
0.90	0.1223
1.00	0.0684
Mean average precision	
non-interpolated	0.3604

Document Level Averages	
	Precision
At 5 docs	0.6480
At 10 docs	0.5840
At 15 docs	0.5680
At 20 docs	0.5600
At 30 docs	0.5347
At 100 docs	0.3940
At 200 docs	0.2920
At 500 docs	0.1786
At 1000 docs	0.1100
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3820



Recall-Precision Curve



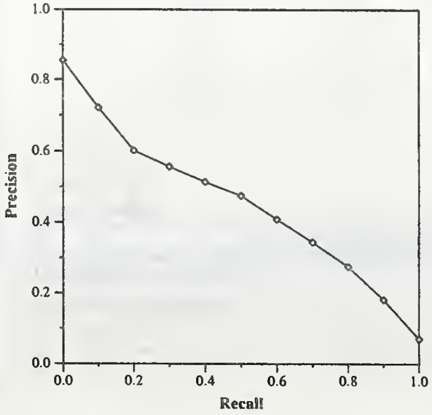
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	BBN10XLD
Run Description	English topics, automatic, title+desc+narr
Number of Topics:	25

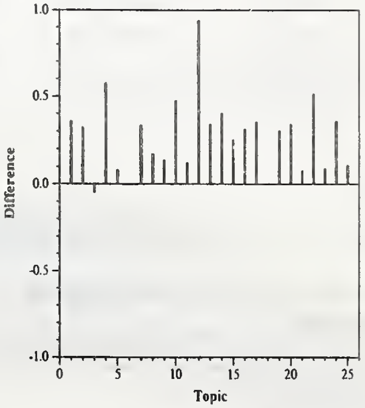
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	3117

Recall Level Averages	
Recall	Precision
0.00	0.8541
0.10	0.7213
0.20	0.6011
0.30	0.5568
0.40	0.5143
0.50	0.4754
0.60	0.4088
0.70	0.3448
0.80	0.2758
0.90	0.1815
1.00	0.0711
Mean average precision	
non-interpolated	0.4453

Document Level Averages	
	Precision
At 5 docs	0.7120
At 10 docs	0.6800
At 15 docs	0.6587
At 20 docs	0.6480
At 30 docs	0.6027
At 100 docs	0.4660
At 200 docs	0.3542
At 500 docs	0.2084
At 1000 docs	0.1247
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.4593



Recall-Precision Curve



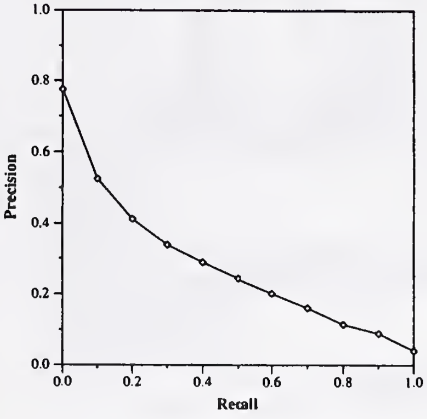
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	humAR01t
Run Description	Arabic topics, automatic, title
Number of Topics:	25

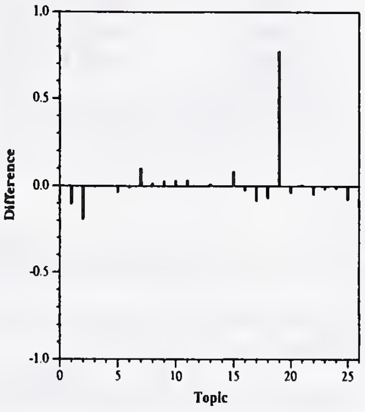
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	1869

Recall Level Averages	
Recall	Precision
0.00	0.7755
0.10	0.5253
0.20	0.4131
0.30	0.3390
0.40	0.2890
0.50	0.2436
0.60	0.2016
0.70	0.1612
0.80	0.1149
0.90	0.0897
1.00	0.0412
Mean average precision	
non-interpolated	0.2663

Document Level Averages	
	Precision
At 5 docs	0.5360
At 10 docs	0.4880
At 15 docs	0.4800
At 20 docs	0.4500
At 30 docs	0.4133
At 100 docs	0.2908
At 200 docs	0.2084
At 500 docs	0.1207
At 1000 docs	0.0748
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3263



Recall-Precision Curve



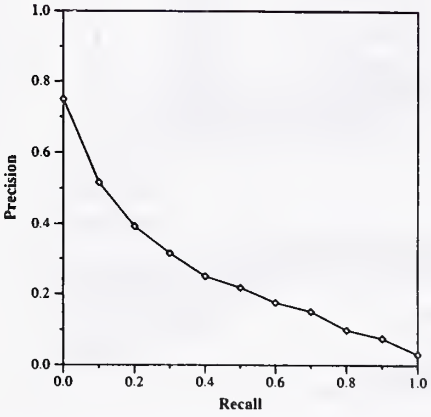
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	humAR01td
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	25

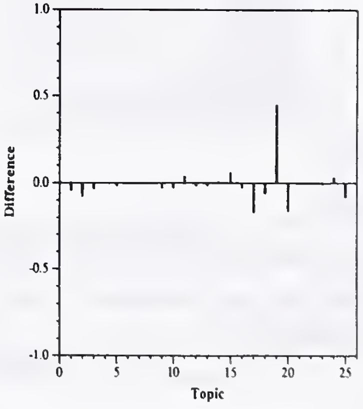
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	1846

Recall Level Averages	
Recall	Precision
0.00	0.7494
0.10	0.5157
0.20	0.3928
0.30	0.3149
0.40	0.2502
0.50	0.2183
0.60	0.1761
0.70	0.1511
0.80	0.0996
0.90	0.0763
1.00	0.0311
Mean average precision	
non-interpolated	0.2441

Document Level Averages	
	Precision
At 5 docs	0.5040
At 10 docs	0.4920
At 15 docs	0.4773
At 20 docs	0.4620
At 30 docs	0.4307
At 100 docs	0.2924
At 200 docs	0.2004
At 500 docs	0.1191
At 1000 docs	0.0738
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3087



Recall-Precision Curve



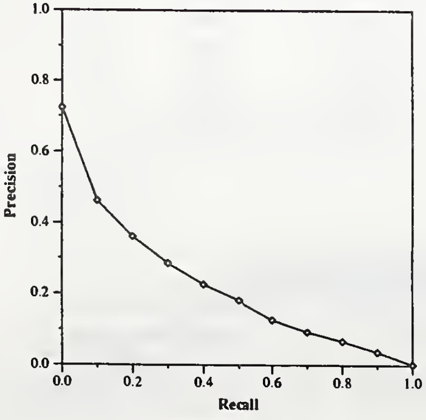
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	humAR01tdm
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	25

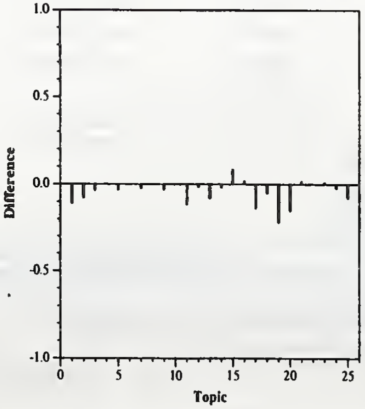
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	1761

Recall Level Averages	
Recall	Precision
0.00	0.7238
0.10	0.4627
0.20	0.3612
0.30	0.2848
0.40	0.2260
0.50	0.1818
0.60	0.1273
0.70	0.0932
0.80	0.0670
0.90	0.0368
1.00	0.0016
Mean average precision	
non-interpolated	0.2087

Document Level Averages	
	Precision
At 5 docs	0.4800
At 10 docs	0.4840
At 15 docs	0.4827
At 20 docs	0.4620
At 30 docs	0.4120
At 100 docs	0.2800
At 200 docs	0.1970
At 500 docs	0.1123
At 1000 docs	0.0704
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2709



Recall-Precision Curve



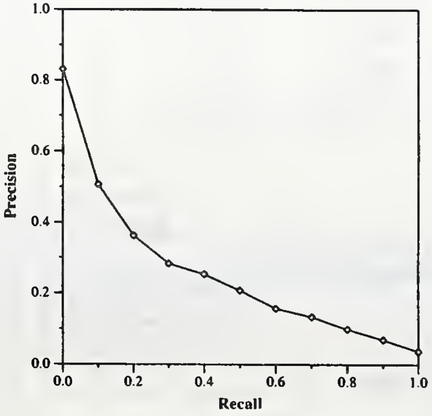
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	humAR01tdn
Run Description	Arabic topics, automatic, title+desc+narr
Number of Topics:	25

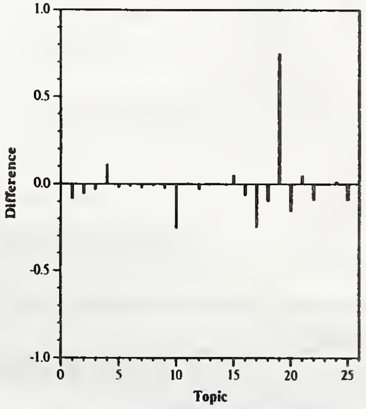
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	1765

Recall Level Averages	
Recall	Precision
0.00	0.8312
0.10	0.5064
0.20	0.3625
0.30	0.2823
0.40	0.2527
0.50	0.2073
0.60	0.1575
0.70	0.1340
0.80	0.0990
0.90	0.0695
1.00	0.0367
Mean average precision	
non-interpolated	0.2395

Document Level Averages	
	Precision
At 5 docs	0.6240
At 10 docs	0.5160
At 15 docs	0.4933
At 20 docs	0.4580
At 30 docs	0.4347
At 100 docs	0.2716
At 200 docs	0.1934
At 500 docs	0.1126
At 1000 docs	0.0706
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2914



Recall-Precision Curve



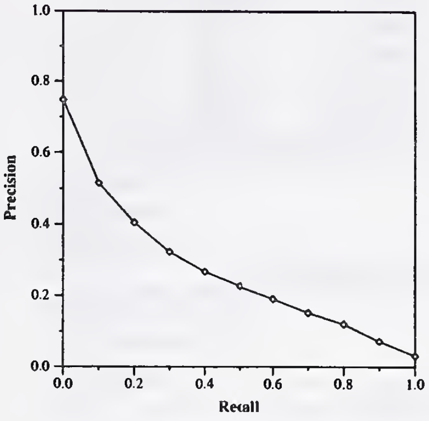
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	humAR01tdx
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	25

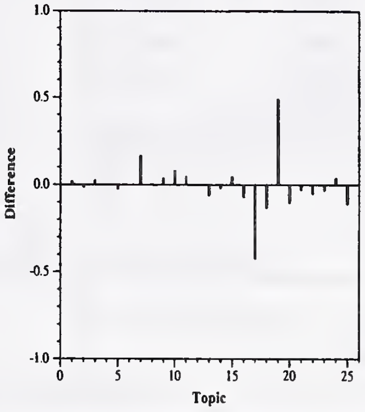
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	1721

Recall Level Averages	
Recall	Precision
0.00	0.7486
0.10	0.5159
0.20	0.4050
0.30	0.3235
0.40	0.2675
0.50	0.2272
0.60	0.1912
0.70	0.1524
0.80	0.1209
0.90	0.0724
1.00	0.0314
Mean average precision	
non-interpolated	0.2465

Document Level Averages	
	Precision
At 5 docs	0.5120
At 10 docs	0.4800
At 15 docs	0.4613
At 20 docs	0.4380
At 30 docs	0.3907
At 100 docs	0.2844
At 200 docs	0.2004
At 500 docs	0.1088
At 1000 docs	0.0688
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3126



Recall-Precsion Curve



Difference from Median in Average Precision per Topic

Cross-language track results — Illinois Institute of Technology, IIT Information Retrieval Lab

Summary Statistics	
Run ID:	iit01md
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	25

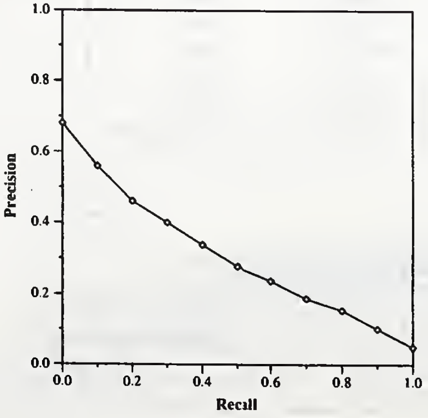
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2405

Recall Level Averages	
Recall	Precision
0.00	0.6805
0.10	0.5595
0.20	0.4606
0.30	0.3998
0.40	0.3371
0.50	0.2775
0.60	0.2378
0.70	0.1881
0.80	0.1547
0.90	0.1011
1.00	0.0490

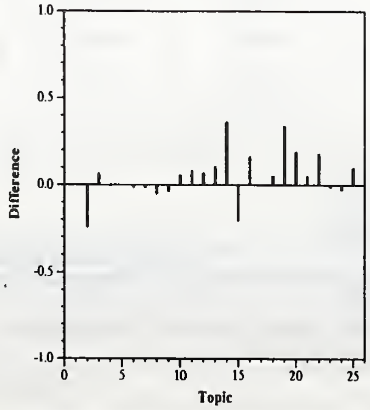
Mean average precision	
non-interpolated	0.2987

Document Level Averages	
	Precision
At 5 docs	0.5040
At 10 docs	0.4960
At 15 docs	0.4853
At 20 docs	0.4660
At 30 docs	0.4333
At 100 docs	0.2996
At 200 docs	0.2306
At 500 docs	0.1524
At 1000 docs	0.0962

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3362



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	iit01ml
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	25

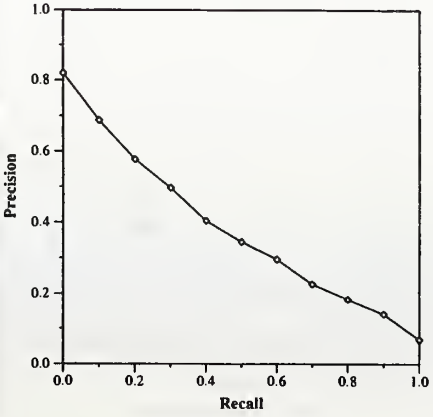
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2720

Recall Level Averages	
Recall	Precision
0.00	0.8201
0.10	0.6881
0.20	0.5773
0.30	0.4975
0.40	0.4046
0.50	0.3444
0.60	0.2965
0.70	0.2274
0.80	0.1842
0.90	0.1416
1.00	0.0686

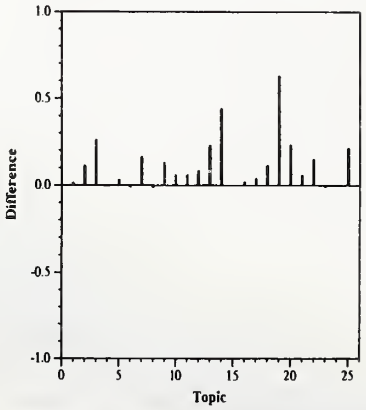
Mean average precision	
non-interpolated	0.3709

Document Level Averages	
	Precision
At 5 docs	0.6640
At 10 docs	0.6320
At 15 docs	0.5653
At 20 docs	0.5360
At 30 docs	0.5173
At 100 docs	0.3928
At 200 docs	0.2966
At 500 docs	0.1778
At 1000 docs	0.1088

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.4112



Recall-Precision Curve



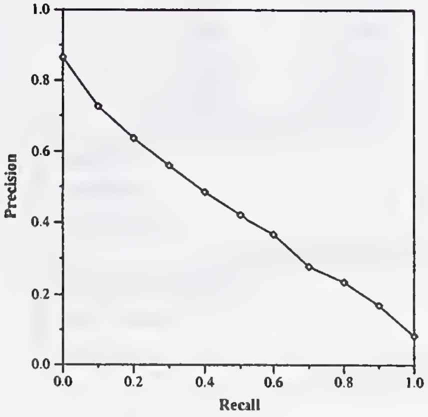
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	iit01mlr
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	25

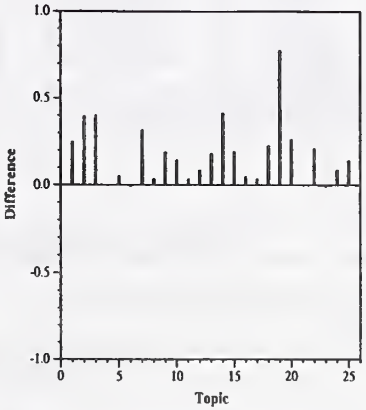
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2951

Recall Level Averages	
Recall	Precision
0.00	0.8653
0.10	0.7269
0.20	0.6375
0.30	0.5602
0.40	0.4855
0.50	0.4219
0.60	0.3674
0.70	0.2779
0.80	0.2351
0.90	0.1708
1.00	0.0826
Mean average precision	
non-interpolated	0.4288

Document Level Averages	
	Precision
At 5 docs	0.7120
At 10 docs	0.6680
At 15 docs	0.6533
At 20 docs	0.6360
At 30 docs	0.5840
At 100 docs	0.4380
At 200 docs	0.3296
At 500 docs	0.1977
At 1000 docs	0.1180
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.4615



Recall-Precision Curve



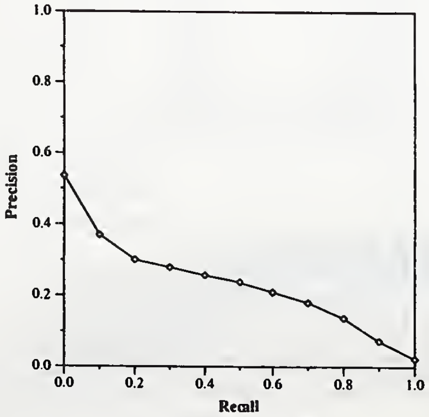
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	iit01xdi
Run Description	English topics, automatic, title+desc
Number of Topics:	25

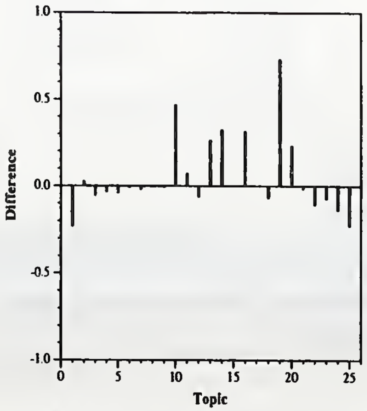
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2026

Recall Level Averages	
Recall	Precision
0.00	0.5361
0.10	0.3704
0.20	0.3009
0.30	0.2794
0.40	0.2568
0.50	0.2379
0.60	0.2101
0.70	0.1813
0.80	0.1380
0.90	0.0737
1.00	0.0226
Mean average precision	
non-interpolated	0.2237

Document Level Averages	
	Precision
At 5 docs	0.3840
At 10 docs	0.3160
At 15 docs	0.3013
At 20 docs	0.2960
At 30 docs	0.2933
At 100 docs	0.2272
At 200 docs	0.1754
At 500 docs	0.1174
At 1000 docs	0.0810
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2425



Recall-Precision Curve



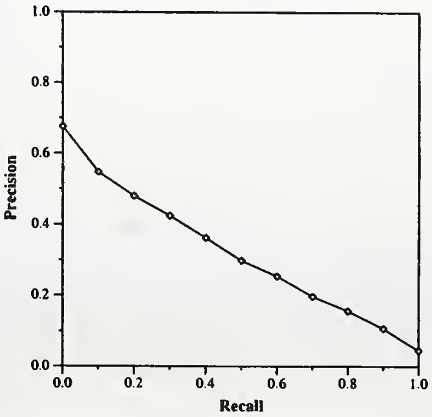
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	iit01xma
Run Description	English topics, automatic, title+desc
Number of Topics:	25

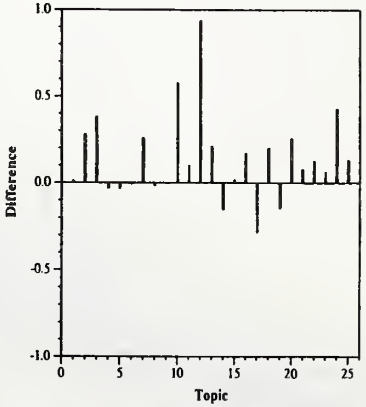
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2511

Recall Level Averages	
Recall	Precision
0.00	0.6759
0.10	0.5467
0.20	0.4793
0.30	0.4238
0.40	0.3624
0.50	0.2982
0.60	0.2538
0.70	0.1967
0.80	0.1568
0.90	0.1078
1.00	0.0457
Mean average precision	
non-interpolated	0.3119

Document Level Averages	
	Precision
At 5 docs	0.5360
At 10 docs	0.5160
At 15 docs	0.5013
At 20 docs	0.4760
At 30 docs	0.4373
At 100 docs	0.3396
At 200 docs	0.2710
At 500 docs	0.1669
At 1000 docs	0.1004
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3376



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	apl10ca1
Run Description	Arabic topics, automatic, title+desc+narr
Number of Topics:	25

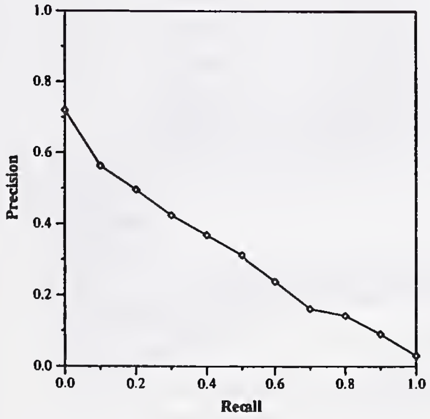
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2669

Recall Level Averages	
Recall	Precision
0.00	0.7199
0.10	0.5628
0.20	0.4956
0.30	0.4241
0.40	0.3690
0.50	0.3125
0.60	0.2386
0.70	0.1620
0.80	0.1423
0.90	0.0911
1.00	0.0307

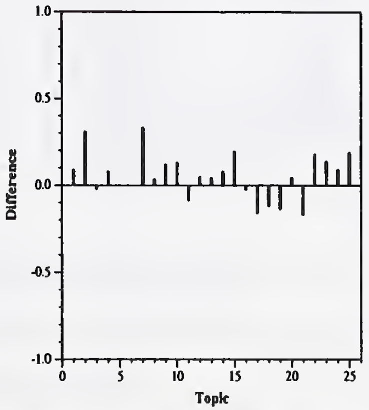
Mean average precision	
non-interpolated	0.3064

Document Level Averages	
	Precision
At 5 docs	0.5600
At 10 docs	0.5520
At 15 docs	0.5493
At 20 docs	0.5480
At 30 docs	0.5080
At 100 docs	0.3608
At 200 docs	0.2638
At 500 docs	0.1694
At 1000 docs	0.1068

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3387



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	apl10ce1
Run Description	English topics, automatic, title+desc+narr
Number of Topics:	25

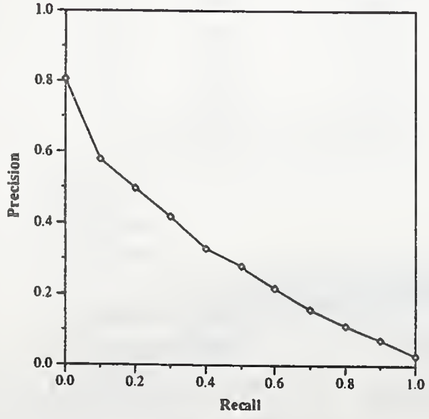
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2819

Recall Level Averages	
Recall	Precision
0.00	0.8054
0.10	0.5780
0.20	0.4969
0.30	0.4175
0.40	0.3279
0.50	0.2791
0.60	0.2172
0.70	0.1573
0.80	0.1115
0.90	0.0721
1.00	0.0276

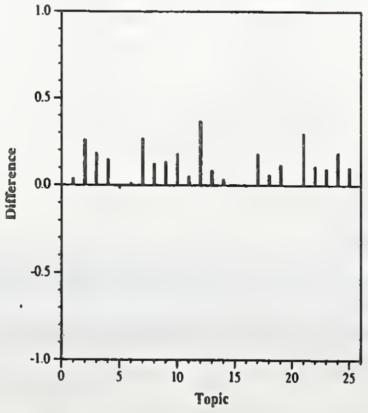
Mean average precision	
non-interpolated	0.2891

Document Level Averages	
	Precision
At 5 docs	0.5600
At 10 docs	0.5520
At 15 docs	0.5280
At 20 docs	0.4860
At 30 docs	0.4533
At 100 docs	0.3800
At 200 docs	0.2944
At 500 docs	0.1831
At 1000 docs	0.1128

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3316



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	apl10ce2
Run Description	English topics, automatic, title+desc
Number of Topics:	25

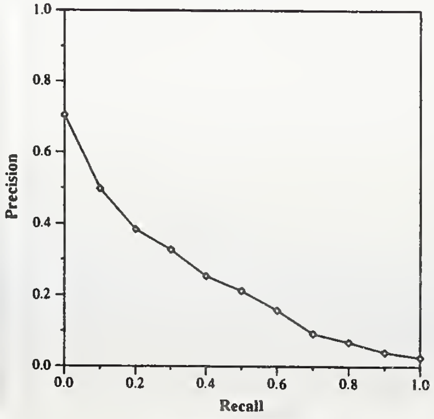
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2593

Recall Level Averages	
Recall	Precision
0.00	0.7046
0.10	0.4969
0.20	0.3839
0.30	0.3264
0.40	0.2531
0.50	0.2121
0.60	0.1576
0.70	0.0924
0.80	0.0680
0.90	0.0403
1.00	0.0260

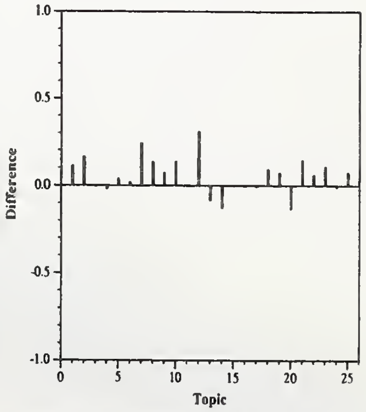
Mean average precision	
non-interpolated	0.2250

Document Level Averages	
	Precision
At 5 docs	0.5360
At 10 docs	0.5240
At 15 docs	0.4800
At 20 docs	0.4380
At 30 docs	0.4147
At 100 docs	0.3184
At 200 docs	0.2380
At 500 docs	0.1584
At 1000 docs	0.1037

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2602



Recall-Precision Curve



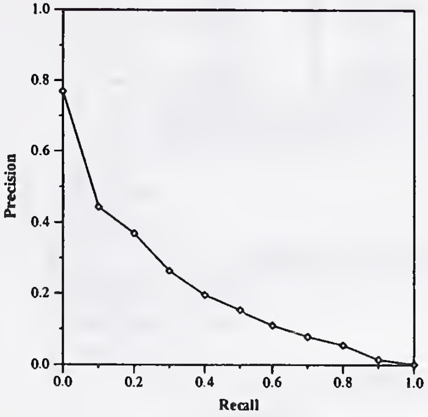
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	apl10ce3
Run Description	English topics, automatic, title+desc
Number of Topics:	25

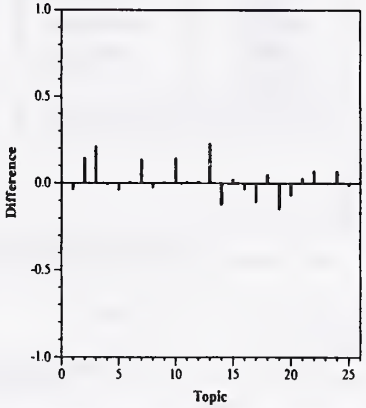
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2350

Recall Level Averages	
Recall	Precision
0.00	0.7694
0.10	0.4434
0.20	0.3696
0.30	0.2637
0.40	0.1958
0.50	0.1532
0.60	0.1104
0.70	0.0789
0.80	0.0549
0.90	0.0148
1.00	0.0016
Mean average precision	
non-interpolated	0.1914

Document Level Averages	
	Precision
At 5 docs	0.4560
At 10 docs	0.4480
At 15 docs	0.4267
At 20 docs	0.4060
At 30 docs	0.3733
At 100 docs	0.2900
At 200 docs	0.2186
At 500 docs	0.1400
At 1000 docs	0.0940
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2224



Recall-Precision Curve



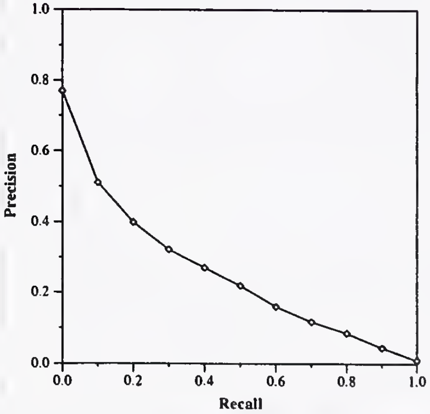
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	apl10cf1
Run Description	French topics, automatic, title+desc+narr
Number of Topics:	25

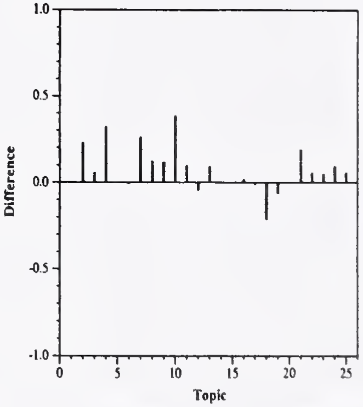
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2574

Recall Level Averages	
Recall	Precision
0.00	0.7703
0.10	0.5104
0.20	0.3985
0.30	0.3215
0.40	0.2698
0.50	0.2188
0.60	0.1597
0.70	0.1171
0.80	0.0854
0.90	0.0442
1.00	0.0093
Mean average precision	
non-interpolated	0.2415

Document Level Averages	
	Precision
At 5 docs	0.5680
At 10 docs	0.5240
At 15 docs	0.4880
At 20 docs	0.4600
At 30 docs	0.4360
At 100 docs	0.3200
At 200 docs	0.2548
At 500 docs	0.1602
At 1000 docs	0.1030
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2891



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	NMMLTN
Run Description	Arabic topics, automatic, title
Number of Topics:	25

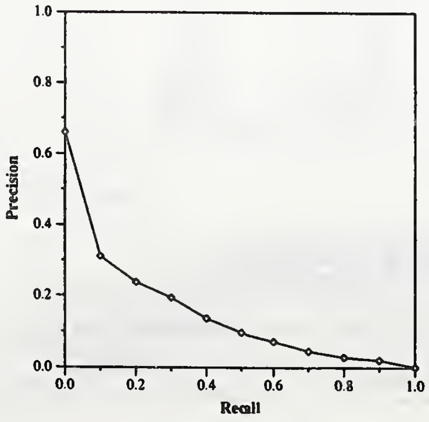
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	1515

Recall Level Averages	
Recall	Precision
0.00	0.6609
0.10	0.3108
0.20	0.2375
0.30	0.1939
0.40	0.1364
0.50	0.0975
0.60	0.0719
0.70	0.0458
0.80	0.0288
0.90	0.0208
1.00	0.0015

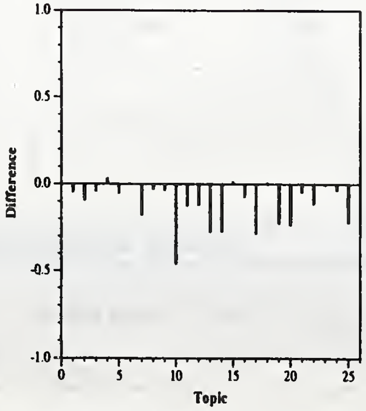
Mean average precision	
non-interpolated	0.1353

Document Level Averages	
	Precision
At 5 docs	0.3840
At 10 docs	0.3760
At 15 docs	0.3387
At 20 docs	0.3220
At 30 docs	0.2853
At 100 docs	0.1840
At 200 docs	0.1418
At 500 docs	0.0894
At 1000 docs	0.0606

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1946



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	NMCLDS
Run Description	English topics, automatic, description
Number of Topics:	25

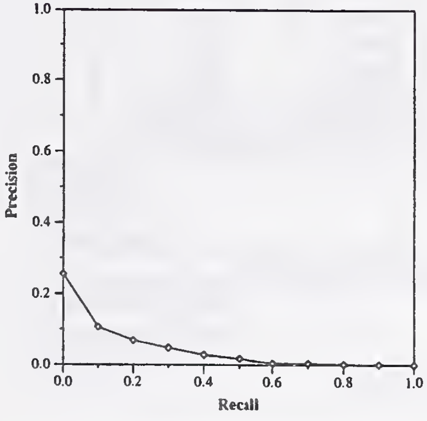
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	845

Recall Level Averages	
Recall	Precision
0.00	0.2560
0.10	0.1064
0.20	0.0688
0.30	0.0479
0.40	0.0285
0.50	0.0176
0.60	0.0043
0.70	0.0043
0.80	0.0018
0.90	0.0012
1.00	0.0003

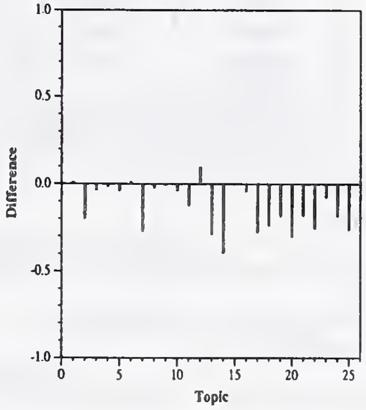
Mean average precision	
non-interpolated	0.0370

Document Level Averages	
	Precision
At 5 docs	0.1120
At 10 docs	0.1360
At 15 docs	0.1253
At 20 docs	0.1360
At 30 docs	0.1253
At 100 docs	0.0936
At 200 docs	0.0728
At 500 docs	0.0478
At 1000 docs	0.0338

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0760



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	NMCLMN
Run Description	English topics, manual
Number of Topics:	25

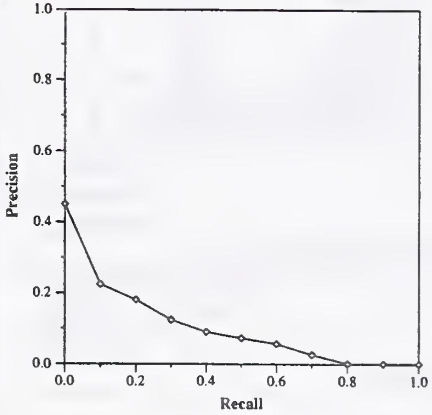
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	995

Recall Level Averages	
Recall	Precision
0.00	0.4502
0.10	0.2250
0.20	0.1807
0.30	0.1241
0.40	0.0903
0.50	0.0729
0.60	0.0572
0.70	0.0270
0.80	0.0016
0.90	0.0012
1.00	0.0010

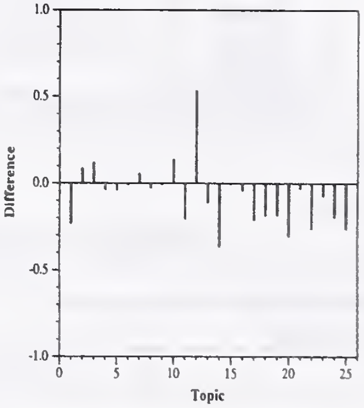
Mean average precision	
non-interpolated	0.0950

Document Level Averages	
	Precision
At 5 docs	0.3040
At 10 docs	0.2680
At 15 docs	0.2347
At 20 docs	0.2260
At 30 docs	0.2120
At 100 docs	0.1420
At 200 docs	0.1026
At 500 docs	0.0602
At 1000 docs	0.0398

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1228



Recall-Precision Curve



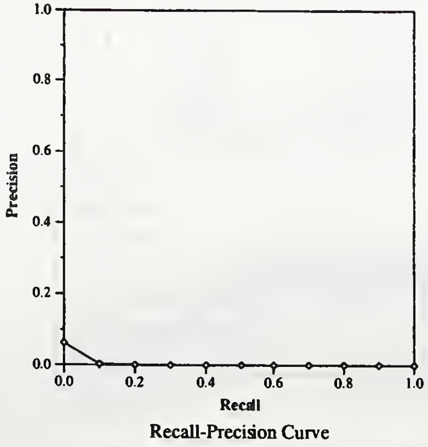
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	NMCLTNv1
Run Description	English topics, automatic, title
Number of Topics:	25

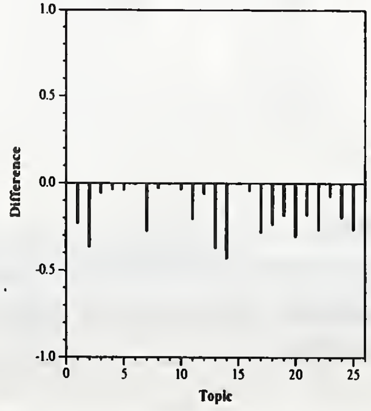
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	161

Recall Level Averages	
Recall	Precision
0.00	0.0624
0.10	0.0027
0.20	0.0007
0.30	0.0005
0.40	0.0005
0.50	0.0005
0.60	0.0005
0.70	0.0005
0.80	0.0005
0.90	0.0005
1.00	0.0005
Mean average precision	
non-interpolated	0.0019

Document Level Averages	
	Precision
At 5 docs	0.0080
At 10 docs	0.0040
At 15 docs	0.0080
At 20 docs	0.0120
At 30 docs	0.0147
At 100 docs	0.0112
At 200 docs	0.0104
At 500 docs	0.0079
At 1000 docs	0.0064
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0106



Recall-Precision Curve



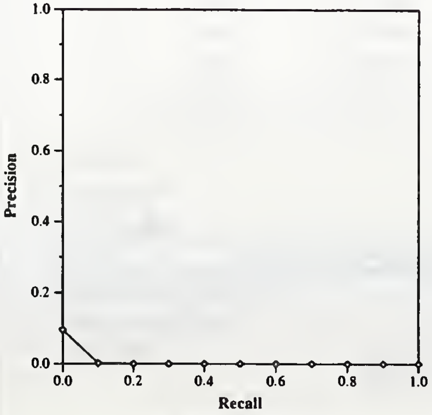
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	NMCLTS
Run Description	English topics, automatic, title
Number of Topics:	25

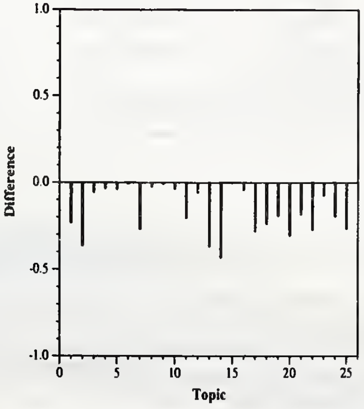
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	116

Recall Level Averages	
Recall	Precision
0.00	0.0957
0.10	0.0016
0.20	0.0000
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	0.0013

Document Level Averages	
	Precision
At 5 docs	0.0160
At 10 docs	0.0120
At 15 docs	0.0187
At 20 docs	0.0180
At 30 docs	0.0147
At 100 docs	0.0092
At 200 docs	0.0070
At 500 docs	0.0052
At 1000 docs	0.0046
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0067



Recall-Precision Curve



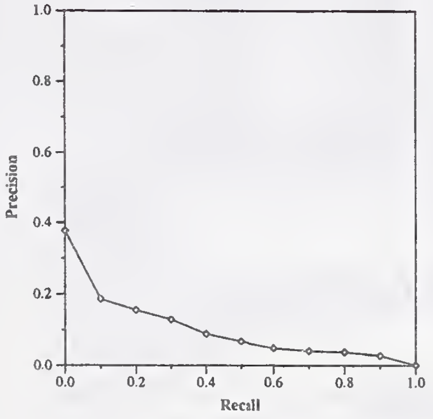
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	pir1XAtd
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	25

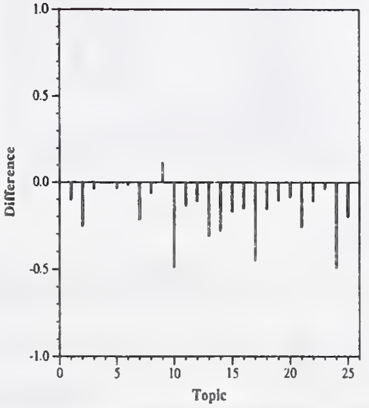
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	974

Recall Level Averages	
Recall	Precision
0.00	0.3765
0.10	0.1865
0.20	0.1547
0.30	0.1287
0.40	0.0890
0.50	0.0680
0.60	0.0490
0.70	0.0412
0.80	0.0382
0.90	0.0280
1.00	0.0005
Mean average precision	
non-interpolated	0.0852

Document Level Averages	
	Precision
At 5 docs	0.2080
At 10 docs	0.1720
At 15 docs	0.1653
At 20 docs	0.1540
At 30 docs	0.1520
At 100 docs	0.1052
At 200 docs	0.0856
At 500 docs	0.0553
At 1000 docs	0.0390
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1405



Recall-Precision Curve



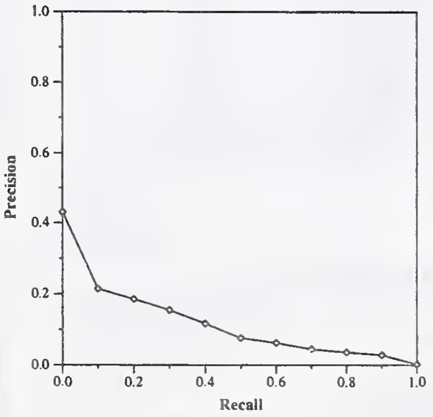
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	pir1XAtdn
Run Description	Arabic topics, automatic, title+desc+narr
Number of Topics:	25

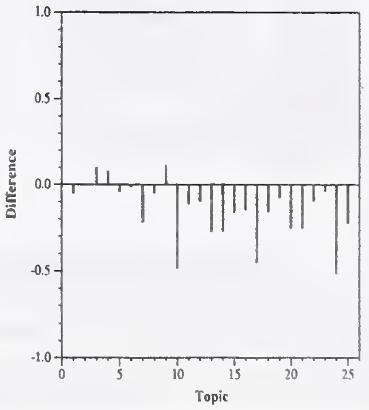
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	1254

Recall Level Averages	
Recall	Precision
0.00	0.4305
0.10	0.2146
0.20	0.1857
0.30	0.1545
0.40	0.1170
0.50	0.0766
0.60	0.0634
0.70	0.0452
0.80	0.0365
0.90	0.0288
1.00	0.0021
Mean average precision	
non-interpolated	0.1036

Document Level Averages	
	Precision
At 5 docs	0.2880
At 10 docs	0.2440
At 15 docs	0.2187
At 20 docs	0.2120
At 30 docs	0.2000
At 100 docs	0.1320
At 200 docs	0.1090
At 500 docs	0.0746
At 1000 docs	0.0502
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1602



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Cross-language track results — Queens College, CUNY

Summary Statistics	
Run ID:	pir1XEtd
Run Description	English topics, automatic, title+desc
Number of Topics:	25

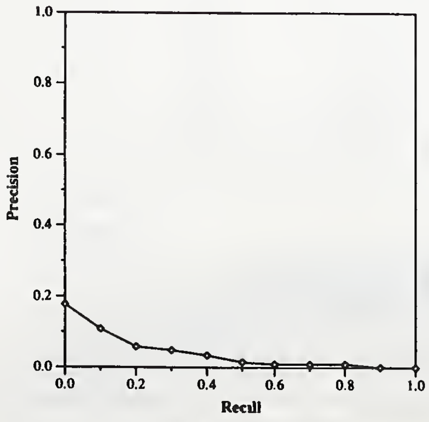
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	802

Recall Level Averages	
Recall	Precision
0.00	0.1777
0.10	0.1081
0.20	0.0585
0.30	0.0484
0.40	0.0342
0.50	0.0152
0.60	0.0094
0.70	0.0094
0.80	0.0094
0.90	0.0007
1.00	0.0007

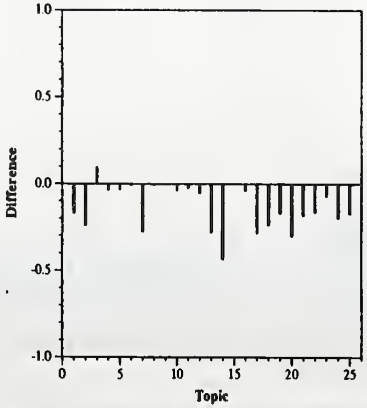
Mean average precision	
non-interpolated	0.0360

Document Level Averages	
	Precision
At 5 docs	0.0960
At 10 docs	0.1040
At 15 docs	0.0933
At 20 docs	0.0920
At 30 docs	0.0867
At 100 docs	0.0812
At 200 docs	0.0718
At 500 docs	0.0501
At 1000 docs	0.0321

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0647



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	pir1XEtdn
Run Description	English topics, automatic, title+desc+narr
Number of Topics:	25

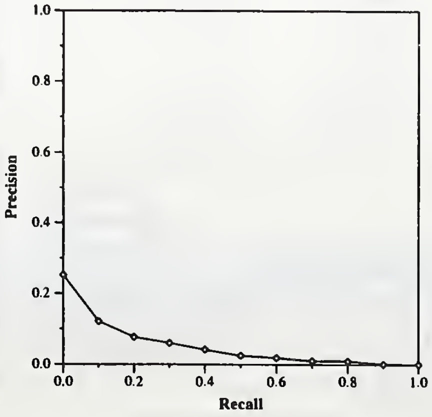
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	899

Recall Level Averages	
Recall	Precision
0.00	0.2523
0.10	0.1215
0.20	0.0775
0.30	0.0606
0.40	0.0420
0.50	0.0251
0.60	0.0193
0.70	0.0102
0.80	0.0102
0.90	0.0015
1.00	0.0005

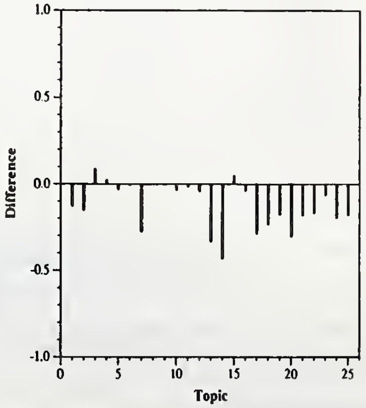
Mean average precision	
non-interpolated	0.0440

Document Level Averages	
	Precision
At 5 docs	0.1360
At 10 docs	0.1280
At 15 docs	0.1200
At 20 docs	0.1220
At 30 docs	0.1200
At 100 docs	0.1028
At 200 docs	0.0816
At 500 docs	0.0561
At 1000 docs	0.0360

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0768



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	BKYAAA1
Run Description	Arabic topics, automatic, title+desc+narr
Number of Topics:	25

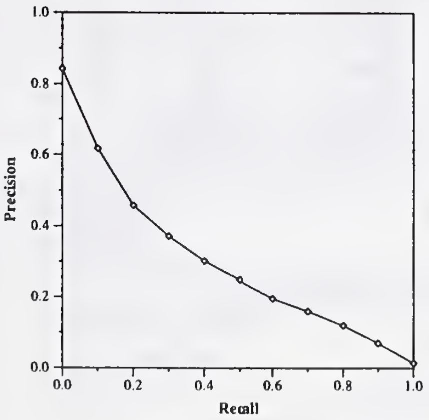
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2393

Recall Level Averages	
Recall	Precision
0.00	0.8432
0.10	0.6174
0.20	0.4582
0.30	0.3716
0.40	0.3021
0.50	0.2487
0.60	0.1959
0.70	0.1604
0.80	0.1200
0.90	0.0701
1.00	0.0141

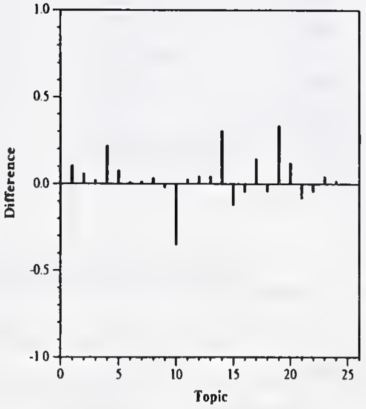
Mean average precision	
non-interpolated	0.2877

Document Level Averages	
	Precision
At 5 docs	0.6320
At 10 docs	0.6080
At 15 docs	0.5573
At 20 docs	0.5440
At 30 docs	0.4973
At 100 docs	0.3576
At 200 docs	0.2624
At 500 docs	0.1562
At 1000 docs	0.0957

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3430



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	BKYEAA1
Run Description	English topics, automatic, title+desc+narr
Number of Topics:	25

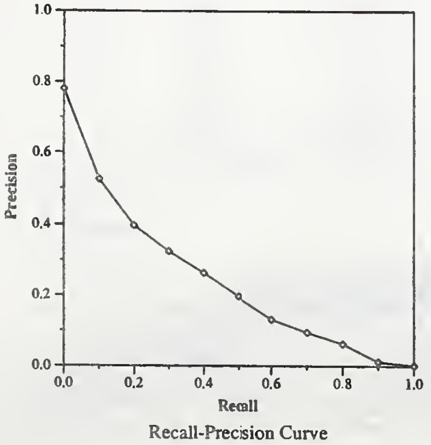
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2579

Recall Level Averages	
Recall	Precision
0.00	0.7803
0.10	0.5250
0.20	0.3970
0.30	0.3241
0.40	0.2627
0.50	0.1967
0.60	0.1309
0.70	0.0945
0.80	0.0620
0.90	0.0121
1.00	0.0014

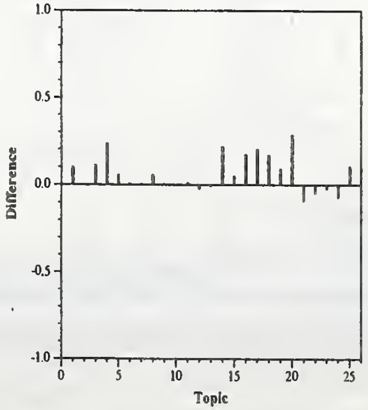
Mean average precision	
non-interpolated	0.2337

Document Level Averages	
	Precision
At 5 docs	0.5360
At 10 docs	0.5120
At 15 docs	0.4907
At 20 docs	0.4740
At 30 docs	0.4573
At 100 docs	0.3624
At 200 docs	0.2710
At 500 docs	0.1646
At 1000 docs	0.1032

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2887



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	BKYEAA2
Run Description	English topics, automatic, title+desc+narr
Number of Topics:	25

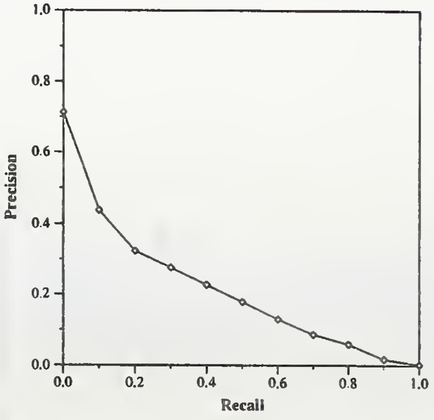
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2485

Recall Level Averages	
Recall	Precision
0.00	0.7133
0.10	0.4374
0.20	0.3229
0.30	0.2752
0.40	0.2265
0.50	0.1780
0.60	0.1290
0.70	0.0861
0.80	0.0588
0.90	0.0170
1.00	0.0015

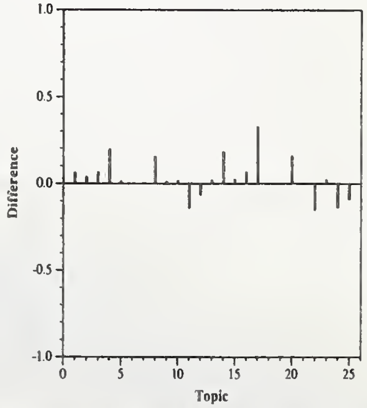
Mean average precision	
non-interpolated	0.2006

Document Level Averages	
	Precision
At 5 docs	0.4720
At 10 docs	0.4440
At 15 docs	0.4347
At 20 docs	0.4220
At 30 docs	0.3987
At 100 docs	0.3104
At 200 docs	0.2494
At 500 docs	0.1598
At 1000 docs	0.0994

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2644



Recall-Precision Curve



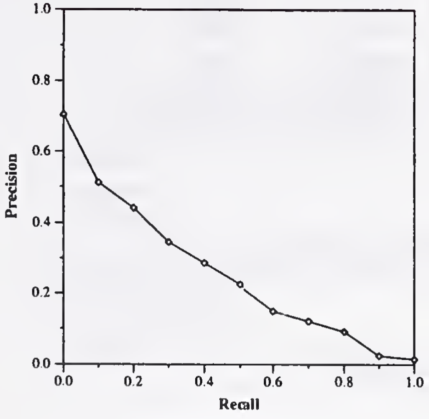
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	BKYEAA3
Run Description	English topics, automatic, title+desc+narr
Number of Topics:	25

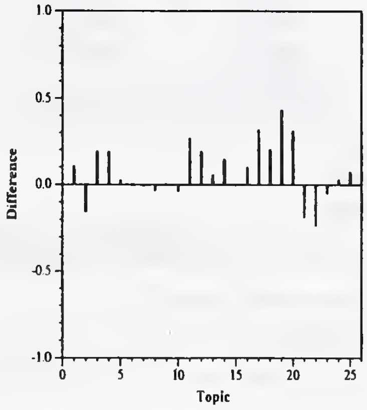
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2490

Recall Level Averages	
Recall	Precision
0.00	0.7052
0.10	0.5119
0.20	0.4418
0.30	0.3463
0.40	0.2870
0.50	0.2257
0.60	0.1490
0.70	0.1206
0.80	0.0915
0.90	0.0240
1.00	0.0141
Mean average precision	
non-interpolated	0.2465

Document Level Averages	
	Precision
At 5 docs	0.5200
At 10 docs	0.4880
At 15 docs	0.4907
At 20 docs	0.4740
At 30 docs	0.4213
At 100 docs	0.3312
At 200 docs	0.2572
At 500 docs	0.1611
At 1000 docs	0.0996
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2870



Recall-Precision Curve



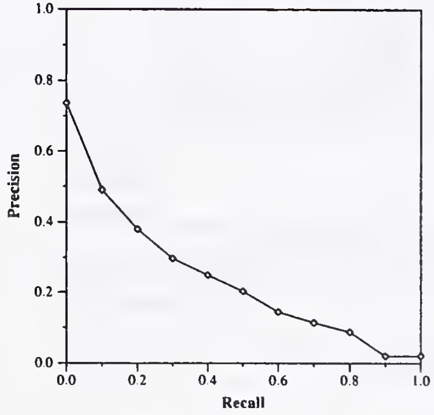
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	BKYEAA4
Run Description	English topics, automatic, title+desc
Number of Topics:	25

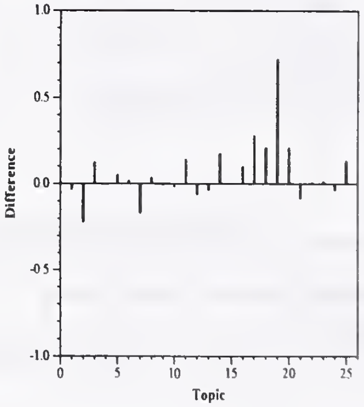
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2300

Recall Level Averages	
Recall	Precision
0.00	0.7372
0.10	0.4901
0.20	0.3807
0.30	0.2967
0.40	0.2493
0.50	0.2026
0.60	0.1437
0.70	0.1134
0.80	0.0874
0.90	0.0200
1.00	0.0200
Mean average precision	
non-interpolated	0.2316

Document Level Averages	
	Precision
At 5 docs	0.5600
At 10 docs	0.4920
At 15 docs	0.4613
At 20 docs	0.4460
At 30 docs	0.4267
At 100 docs	0.3236
At 200 docs	0.2402
At 500 docs	0.1470
At 1000 docs	0.0920
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2783



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UMmanual
Run Description	Arabic topics, manual
Number of Topics:	25

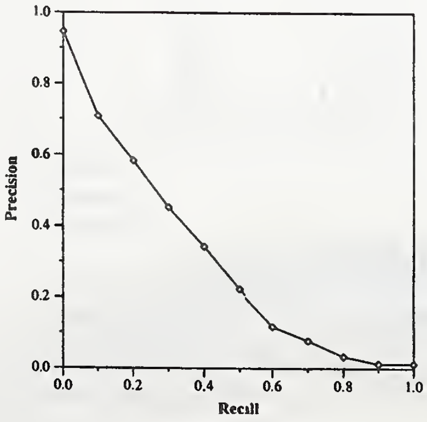
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2019

Recall Level Averages	
Recall	Precision
0.00	0.9460
0.10	0.7082
0.20	0.5829
0.30	0.4523
0.40	0.3425
0.50	0.2227
0.60	0.1166
0.70	0.0770
0.80	0.0333
0.90	0.0126
1.00	0.0126

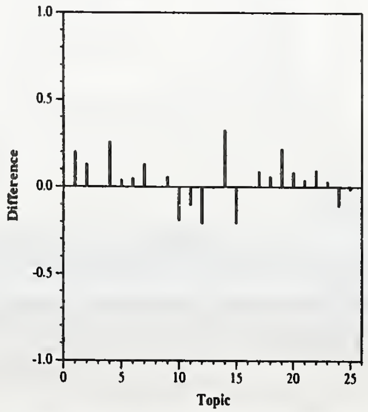
Mean average precision	
non-interpolated	0.2898

Document Level Averages	
	Precision
At 5 docs	0.7760
At 10 docs	0.7240
At 15 docs	0.6747
At 20 docs	0.6300
At 30 docs	0.5813
At 100 docs	0.4004
At 200 docs	0.2836
At 500 docs	0.1458
At 1000 docs	0.0808

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3613



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UMmonoAuto
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	25

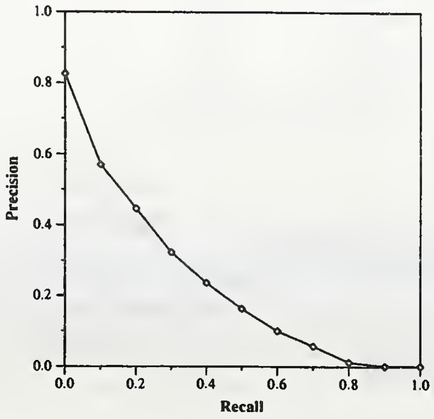
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	1889

Recall Level Averages	
Recall	Precision
0.00	0.8258
0.10	0.5696
0.20	0.4460
0.30	0.3228
0.40	0.2365
0.50	0.1629
0.60	0.1001
0.70	0.0573
0.80	0.0123
0.90	0.0014
1.00	0.0014

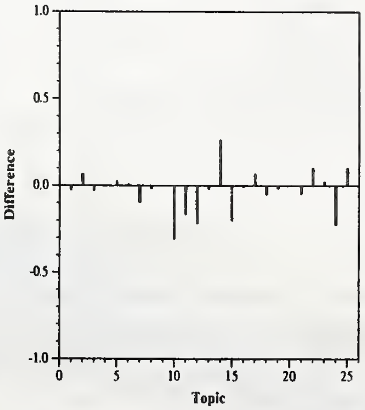
Mean average precision	
non-interpolated	0.2209

Document Level Averages	
	Precision
At 5 docs	0.6160
At 10 docs	0.5800
At 15 docs	0.5387
At 20 docs	0.5040
At 30 docs	0.4627
At 100 docs	0.3216
At 200 docs	0.2358
At 500 docs	0.1302
At 1000 docs	0.0756

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2930



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Cross-language track results — University of Maryland (Oard)

Summary Statistics	
Run ID:	UMclirAutoFL
Run Description	English topics, automatic, title+desc
Number of Topics:	25

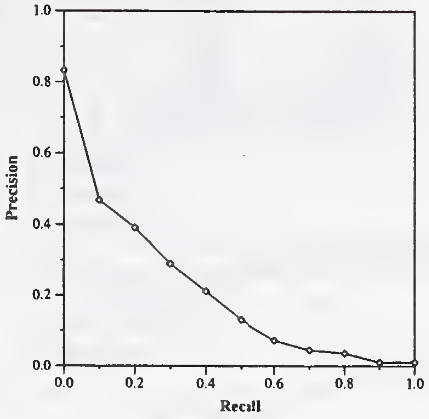
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	1906

Recall Level Averages	
Recall	Precision
0.00	0.8326
0.10	0.4685
0.20	0.3917
0.30	0.2894
0.40	0.2116
0.50	0.1309
0.60	0.0720
0.70	0.0446
0.80	0.0363
0.90	0.0104
1.00	0.0104

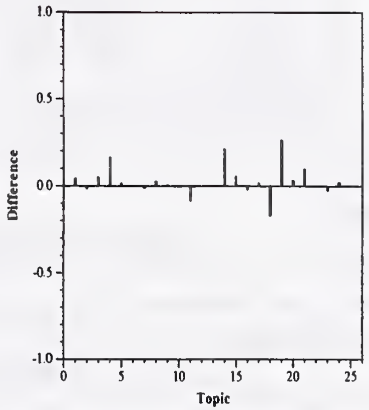
Mean average precision	
non-interpolated	0.1974

Document Level Averages	
	Precision
At 5 docs	0.5920
At 10 docs	0.5280
At 15 docs	0.4933
At 20 docs	0.4660
At 30 docs	0.4360
At 100 docs	0.3132
At 200 docs	0.2296
At 500 docs	0.1292
At 1000 docs	0.0762

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2628



Recall-Preciiion Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UMclirAutoTJ
Run Description	English topics, automatic, title+desc
Number of Topics:	25

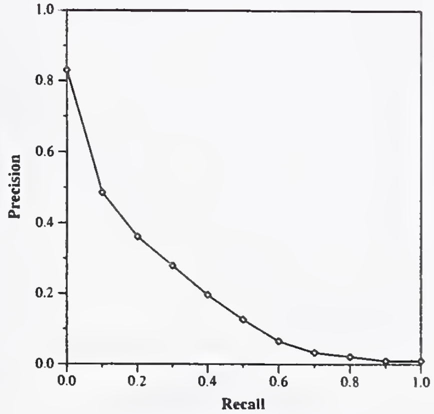
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	1892

Recall Level Averages	
Recall	Precision
0.00	0.8304
0.10	0.4860
0.20	0.3611
0.30	0.2792
0.40	0.1963
0.50	0.1262
0.60	0.0649
0.70	0.0328
0.80	0.0213
0.90	0.0096
1.00	0.0096

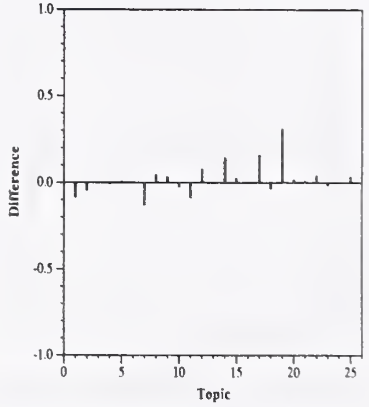
Mean average precision	
non-interpolated	0.1879

Document Level Averages	
	Precision
At 5 docs	0.5680
At 10 docs	0.5000
At 15 docs	0.4587
At 20 docs	0.4440
At 30 docs	0.4067
At 100 docs	0.2900
At 200 docs	0.2248
At 500 docs	0.1298
At 1000 docs	0.0757

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2603



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Cross-language track results — University of Maryland (Oard)

Summary Statistics	
Run ID:	UMclirAutoXP
Run Description	English topics, automatic, title+desc
Number of Topics:	25

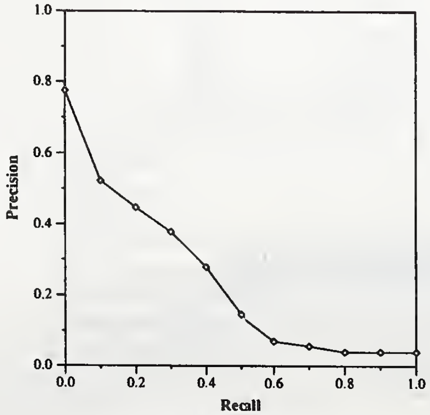
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	1722

Recall Level Averages	
Recall	Precision
0.00	0.7750
0.10	0.5224
0.20	0.4475
0.30	0.3783
0.40	0.2783
0.50	0.1465
0.60	0.0702
0.70	0.0567
0.80	0.0400
0.90	0.0400
1.00	0.0400

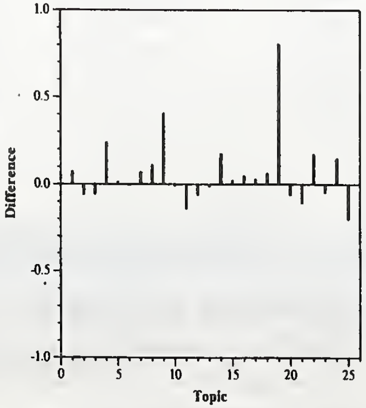
Mean average precision	
non-interpolated	0.2340

Document Level Averages	
	Precision
At 5 docs	0.5760
At 10 docs	0.5440
At 15 docs	0.5200
At 20 docs	0.5000
At 30 docs	0.4787
At 100 docs	0.3456
At 200 docs	0.2292
At 500 docs	0.1214
At 1000 docs	0.0689

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2889



Recall-Precision Curve



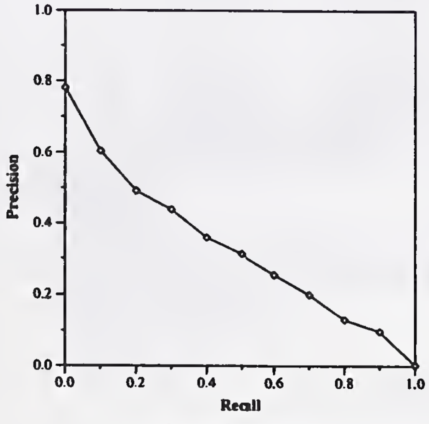
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UMass1
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	25

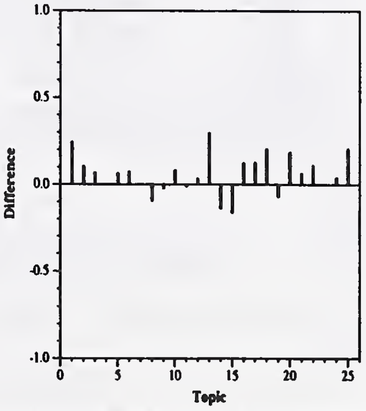
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2779

Recall Level Averages	
Recall	Precision
0.00	0.7810
0.10	0.6035
0.20	0.4914
0.30	0.4394
0.40	0.3597
0.50	0.3136
0.60	0.2546
0.70	0.2001
0.80	0.1311
0.90	0.0977
1.00	0.0022
Mean average precision	
non-interpolated	0.3129

Document Level Averages	
	Precision
At 5 docs	0.6080
At 10 docs	0.5320
At 15 docs	0.5253
At 20 docs	0.5080
At 30 docs	0.4813
At 100 docs	0.3544
At 200 docs	0.2806
At 500 docs	0.1770
At 1000 docs	0.1112
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3773



Recall-Precision Curve



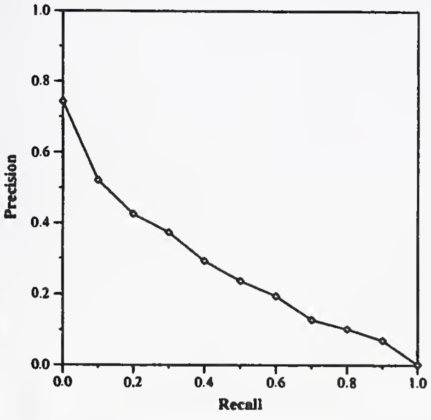
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UMass2
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	25

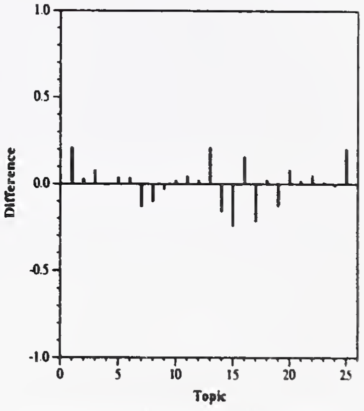
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2484

Recall Level Averages	
Recall	Precision
0.00	0.7435
0.10	0.5217
0.20	0.4257
0.30	0.3734
0.40	0.2919
0.50	0.2371
0.60	0.1956
0.70	0.1291
0.80	0.1025
0.90	0.0703
1.00	0.0013
Mean average precision	
non-interpolated	0.2597

Document Level Averages	
	Precision
At 5 docs	0.5600
At 10 docs	0.5080
At 15 docs	0.4560
At 20 docs	0.4580
At 30 docs	0.4293
At 100 docs	0.3124
At 200 docs	0.2438
At 500 docs	0.1526
At 1000 docs	0.0994
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3109



Recall-Precision Curve



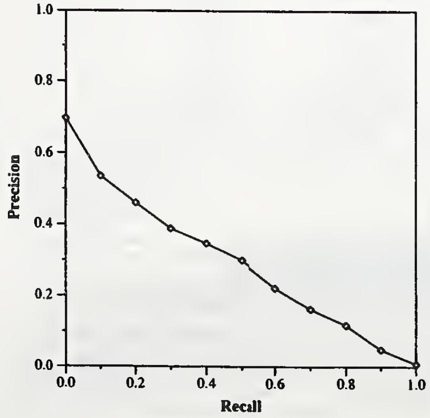
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UMass3
Run Description	English topics, automatic, title+desc
Number of Topics	25

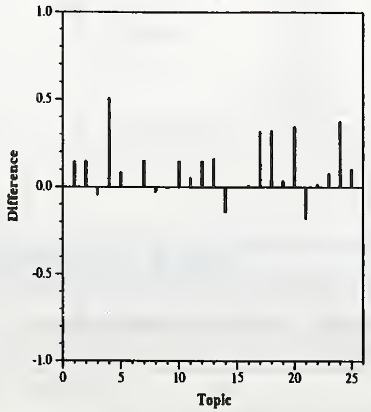
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	2751

Recall Level Averages	
Recall	Precision
0.00	0.6969
0.10	0.5361
0.20	0.4611
0.30	0.3888
0.40	0.3476
0.50	0.3003
0.60	0.2213
0.70	0.1626
0.80	0.1178
0.90	0.0487
1.00	0.0080
Mean average precision	
non-interpolated	0.2795

Document Level Averages	
	Precision
At 5 docs	0.5120
At 10 docs	0.4920
At 15 docs	0.4853
At 20 docs	0.4780
At 30 docs	0.4653
At 100 docs	0.3820
At 200 docs	0.2890
At 500 docs	0.1788
At 1000 docs	0.1100
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3197



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UMass4
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	25

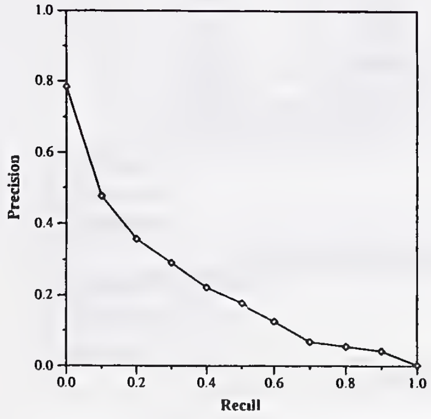
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	1972

Recall Level Averages	
Recall	Precision
0.00	0.7842
0.10	0.4771
0.20	0.3570
0.30	0.2918
0.40	0.2227
0.50	0.1780
0.60	0.1256
0.70	0.0681
0.80	0.0551
0.90	0.0422
1.00	0.0017

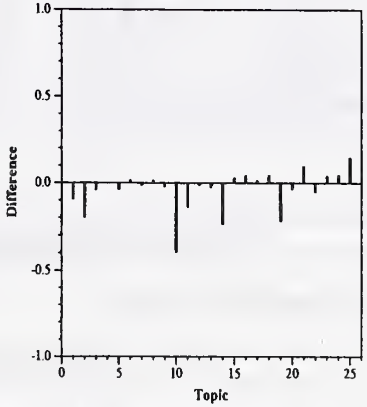
Mean average precision	
non-interpolated	0.2104

Document Level Averages	
	Precision
At 5 docs	0.5440
At 10 docs	0.5000
At 15 docs	0.4613
At 20 docs	0.4380
At 30 docs	0.3973
At 100 docs	0.2716
At 200 docs	0.2052
At 500 docs	0.1252
At 1000 docs	0.0789

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2663



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	shefma
Run Description	Arabic topics, automatic, title
Number of Topics	25

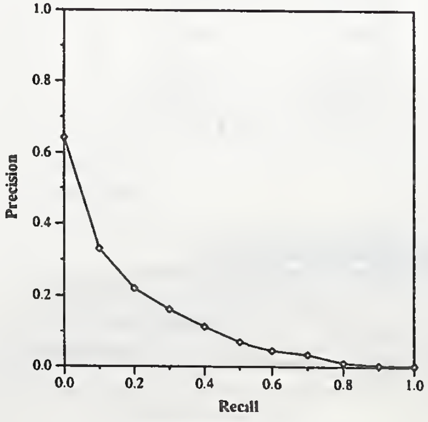
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	1468

Recall Level Averages	
Recall	Precision
0.00	0.6420
0.10	0.3303
0.20	0.2205
0.30	0.1626
0.40	0.1144
0.50	0.0705
0.60	0.0455
0.70	0.0348
0.80	0.0108
0.90	0.0028
1.00	0.0021

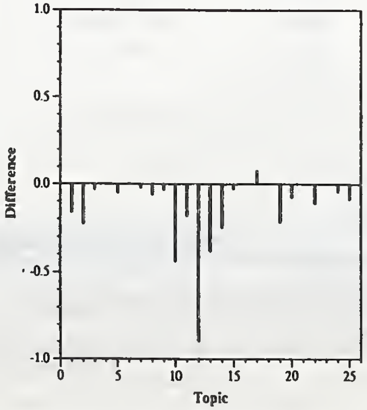
Mean average precision	
non-interpolated	0.1226

Document Level Averages	
	Precision
At 5 docs	0.4080
At 10 docs	0.3840
At 15 docs	0.3573
At 20 docs	0.3420
At 30 docs	0.3080
At 100 docs	0.2128
At 200 docs	0.1554
At 500 docs	0.0939
At 1000 docs	0.0587

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1846



Recall-Preci:ion Curve



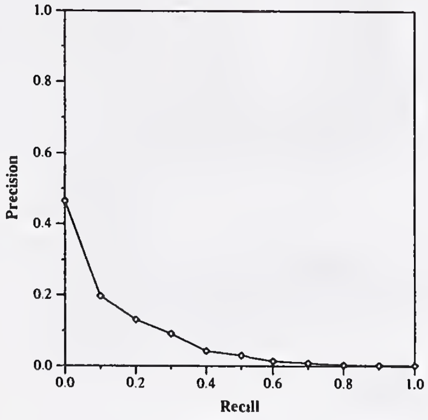
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	shefea
Run Description	English topics, automatic, title
Number of Topics:	25

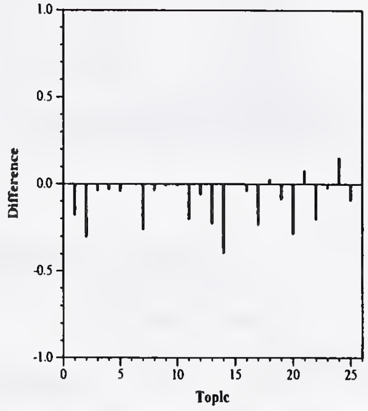
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	1107

Recall Level Averages	
Recall	Precision
0.00	0.4643
0.10	0.1968
0.20	0.1314
0.30	0.0923
0.40	0.0431
0.50	0.0306
0.60	0.0141
0.70	0.0088
0.80	0.0029
0.90	0.0025
1.00	0.0018
Mean average precision	
non-interpolated	0.0698

Document Level Averages	
	Precision
At 5 docs	0.2800
At 10 docs	0.2560
At 15 docs	0.2453
At 20 docs	0.2300
At 30 docs	0.2200
At 100 docs	0.1492
At 200 docs	0.1072
At 500 docs	0.0664
At 1000 docs	0.0443
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1175



Recall-Precision Curve



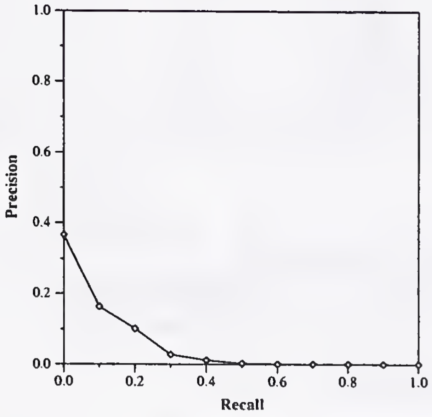
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	shefeaa
Run Description	English topics, automatic, title
Number of Topics:	25

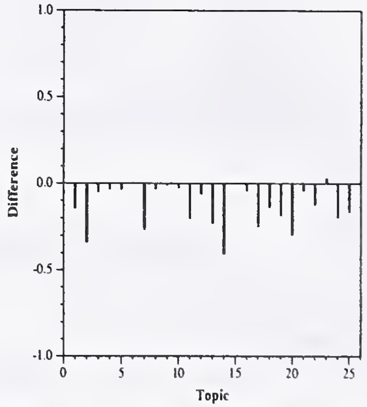
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	922

Recall Level Averages	
Recall	Precision
0.00	0.3666
0.10	0.1633
0.20	0.1021
0.30	0.0277
0.40	0.0118
0.50	0.0026
0.60	0.0007
0.70	0.0006
0.80	0.0005
0.90	0.0000
1.00	0.0000
Mean average precision	
non-interpolated	0.0408

Document Level Averages	
	Precision
At 5 docs	0.2480
At 10 docs	0.2040
At 15 docs	0.1840
At 20 docs	0.1740
At 30 docs	0.1493
At 100 docs	0.1016
At 200 docs	0.0796
At 500 docs	0.0511
At 1000 docs	0.0369
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0850



Recall-Precision Curve



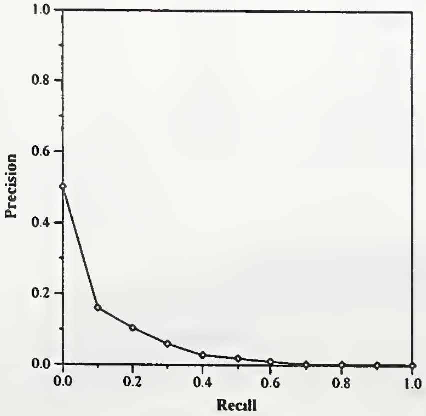
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	sheffea
Run Description	French topics, automatic, title
Number of Topics:	25

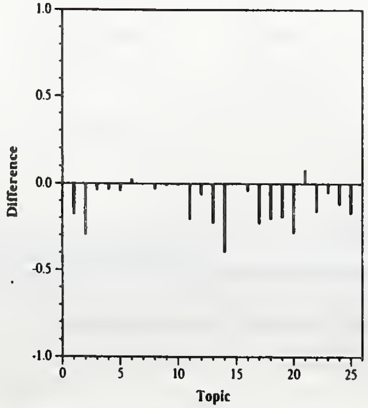
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	982

Recall Level Averages	
Recall	Precision
0.00	0.5027
0.10	0.1601
0.20	0.1043
0.30	0.0596
0.40	0.0283
0.50	0.0192
0.60	0.0106
0.70	0.0030
0.80	0.0029
0.90	0.0025
1.00	0.0018
Mean average precision	
non-interpolated	0.0551

Document Level Averages	
	Precision
At 5 docs	0.2800
At 10 docs	0.2640
At 15 docs	0.2240
At 20 docs	0.2100
At 30 docs	0.1880
At 100 docs	0.1344
At 200 docs	0.1014
At 500 docs	0.0604
At 1000 docs	0.0393
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0974



Recall-Preciiion Curve



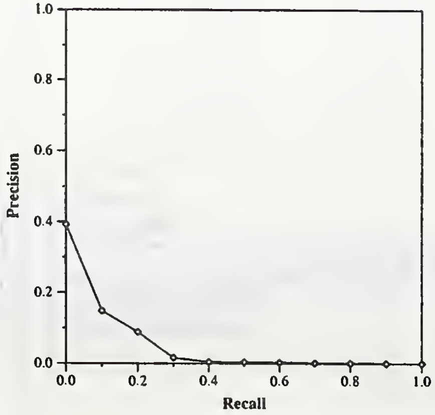
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	sheffeaa
Run Description	French topics, automatic, title
Number of Topics:	25

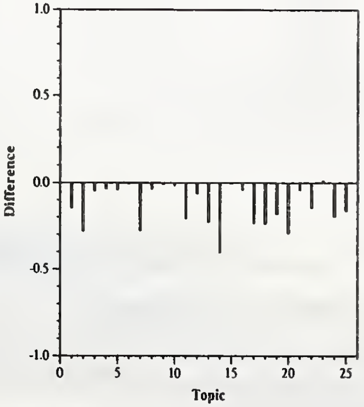
Total number of documents over all topics	
Retrieved:	25000
Relevant:	4122
Rel-ret:	731

Recall Level Averages	
Recall	Precision
0.00	0.3924
0.10	0.1481
0.20	0.0887
0.30	0.0158
0.40	0.0039
0.50	0.0034
0.60	0.0027
0.70	0.0006
0.80	0.0005
0.90	0.0000
1.00	0.0000
Mean average precision	
non-interpolated	0.0367

Document Level Averages	
	Precision
At 5 docs	0.2720
At 10 docs	0.2120
At 15 docs	0.1920
At 20 docs	0.1720
At 30 docs	0.1520
At 100 docs	0.1024
At 200 docs	0.0684
At 500 docs	0.0398
At 1000 docs	0.0292
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0687



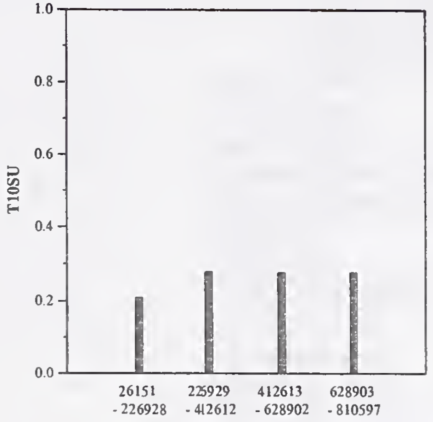
Recall-Precision Curve



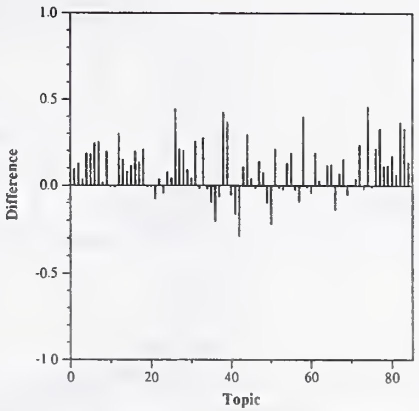
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	ICTAdaFT10Fa
Subtask	adaptive
Resources used:	none
Optimized for	
Number of Topics:	84
Retrieved:	573495
Rel-ret:	338958
Macro average recall	0.364
Macro average precision	0.427
Mean T10SU	0.206
Mean F-Beta	0.387
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.209
226929 - 412612	0.278
412613 - 628902	0.276
628903 - 810597	0.276



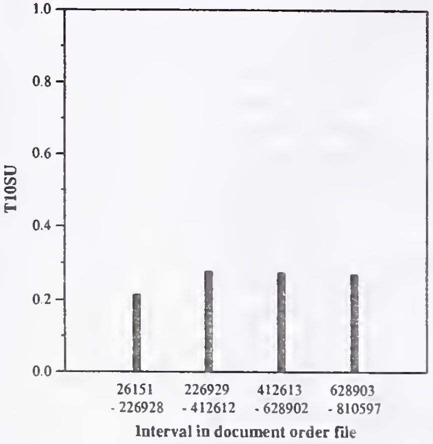
T10SU within time periods



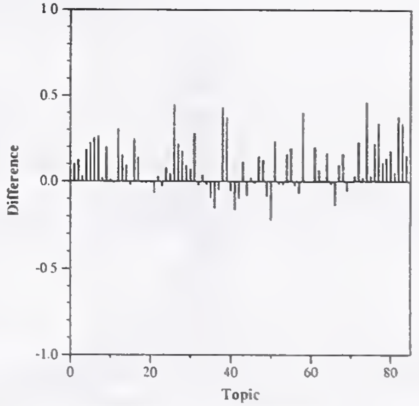
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	ICTAdaFT10Ua
Subtask	adaptive
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	527761
Rel-ret:	326201
Macro average recall	0.325
Macro average precision	0.421
Mean T10SU	0.204
Mean F-Beta	0.368
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.213
226929 - 412612	0.276
412613 - 628902	0.273
628903 - 810597	0.269



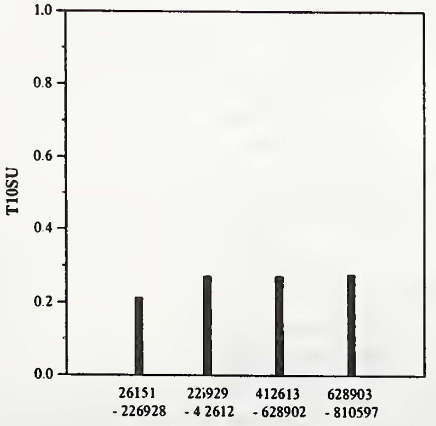
T10SU within time periods



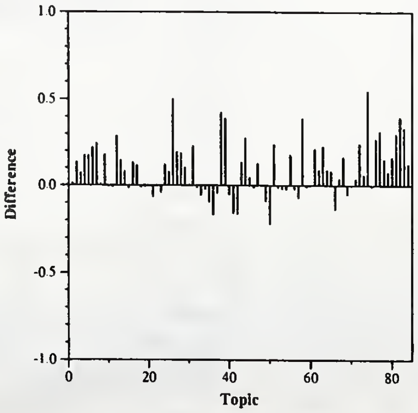
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	ICTAdaFT10Ub
Subtask	adaptive
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	605659
Rel-ret:	356509
Macro average recall	0.356
Macro average precision	0.398
Mean T10SU	0.205
Mean F-Beta	0.366
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.212
226929 - 412612	0.272
412613 - 628902	0.272
628903 - 810597	0.276



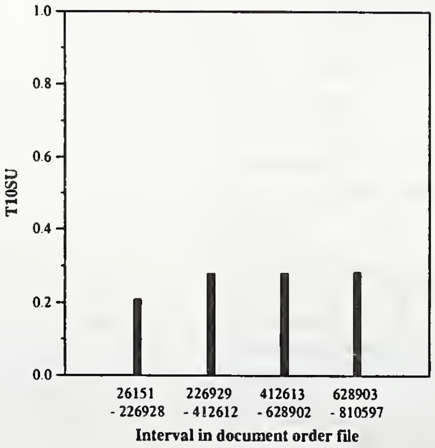
T10SU within time periods



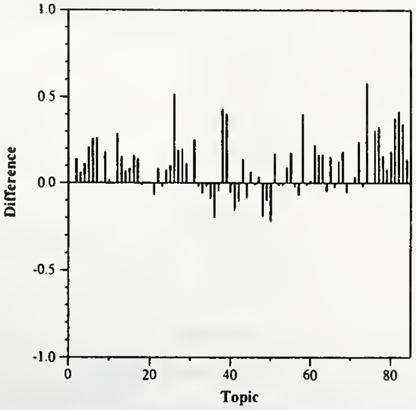
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	ICTAdaFT10Uc
Subtask	adaptive
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	595586
Rel-ret:	360399
Macro average recall	0.330
Macro average precision	0.390
Mean T10SU	0.207
Mean F-Beta	0.354
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.209
226929 - 412612	0.279
412613 - 628902	0.280
628903 - 810597	0.283



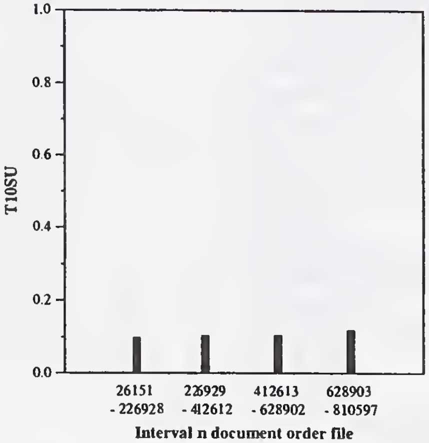
T10SU within time periods



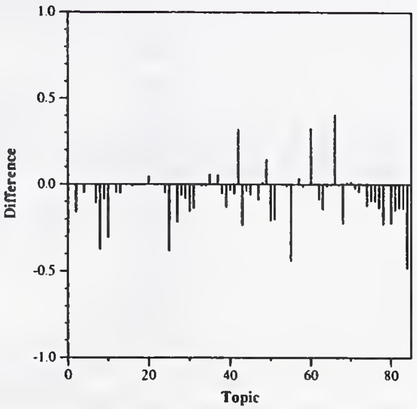
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	CL01afa
Subtask	adaptive
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	11224
Rel-ret:	8324
Macro average recall	0.032
Macro average precision	0.474
Mean T10SU	0.054
Mean F-Beta	0.081
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.098
226929 - 412612	0.104
412613 - 628902	0.105
628903 - 810597	0.119



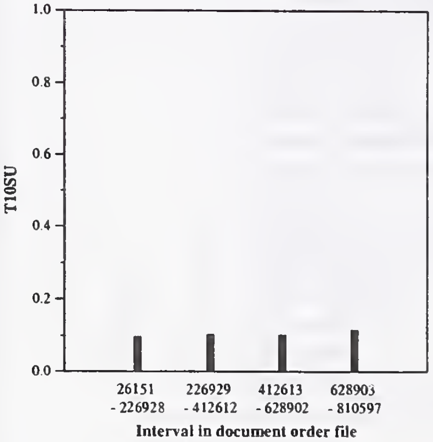
T10SU within time periods



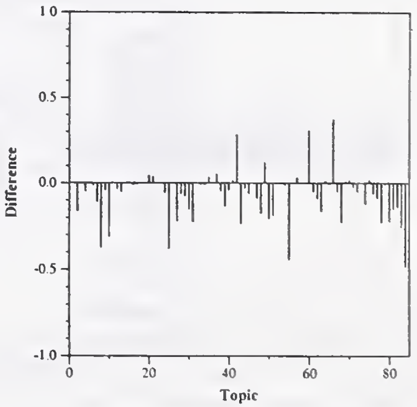
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	CL01afb
Subtask	adaptive
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	14156
Rel-ret:	9210
Macro average recall	0.031
Macro average precision	0.454
Mean T10SU	0.051
Mean F-Beta	0.075
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.097
226929 - 412612	0.103
412613 - 628902	0.102
628903 - 810597	0.116



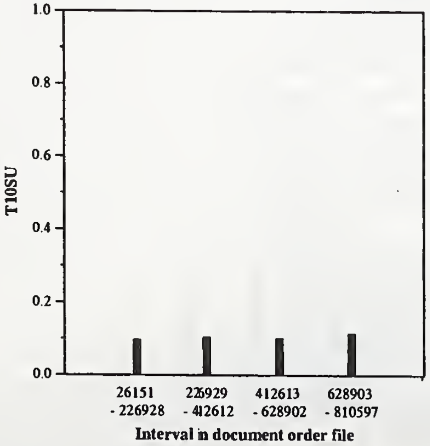
T10SU within time periods



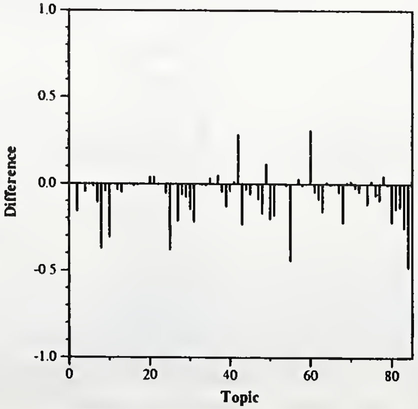
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	CL01afc
Subtask	adaptive
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	14816
Rel-ret:	9781
Macro average recall	0.029
Macro average precision	0.442
Mean T10SU	0.050
Mean F-Beta	0.070
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.097
226929 - 412612	0.104
412613 - 628902	0.101
628903 - 810597	0.114



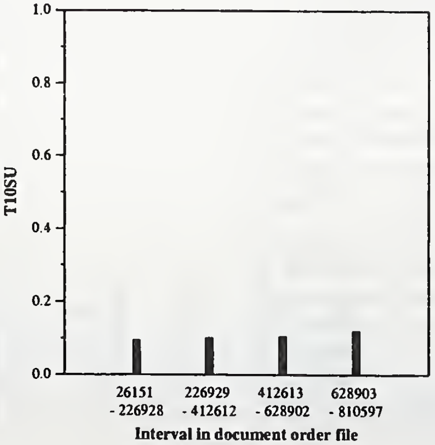
T10SU within time periods



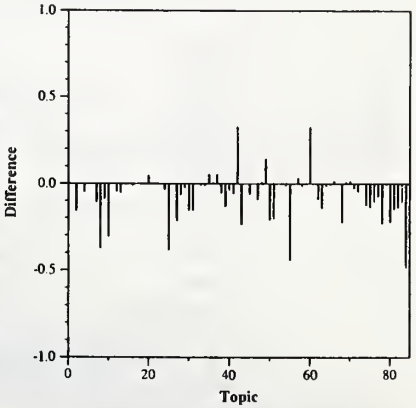
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	CL01afd
Subtask	adaptive
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	15931
Rel-ret:	11522
Macro average recall	0.028
Macro average precision	0.462
Mean T10SU	0.051
Mean F-Beta	0.078
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.094
226929 - 412612	0.101
412613 - 628902	0.105
628903 - 810597	0.120



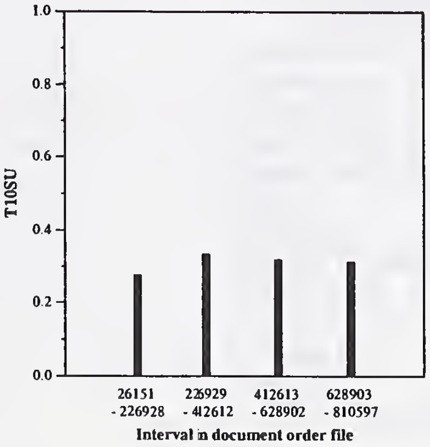
T10SU within time periods



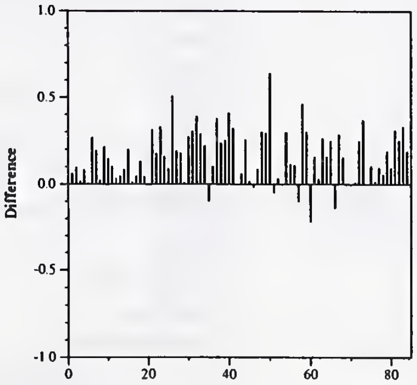
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	CMUCATmr10
Subtask	adaptive
Resources used:	other TREC
Optimized for	T10U
Number of Topics:	84
Retrieved:	263374
Rel-ret:	214248
Macro average recall	0.344
Macro average precision	0.657
Mean T10SU	0.263
Mean F-Beta	0.499
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.276
226929 - 412612	0.334
412613 - 628902	0.319
628903 - 810597	0.313



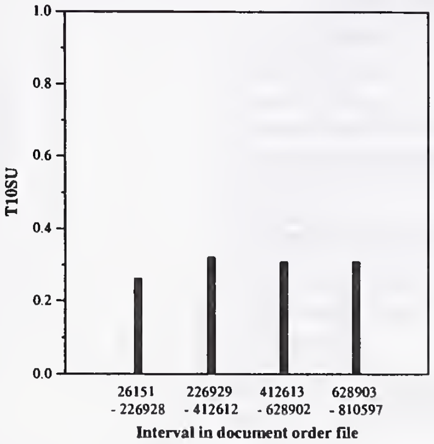
T10SU within time periods



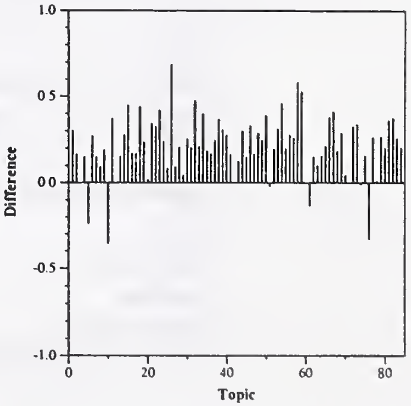
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	CMUCATmrf5
Subtask	adaptive
Resources used:	other TREC
Optimized for	F-beta
Number of Topics:	84
Retrieved:	269402
Rel-ret:	214315
Macro average recall	0.341
Macro average precision	0.661
Mean T10SU	0.251
Mean F-Beta	0.489
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.261
226929 - 412612	0.320
412613 - 628902	0.308
628903 - 810597	0.308



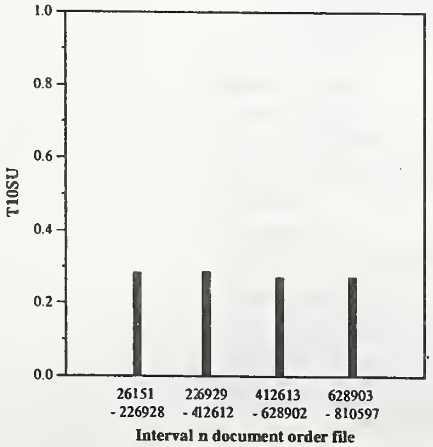
T10SU within time periods



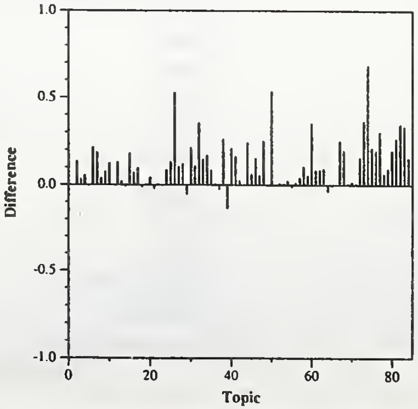
Difference from Median in F-beta per Topic

Summary Statistics	
Run ID:	CMUCATsr10
Subtask	adaptive
Resources used:	other TREC
Optimized for	T10U
Number of Topics:	84
Retrieved:	342552
Rel-ret:	245386
Macro average recall	0.248
Macro average precision	0.603
Mean T10SU	0.228
Mean F-Beta	0.415
Zero returns	10

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.283
226929 - 412612	0.287
412613 - 628902	0.271
628903 - 810597	0.272



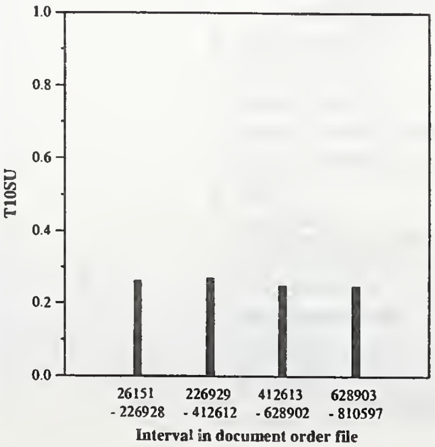
T10SU within time periods



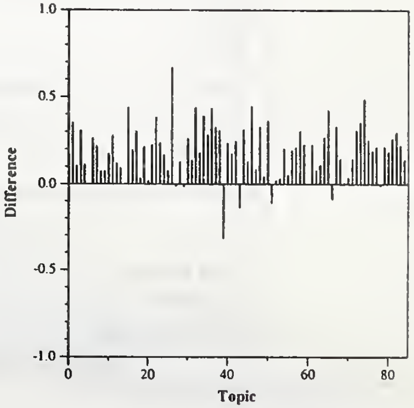
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	CMUCATsrf5
Subtask	adaptive
Resources used:	other TREC
Optimized for	F-beta
Number of Topics:	84
Retrieved:	301856
Rel-ret:	215554
Macro average recall	0.248
Macro average precision	0.658
Mean T10SU	0.211
Mean F-Beta	0.467
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.261
226929 - 412612	0.268
412613 - 628902	0.249
628903 - 810597	0.247



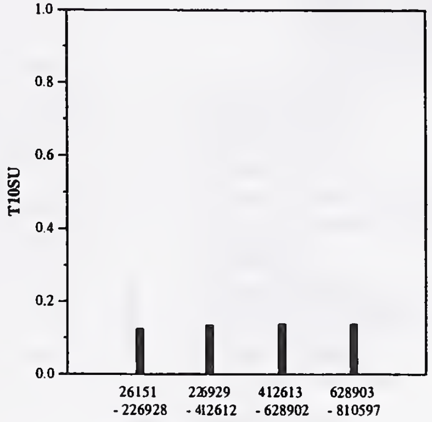
T10SU within time periods



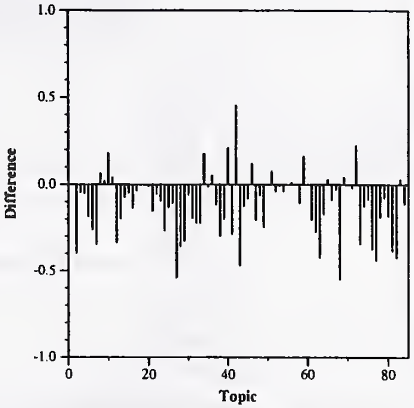
Difference from Median in F-beta per Topic

Summary Statistics	
Run ID:	CMUDIRFLM
Subtask	adaptive
Resources used:	none
Optimized for	F-beta
Number of Topics:	84
Retrieved:	58427
Rel-ret:	45223
Macro average recall	0.066
Macro average precision	0.486
Mean T10SU	0.080
Mean F-Beta	0.163
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.123
226929 - 412612	0.133
412613 - 628902	0.137
628903 - 810597	0.137



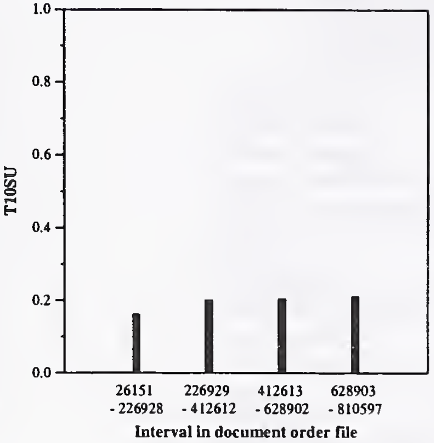
T10SU within time periods



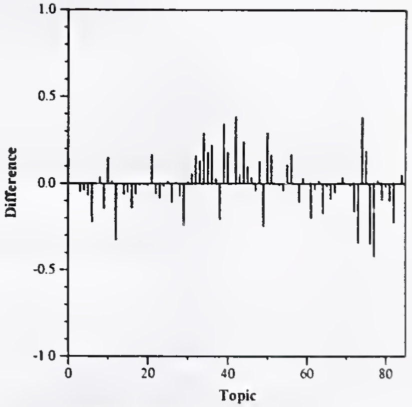
Difference from Median in F-beta per Topic

Summary Statistics	
Run ID:	CMUDIRFRM
Subtask	adaptive
Resources used:	none
Optimized for	F-beta
Number of Topics:	84
Retrieved:	157705
Rel-ret:	125231
Macro average recall	0.142
Macro average precision	0.556
Mean T10SU	0.143
Mean F-Beta	0.275
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.160
226929 - 412612	0.200
412613 - 628902	0.204
628903 - 810597	0.211



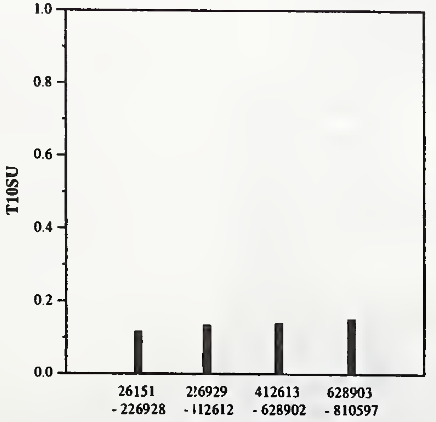
T10SU within time periods



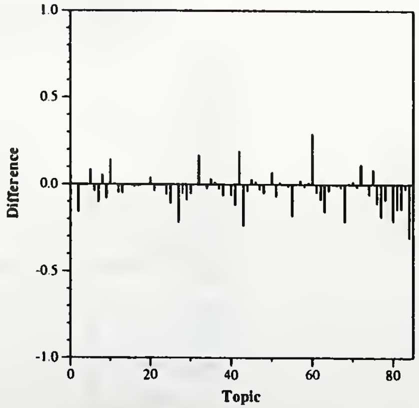
Difference from Median in F-beta per Topic

Summary Statistics	
Run ID:	CMUDIRULM
Subtask	adaptive
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	98347
Rel-ret:	58780
Macro average recall	0.074
Macro average precision	0.467
Mean T10SU	0.081
Mean F-Beta	0.158
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.116
226929 - 412612	0.134
412613 - 628902	0.140
628903 - 810597	0.151



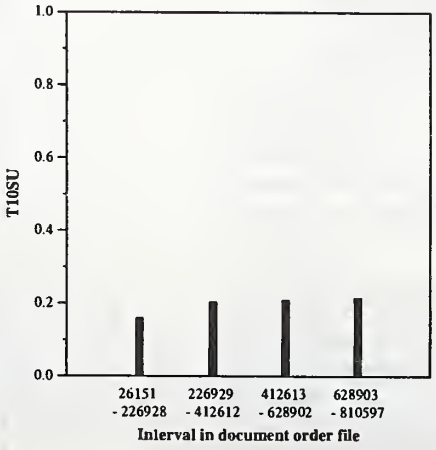
T10SUwithin time periods



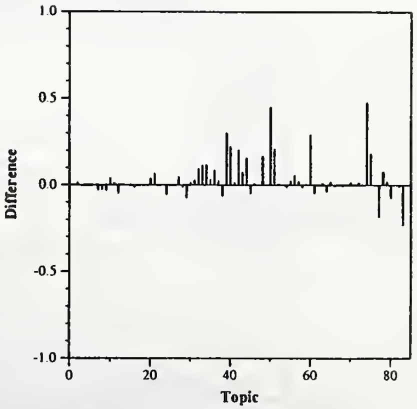
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	CMUDIRURM
Subtask	adaptive
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	222641
Rel-ret:	151292
Macro average recall	0.159
Macro average precision	0.505
Mean T10SU	0.144
Mean F-Beta	0.273
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.159
226929 - 412612	0.204
412613 - 628902	0.209
628903 - 810597	0.215



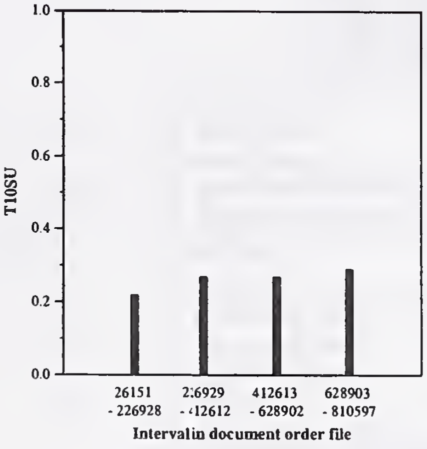
T10SU within time periods



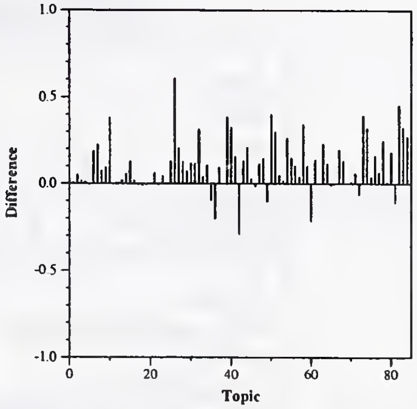
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	FDUT10AF1
Subtask	adaptive
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	429034
Rel-ret:	271676
Macro average recall	0.330
Macro average precision	0.505
Mean T10SU	0.215
Mean F-Beta	0.404
Zero returns	1

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.218
226929 - 412612	0.268
412613 - 628902	0.268
628903 - 810597	0.289



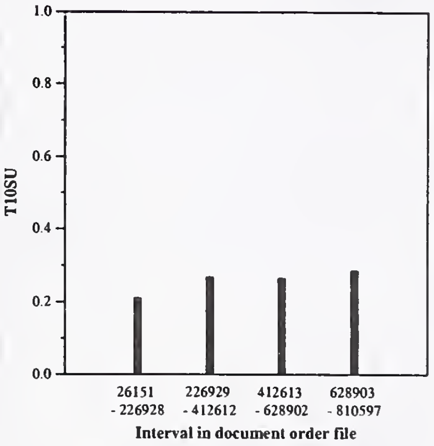
T10SUwithin time periods



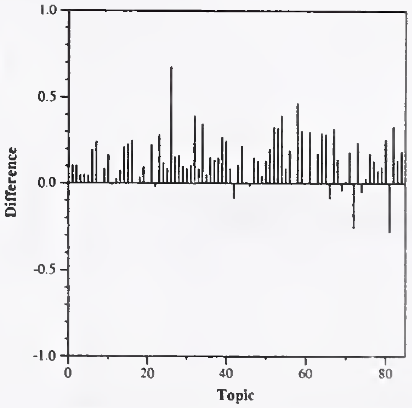
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	FDUT10AF4
Subtask	adaptive
Resources used:	none
Optimized for	F-beta
Number of Topics:	84
Retrieved:	502209
Rel-ret:	285135
Macro average recall	0.363
Macro average precision	0.493
Mean T10SU	0.213
Mean F-Beta	0.414
Zero returns	1

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.210
226929 - 412612	0.267
412613 - 628902	0.265
628903 - 810597	0.285



T10SU within time periods

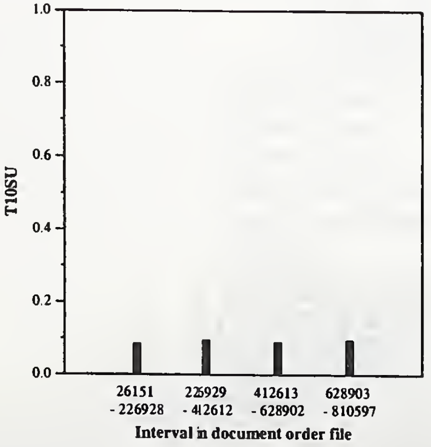


Difference from Median in F-beta per Topic

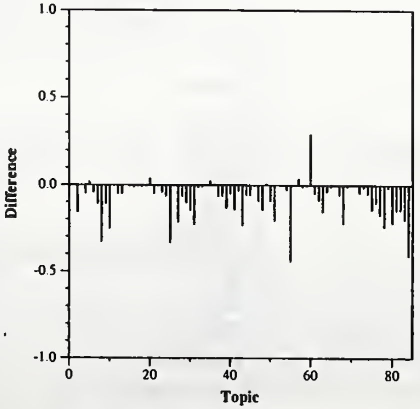
Filtering - adaptive results — Kent Ridge Digital Labs

Summary Statistics	
Run ID:	krdlT10af01
Subtask	adaptive
Resources used:	other TREC, other data
Optimized for	neither
Number of Topics:	84
Retrieved:	13282
Rel-ret:	9369
Macro average recall	0.012
Macro average precision	0.392
Mean T10SU	0.035
Mean F-Beta	0.046
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.085
226929 - 412612	0.095
412613 - 628902	0.090
628903 - 810597	0.095



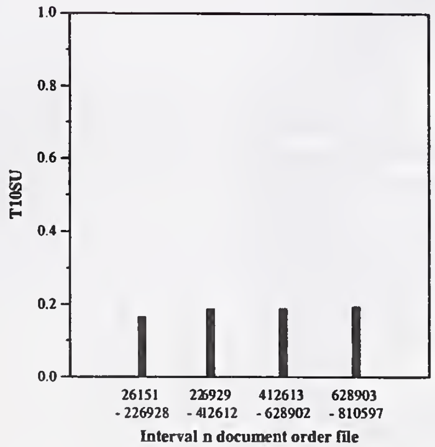
T10SU within time periods



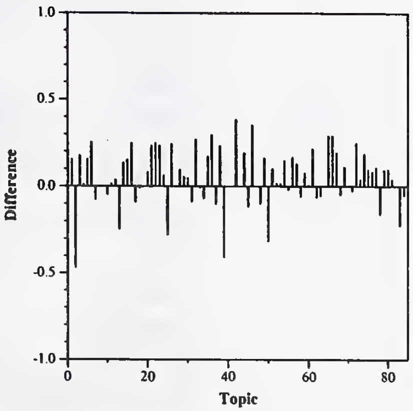
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	ok10f2br
Subtask	adaptive
Resources used:	none
Optimized for	F-beta
Number of Topics:	84
Retrieved:	489686
Rel-ret:	239812
Macro average recall	0.273
Macro average precision	0.427
Mean T10SU	0.137
Mean F-Beta	0.330
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.165
226929 - 412612	0.187
412613 - 628902	0.188
628903 - 810597	0.193



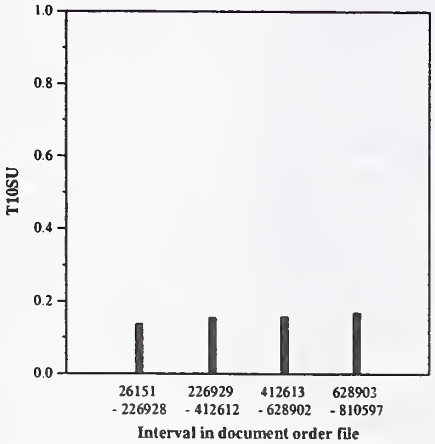
T10SU within time periods



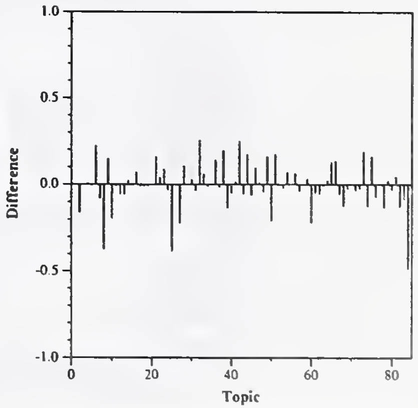
Difference from Median in F-beta per Topic

Summary Statistics	
Run ID:	ok10f2ur
Subtask	adaptive
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	387281
Rel-ret:	166027
Macro average recall	0.214
Macro average precision	0.368
Mean T10SU	0.104
Mean F-Beta	0.254
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.136
226929 - 412612	0.154
412613 - 628902	0.157
628903 - 810597	0.168



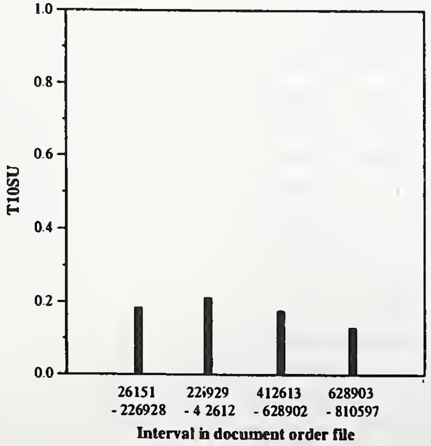
T10SU within time periods



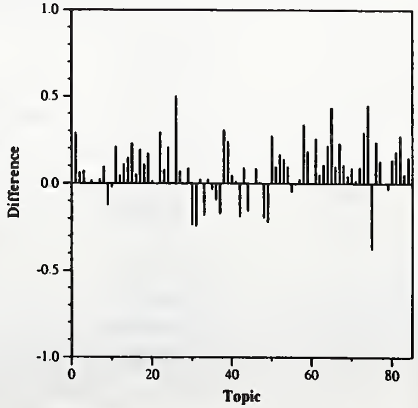
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	KUNaF
Subtask	adaptive
Resources used:	none
Optimized for	F-beta
Number of Topics:	84
Retrieved:	889015
Rel-ret:	313967
Macro average recall	0.518
Macro average precision	0.393
Mean T10SU	0.141
Mean F-Beta	0.356
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.184
226929 - 412612	0.211
412613 - 628902	0.175
628903 - 810597	0.129



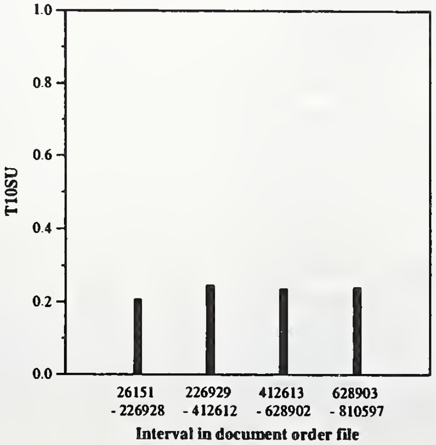
T10SU within time periods



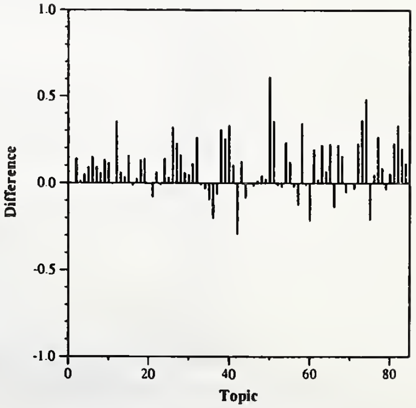
Difference from Median in F-beta per Topic

Summary Statistics	
Run ID:	KUNaU
Subtask	adaptive
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	493653
Rel-ret:	280178
Macro average recall	0.440
Macro average precision	0.509
Mean T10SU	0.203
Mean F-Beta	0.437
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.206
226929 - 412612	0.246
412613 - 628902	0.236
628903 - 810597	0.240



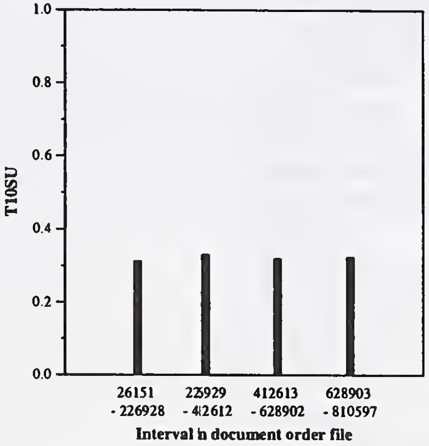
T10SU within time periods



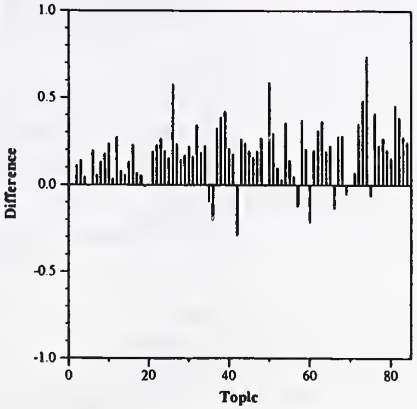
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	oraAU082201
Subtask	adaptive
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	811181
Rel-ret:	460769
Macro average recall	0.496
Macro average precision	0.538
Mean T10SU	0.291
Mean F-Beta	0.519
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.312
226929 - 412612	0.330
412613 - 628902	0.319
628903 - 810597	0.324



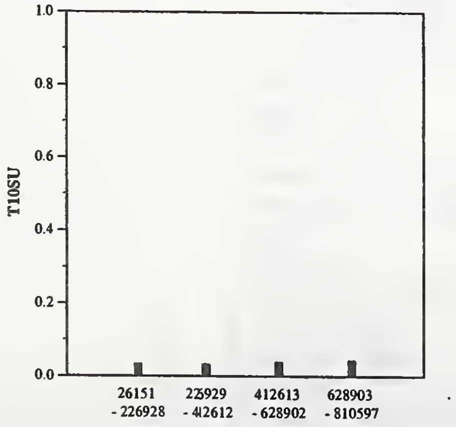
T10SU within time periods



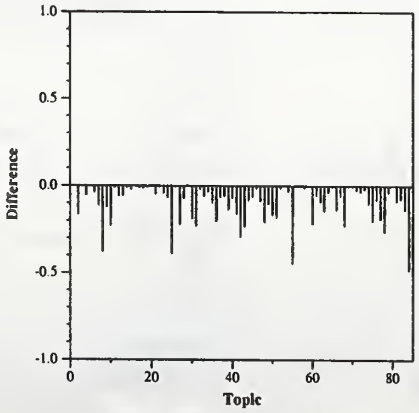
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	RUA301
Subtask	adaptive
Resources used:	none
Optimized for	neither
Number of Topics:	84
Retrieved:	494219
Rel-ret:	92732
Macro average recall	0.167
Macro average precision	0.219
Mean T10SU	0.015
Mean F-Beta	0.123
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.033
226929 - 412612	0.032
412613 - 628902	0.038
628903 - 810597	0.043



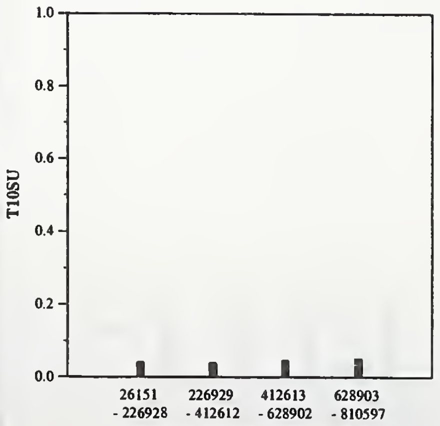
T10SU within time periods



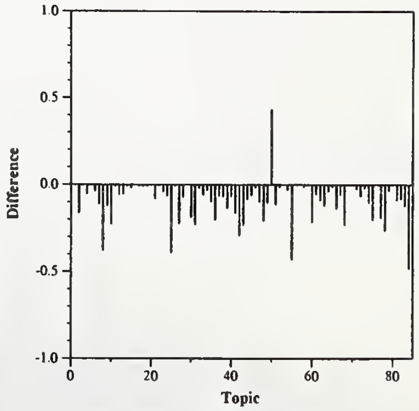
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	RUA501
Subtask	adaptive
Resources used:	none
Optimized for	neither
Number of Topics:	84
Retrieved:	697989
Rel-ret:	131490
Macro average recall	0.213
Macro average precision	0.215
Mean T10SU	0.024
Mean F-Beta	0.144
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.040
226929 - 412612	0.038
412613 - 628902	0.045
628903 - 810597	0.051



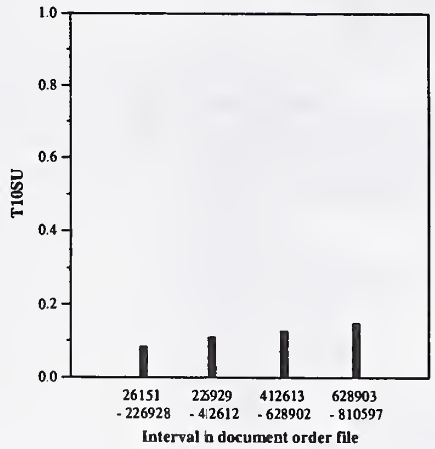
T10SU within time periods



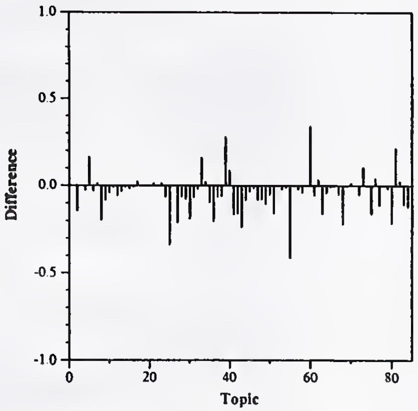
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	serASSAT10ad
Subtask	adaptive
Resources used:	none
Optimized for	neither
Number of Topics:	84
Retrieved:	194801
Rel-ret:	108297
Macro average recall	0.093
Macro average precision	0.376
Mean T10SU	0.069
Mean F-Beta	0.183
Zero returns	1

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.083
226929 - 412612	0.109
412613 - 628902	0.127
628903 - 810597	0.148



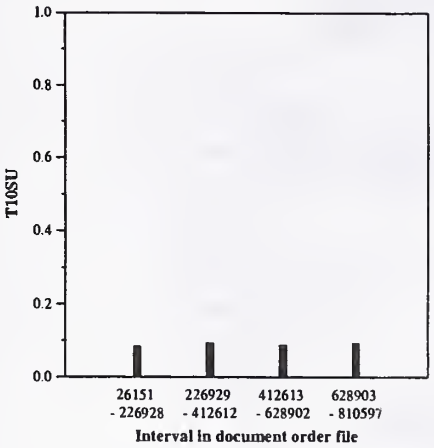
T10SU within time periods



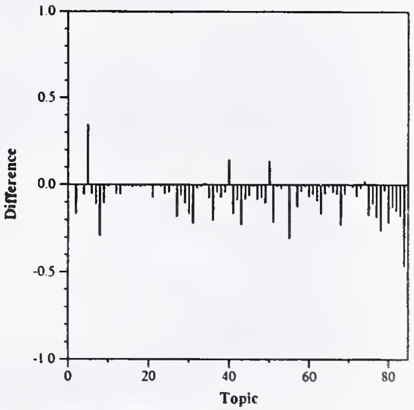
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	serCLST10af
Subtask	adaptive
Resources used:	none
Optimized for	neither
Number of Topics:	84
Retrieved:	43457
Rel-ret:	30971
Macro average recall	0.035
Macro average precision	0.238
Mean T10SU	0.039
Mean F-Beta	0.086
Zero returns	0

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.083
226929 - 412612	0.092
412613 - 628902	0.087
628903 - 810597	0.092



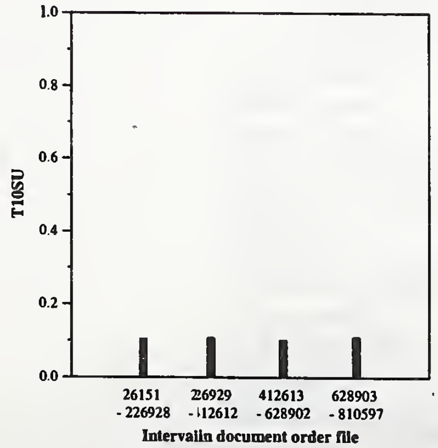
T10SU within time periods



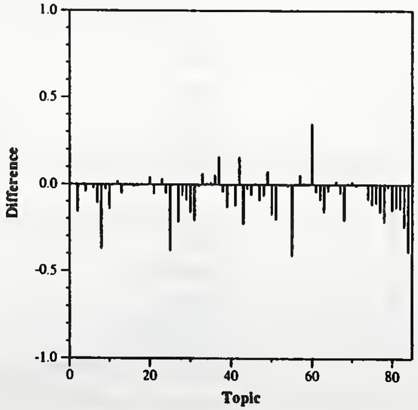
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	UIowa01AF01
Subtask	adaptive
Resources used:	other TREC, other data
Optimized for	T10U
Number of Topics:	84
Retrieved:	15454
Rel-ret:	12606
Macro average recall	0.028
Macro average precision	0.565
Mean T10SU	0.051
Mean F-Beta	0.090
Zero returns	1

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.105
226929 - 412612	0.107
412613 - 628902	0.101
628903 - 810597	0.109



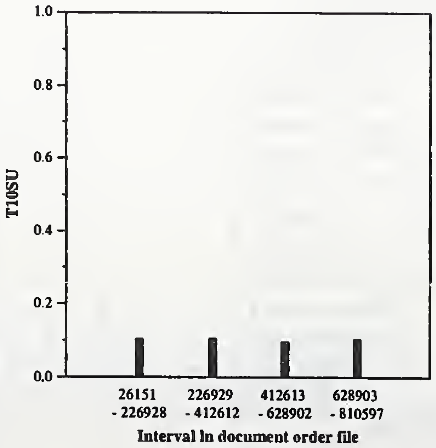
T10SUwithin time periods



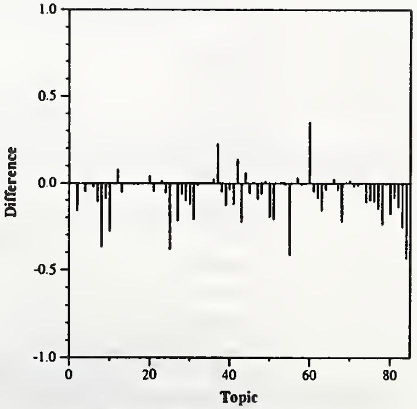
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	UIowa01AF02
Subtask	adaptive
Resources used:	other TREC, other data
Optimized for	T10U
Number of Topics:	84
Retrieved:	12472
Rel-ret:	9871
Macro average recall	0.023
Macro average precision	0.535
Mean T10SU	0.047
Mean F-Beta	0.069
Zero returns	3

T10SU within time periods	
Doc Range	T10SU
26151 - 226928	0.104
226929 - 412612	0.105
412613 - 628902	0.096
628903 - 810597	0.103

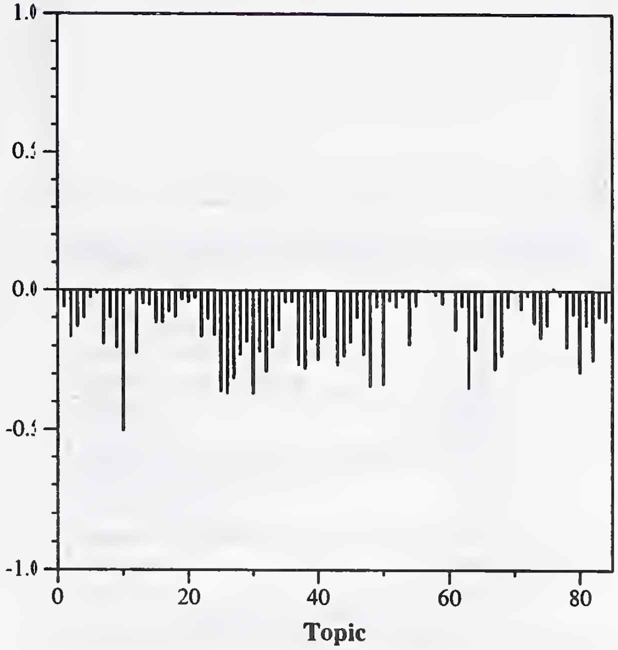


T10SU within time periods



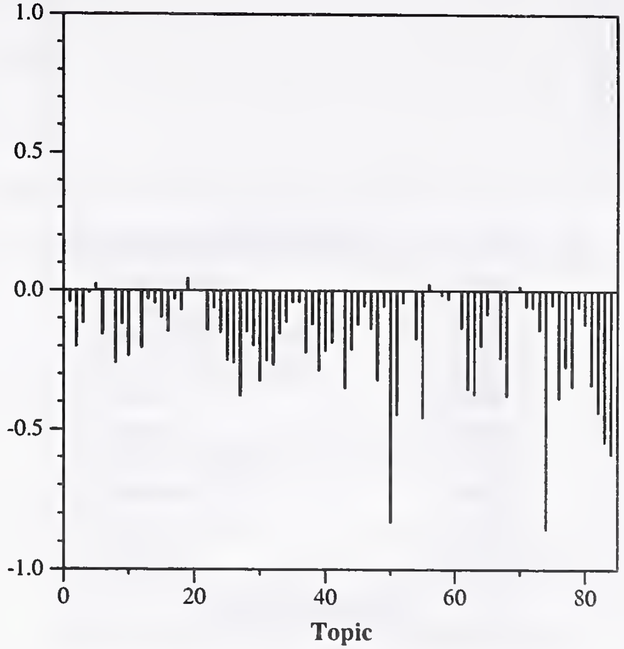
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	apl10fbsvml
Subtask	batch
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	1054134
Rel-ret:	390770
Macro average recall	0.380
Macro average precision	0.303
Mean T10SU	0.115
Mean F-Beta	0.292
Zero returns	6



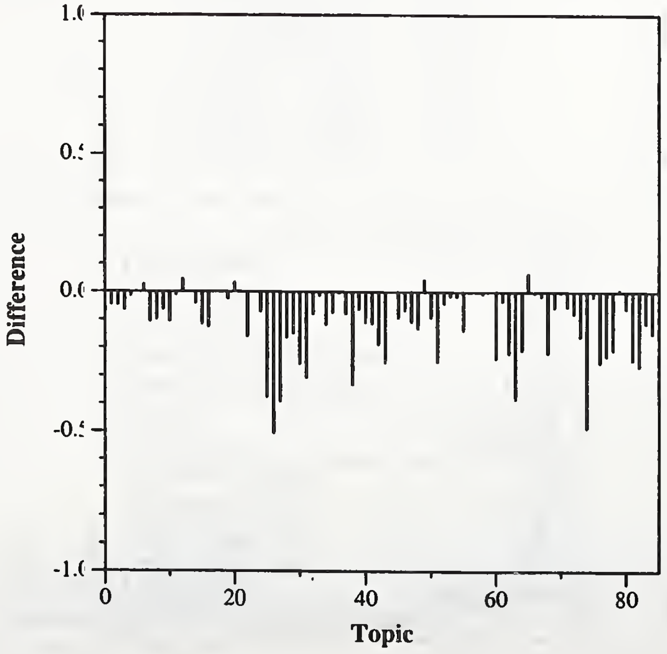
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	apl10fbsvmr
Subtask	batch
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	84192
Rel-ret:	75363
Macro average recall	0.054
Macro average precision	0.627
Mean T10SU	0.081
Mean F-Beta	0.154
Zero returns	25



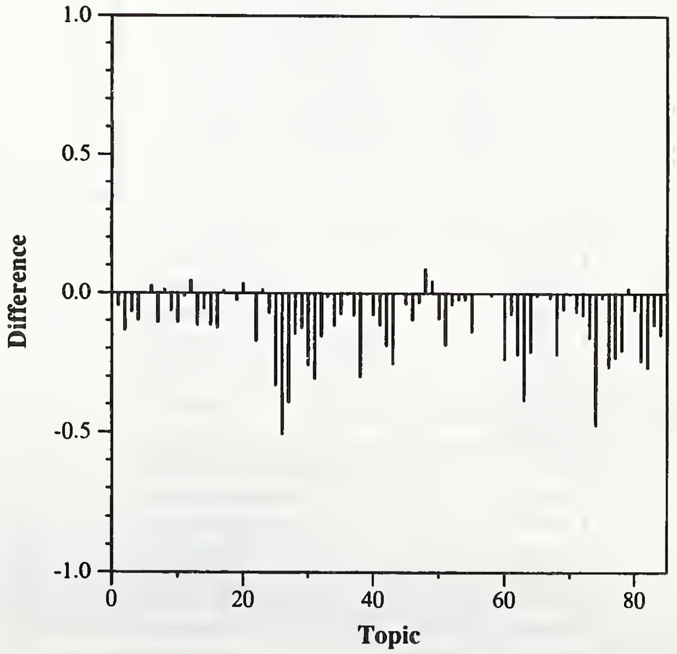
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	CLT10BFA
Subtask	batch
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	269038
Rel-ret:	184076
Macro average recall	0.191
Macro average precision	0.534
Mean T10SU	0.147
Mean F-Beta	0.346
Zero returns	3



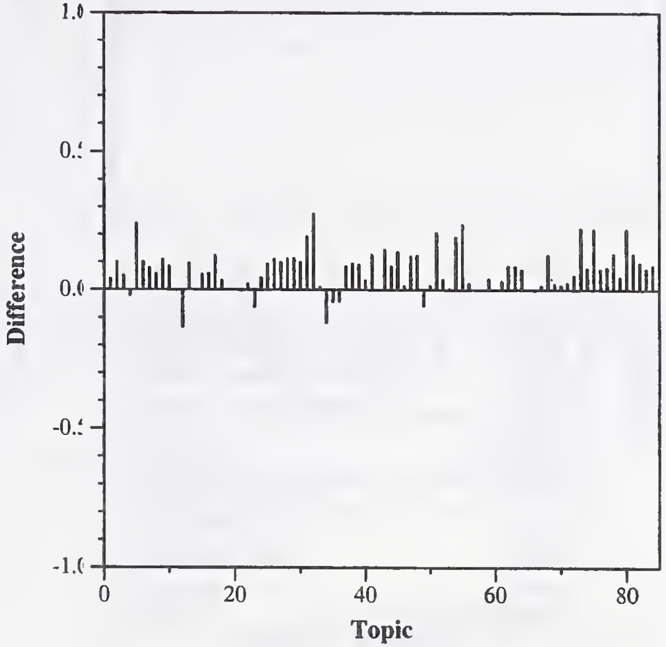
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	CLT10BFB
Subtask	batch
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	376375
Rel-ret:	216476
Macro average recall	0.229
Macro average precision	0.501
Mean T10SU	0.150
Mean F-Beta	0.349
Zero returns	3



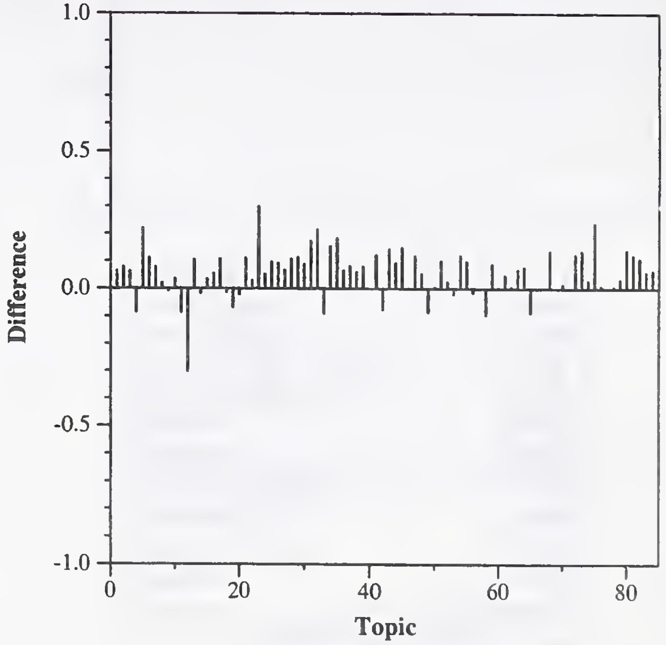
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	CMUCATa210
Subtask	batch
Resources used:	other TREC
Optimized for	T10U
Number of Topics:	84
Retrieved:	591404
Rel-ret:	428393
Macro average recall	0.358
Macro average precision	0.618
Mean T10SU	0.324
Mean F-Beta	0.489
Zero returns	11



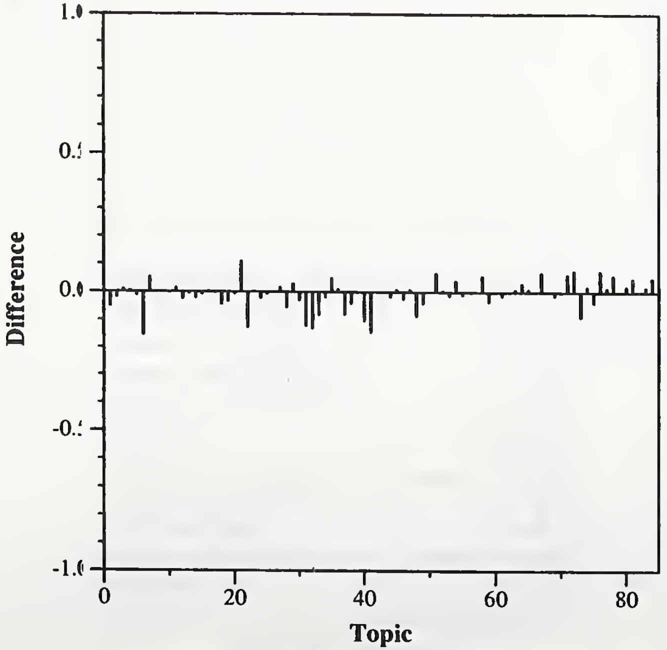
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	CMUCATa2f5
Subtask	batch
Resources used:	other TREC
Optimized for	F-beta
Number of Topics:	84
Retrieved:	456420
Rel-ret:	350174
Macro average recall	0.322
Macro average precision	0.719
Mean T10SU	0.287
Mean F-Beta	0.511
Zero returns	1



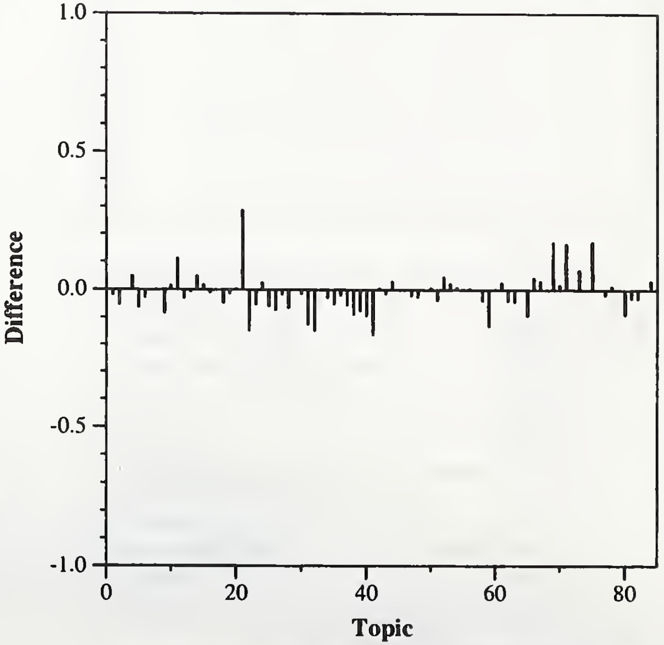
Difference from Median in F-beta per Topic

Summary Statistics	
Run ID:	FDUT10BF1
Subtask	batch
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	595169
Rel-ret:	379405
Macro average recall	0.313
Macro average precision	0.563
Mean T10SU	0.248
Mean F-Beta	0.441
Zero returns	4



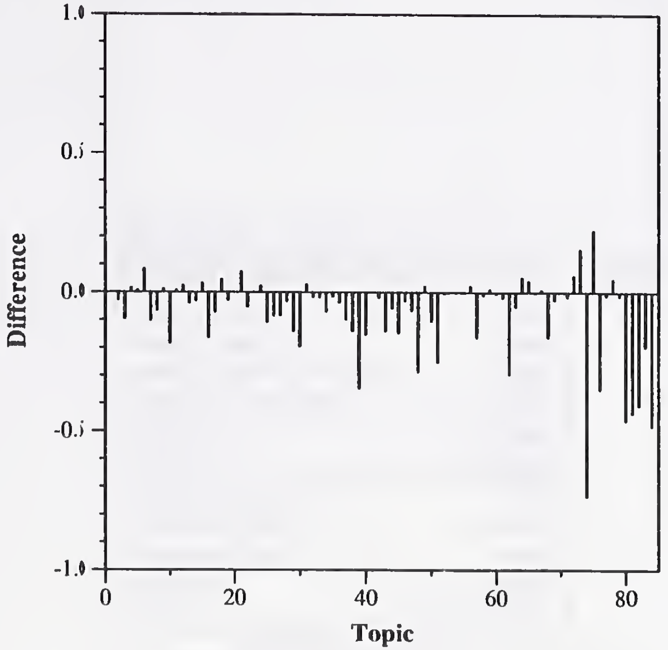
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	FDUT10BF2
Subtask	batch
Resources used:	none
Optimized for	F-beta
Number of Topics:	84
Retrieved:	716569
Rel-ret:	419174
Macro average recall	0.373
Macro average precision	0.526
Mean T10SU	0.244
Mean F-Beta	0.448
Zero returns	1



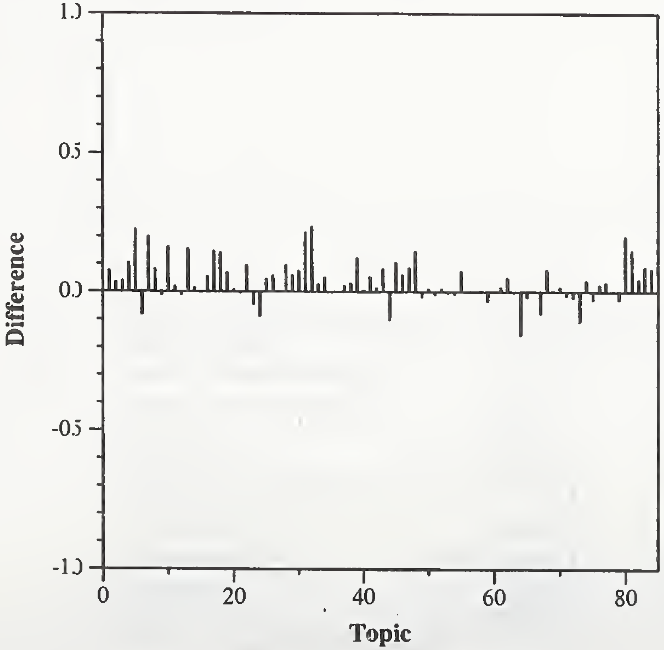
Difference from Median in F-beta per Topic

Summary Statistics	
Run ID:	mer10b
Subtask	batch
Resources used:	other TREC
Optimized for	T10U
Number of Topics:	84
Retrieved:	210059
Rel-ret:	152077
Macro average recall	0.204
Macro average precision	0.631
Mean T10SU	0.181
Mean F-Beta	0.392
Zero returns	0



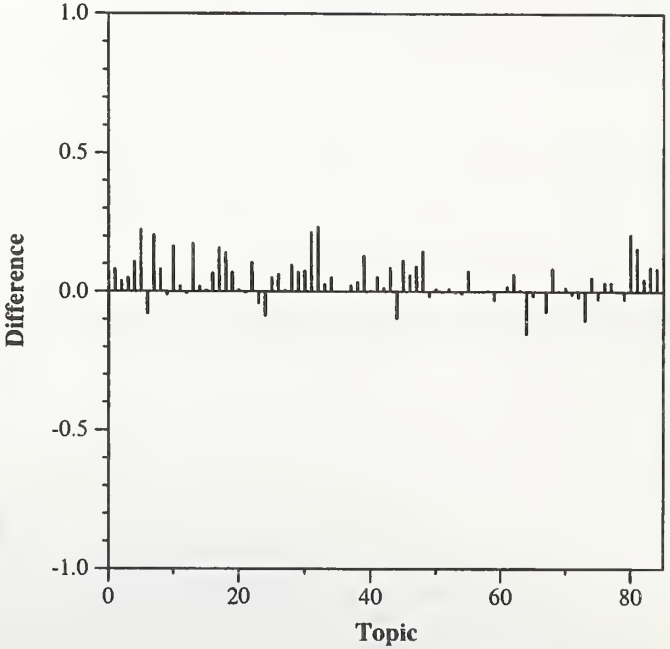
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	KAIST10bf1
Subtask	batch
Resources used:	none
Optimized for	neither
Number of Topics:	84
Retrieved:	408622
Rel-ret:	356044
Macro average recall	0.288
Macro average precision	0.788
Mean T10SU	0.295
Mean F-Beta	0.496
Zero returns	2



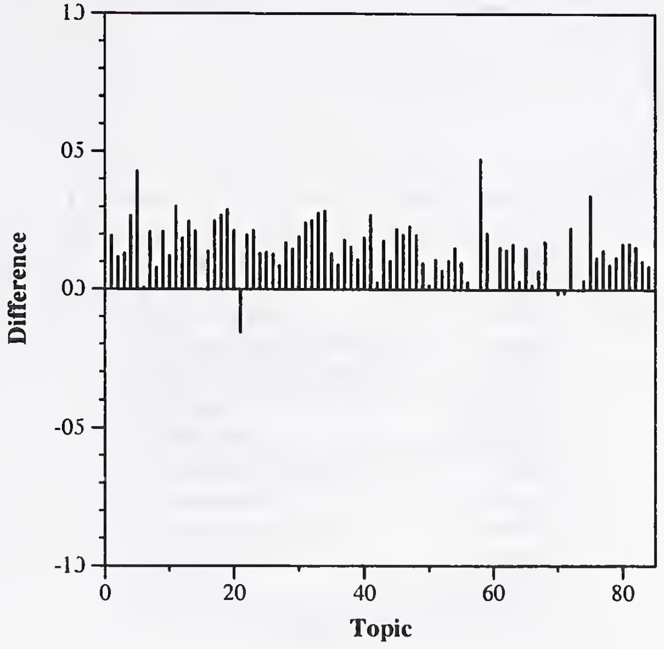
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	KAIST10bf2
Subtask	batch
Resources used:	none
Optimized for	neither
Number of Topics:	84
Retrieved:	421185
Rel-ret:	363878
Macro average recall	0.292
Macro average precision	0.785
Mean T10SU	0.298
Mean F-Beta	0.498
Zero returns	2



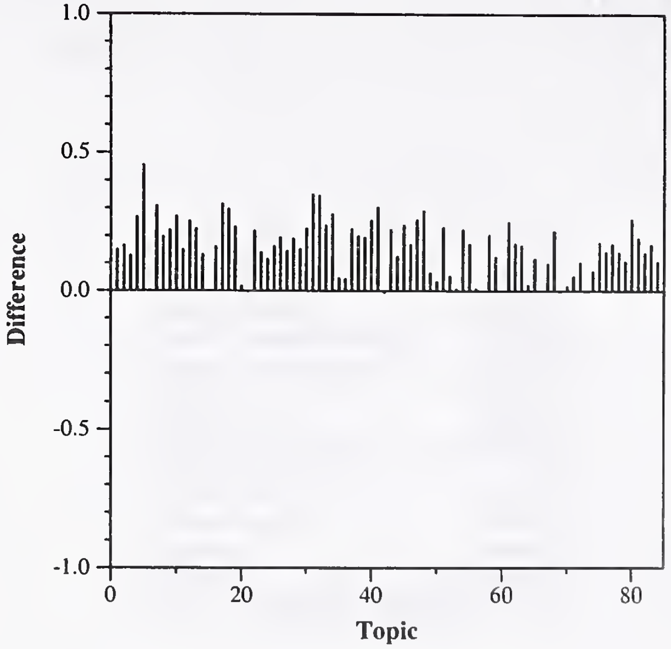
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	DLewis01bfFa
Subtask	batch
Resources used:	none
Optimized for	F-beta
Number of Topics:	84
Retrieved:	475984
Rel-ret:	413618
Macro average recall	0.388
Macro average precision	0.798
Mean T10SU	0.380
Mean F-Beta	0.606
Zero returns	2



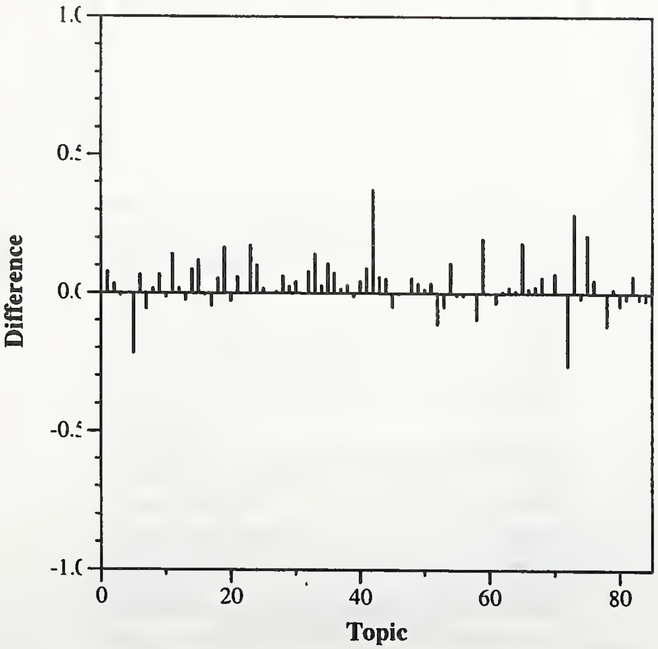
Difference from Median in F-beta per Topic

Summary Statistics	
Run ID:	DLewis01bfUa
Subtask	batch
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	643603
Rel-ret:	499047
Macro average recall	0.450
Macro average precision	0.749
Mean T10SU	0.414
Mean F-Beta	0.600
Zero returns	2



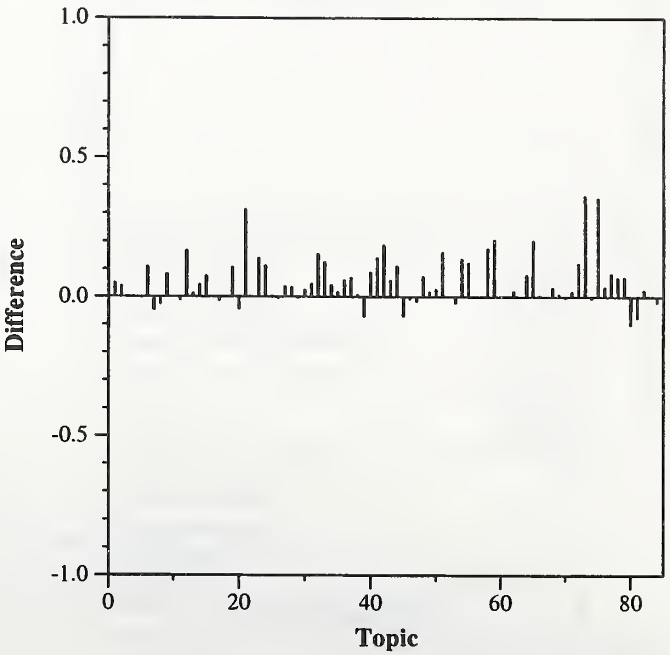
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	KUNbF
Subtask	batch
Resources used:	none
Optimized for	F-beta
Number of Topics:	84
Retrieved:	382199
Rel-ret:	289731
Macro average recall	0.293
Macro average precision	0.683
Mean T10SU	0.264
Mean F-Beta	0.489
Zero returns	3



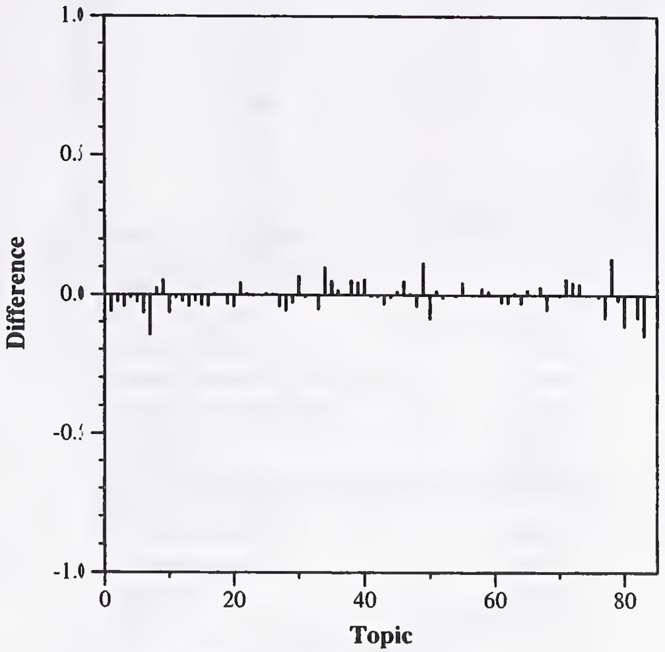
Difference from Median in F-beta per Topic

Summary Statistics	
Run ID:	KUNbU
Subtask	batch
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	635771
Rel-ret:	399503
Macro average recall	0.414
Macro average precision	0.571
Mean T10SU	0.307
Mean F-Beta	0.507
Zero returns	2



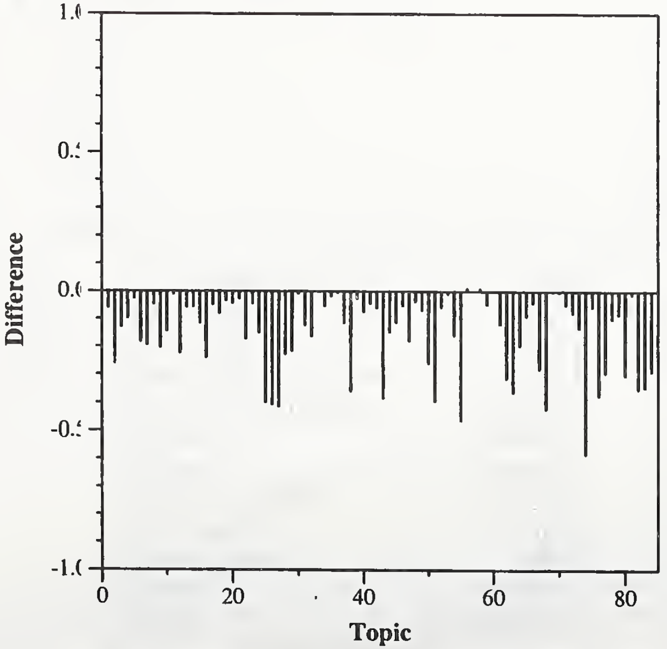
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	oraBU082701
Subtask	batch
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	739734
Rel-ret:	410323
Macro average recall	0.354
Macro average precision	0.556
Mean T10SU	0.249
Mean F-Beta	0.451
Zero returns	3



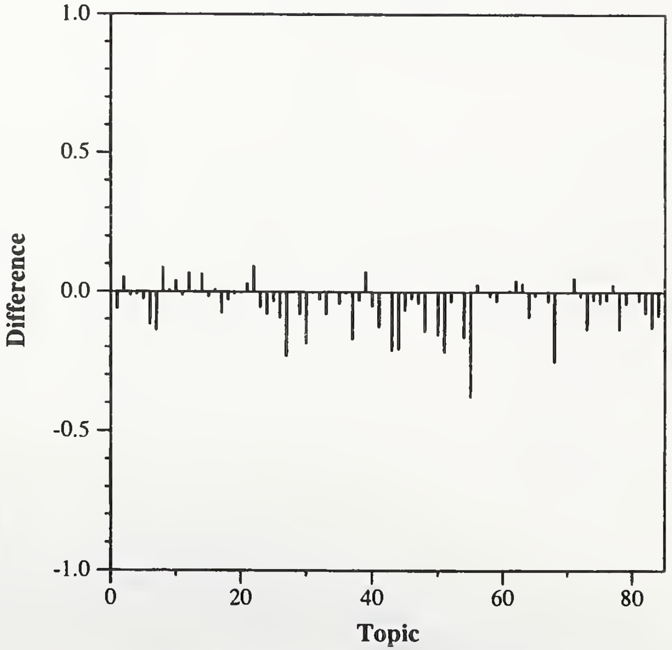
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	serASSAT10ba
Subtask	batch
Resources used:	none
Optimized for	neither
Number of Topics:	84
Retrieved:	362065
Rel-ret:	172714
Macro average recall	0.153
Macro average precision	0.388
Mean T10SU	0.106
Mean F-Beta	0.265
Zero returns	2



Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	serCLST10ba
Subtask	batch
Resources used:	none
Optimized for	neither
Number of Topics:	84
Retrieved:	629474
Rel-ret:	357706
Macro average recall	0.257
Macro average precision	0.634
Mean T10SU	0.209
Mean F-Beta	0.388
Zero returns	6



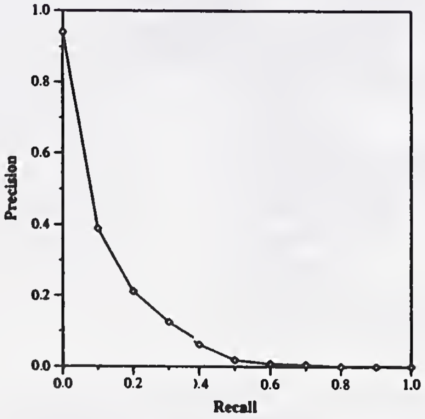
Difference from Median in T10SU per Topic

Summary Statistics	
Run ID:	apl10frn
Subtask	routing
Resources used:	none
Optimized for	neither
Number of Topics:	84

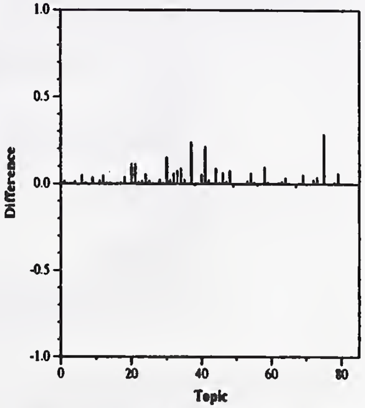
Retrieved:	84000
Rel-ret:	57653
Topics with P@1000 > 0.9	27

Re:all Level Averages	
Fecall	Precision
).00	0.9405
).10	0.3888
).20	0.2117
).30	0.1254
).40	0.0622
).50	0.0188
).60	0.0083
).70	0.0055
).80	0.0000
).90	0.0000
l.00	0.0000
unintrp. MAP	0.1210

Document Level Averages	
	Precision
At 5 docs	0.8524
At 10 docs	0.8333
At 15 docs	0.8167
At 20 docs	0.8083
At 30 docs	0.8020
At 100 docs	0.7752
At 200 docs	0.7643
At 500 docs	0.7300
At 1000 docs	0.6863
Exact RPrec	0.1561



Recal-Precision Curve



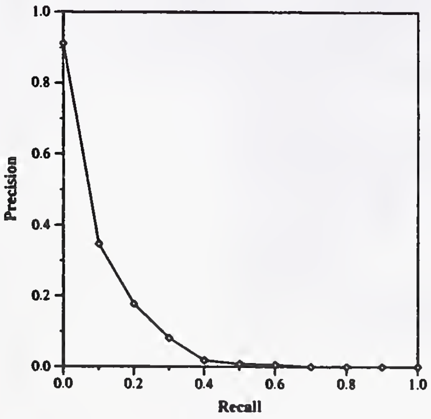
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	apl10frs
Subtask	routing
Resources used:	none
Optimized for	neither
Number of Topics:	84

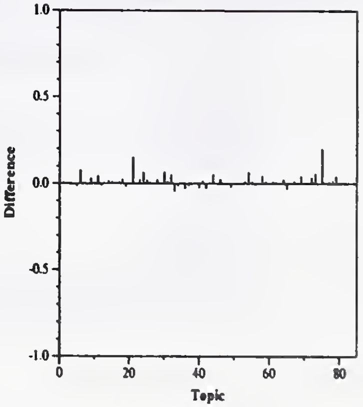
Retrieved:	84000
Rel-ret:	54932
Topics with P@1000 > 0.9	19

Recall Level Averages	
Recall	Precision
0.00	0.9127
0.10	0.3470
0.20	0.1771
0.30	0.0813
0.40	0.0184
0.50	0.0085
0.60	0.0055
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
uninterp. MAP	0.1038

Document Level Averages	
	Precision
At 5 docs	0.7976
At 10 docs	0.7619
At 15 docs	0.7611
At 20 docs	0.7595
At 30 docs	0.7536
At 100 docs	0.7365
At 200 docs	0.7288
At 500 docs	0.6969
At 1000 docs	0.6540
Exact RPrec	0.1410



Recall-Precision Curve



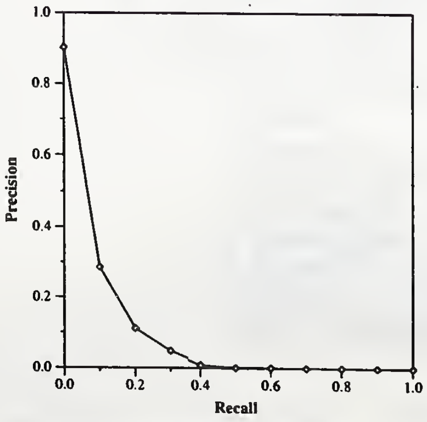
Difference from Median in Average Precision per Topic

Filtering - routing results — Clairvoyance Corporation

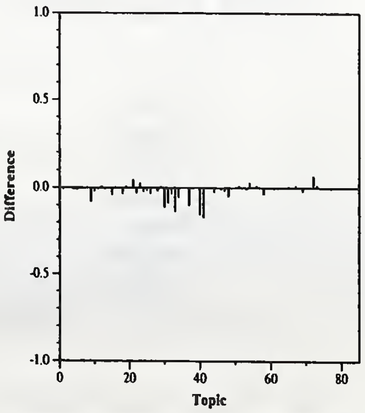
Summary Statistics	
Run ID:	clT10rta
Subtask	routing
Resources used:	none
Optimized for	neither
Number of Topics:	84
Retrieved:	84000
Rel-ret:	48109
Topics with P@1000 > 0.9	14

Recall Level Averages	
Recall	Precision
0.00	0.9028
0.10	0.2853
0.20	0.1110
0.30	0.0482
0.40	0.0084
0.50	0.0011
0.60	0.0010
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
uninterp. MAP	0.0778

Document Level Averages	
	Precision
At 5 docs	0.7190
At 10 docs	0.7190
At 15 docs	0.7254
At 20 docs	0.7113
At 30 docs	0.7091
At 100 docs	0.6832
At 200 docs	0.6600
At 500 docs	0.6213
At 1000 docs	0.5727
Exact RPrec	0.1170



Recal-Precision Curve

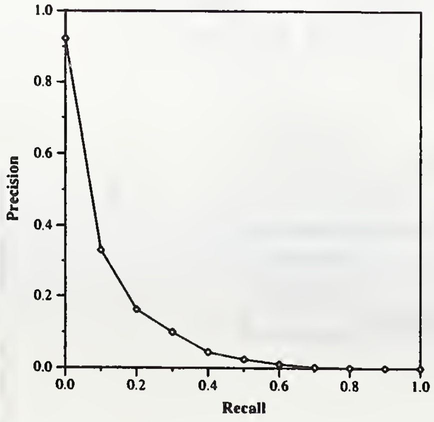


Difference from Median in Average Precision per Topic

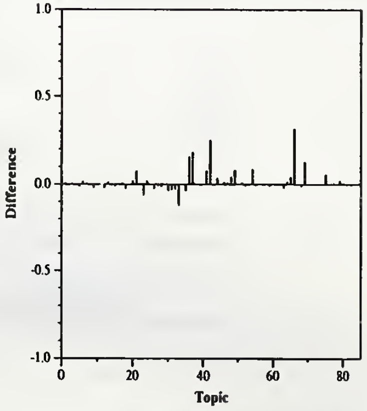
Summary Statistics	
Run ID:	clT10rtb
Subtask	routing
Resources used:	none
Optimized for	neither
Number of Topics:	84
Retrieved:	84000
Rel-ret:	52135
Topics with P@1000 > 0.9	13

Recall Level Averages	
Recall	Precision
0.00	0.9226
0.10	0.3303
0.20	0.1618
0.30	0.0995
0.40	0.0446
0.50	0.0249
0.60	0.0114
0.70	0.0028
0.80	0.0013
0.90	0.0000
1.00	0.0000
uninterp. MAP	0.1048

Document Level Averages	
	Precision
At 5 docs	0.7690
At 10 docs	0.7726
At 15 docs	0.7667
At 20 docs	0.7649
At 30 docs	0.7647
At 100 docs	0.7404
At 200 docs	0.7198
At 500 docs	0.6683
At 1000 docs	0.6207
Exact RPrec	0.1444



Recall-Precision Curve



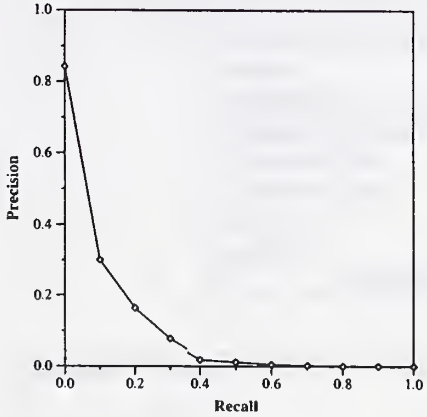
Difference from Median in Average Precision per Topic

Filtering - routing results — IRIT/SIG

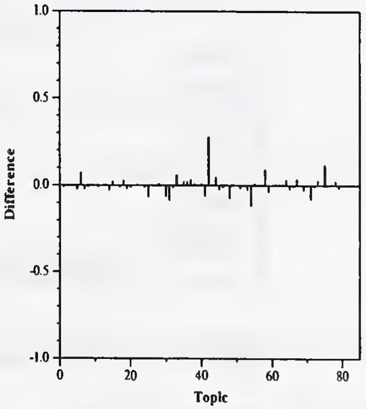
Summary Statistics	
Run ID:	mer10r1
Subtask	routing
Resources used:	other TREC
Optimized for	neither
Number of Topics:	84
Retrieved:	84000
Rel-ret:	46986
Topics with P@1000 > 0.9	15

Recall Level Averages	
Recall	Precision
0.00	0.8429
0.10	0.3000
0.20	0.1637
0.30	0.0780
0.40	0.0182
0.50	0.0121
0.60	0.0051
0.70	0.0022
0.80	0.0000
0.90	0.0000
1.00	0.0000
uninterp. MAP	0.0920

Document Level Averages	
	Precision
At 5 docs	0.7476
At 10 docs	0.7274
At 15 docs	0.7214
At 20 docs	0.7244
At 30 docs	0.7179
At 100 docs	0.6854
At 200 docs	0.6587
At 500 docs	0.6084
At 1000 docs	0.5594
Exact RPrec	0.1263



Recal-Precision Curve



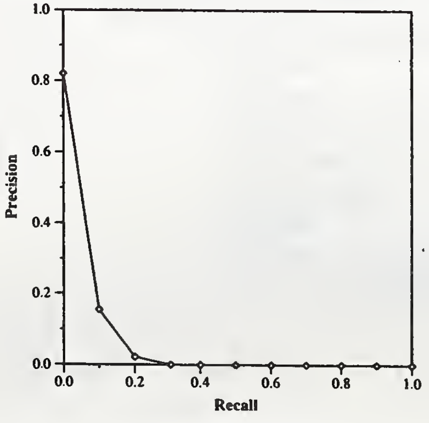
Difference from Median in Average Precision per Topic

Filtering - routing results — Justsystem Corporation

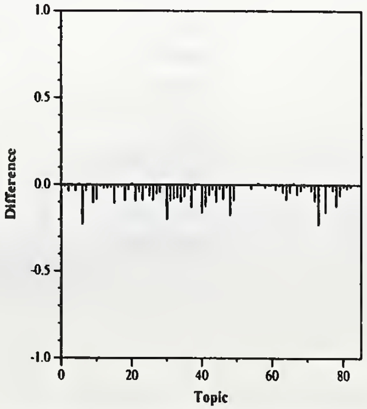
Summary Statistics	
Run ID:	jscbtafr1
Subtask	routing
Resources used:	other TREC
Optimized for	neither
Number of Topics:	84
Retrieved:	84000
Rel-ret:	32692
Topics with P@1000 > 0.9	7

Recall Level Averages	
Recall	Precision
0.00	0.8204
0.10	0.1539
0.20	0.0217
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
uninterp. MAP	0.0420

Document Level Averages	
	Precision
At 5 docs	0.6619
At 10 docs	0.6548
At 15 docs	0.6357
At 20 docs	0.6274
At 30 docs	0.6139
At 100 docs	0.5536
At 200 docs	0.5157
At 500 docs	0.4458
At 1000 docs	0.3892
Exact RPrec	0.0765



Recall-Precision Curve

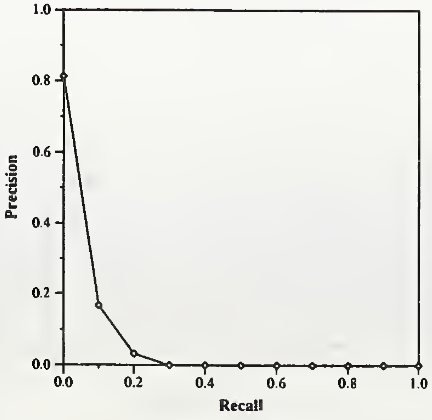


Difference from Median in Average Precision per Topic

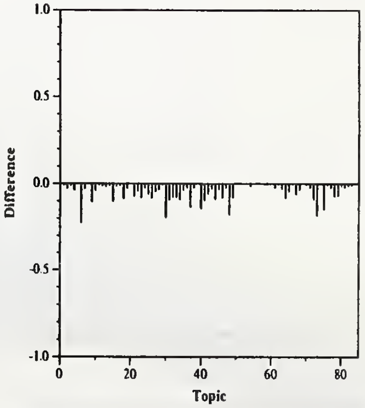
Summary Statistics	
Run ID:	jscbtafr2
Subtask	routing
Resources used:	other TREC
Optimized for	neither
Number of Topics:	84
Retrieved:	84000
Rel-ret:	34710
Topics with P@1000 > 0.9	6

Recall Level Averages	
Recall	Precision
0.00	0.8134
0.10	0.1666
0.20	0.0320
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
uninterp. MAP	0.0466

Document Level Averages	
	Precision
At 5 docs	0.6286
At 10 docs	0.6095
At 15 docs	0.5976
At 20 docs	0.5911
At 30 docs	0.5853
At 100 docs	0.5598
At 200 docs	0.5271
At 500 docs	0.4620
At 1000 docs	0.4132
Exact RPrec	0.0831



Recall-Precision Curve

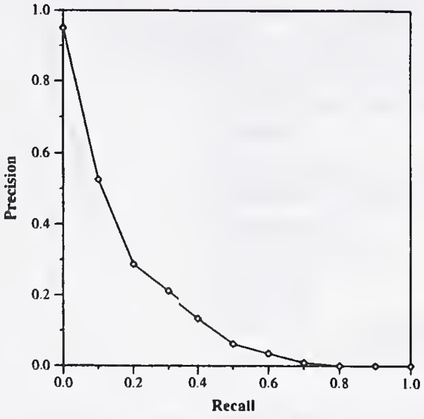


Difference from Median in Average Precision per Topic

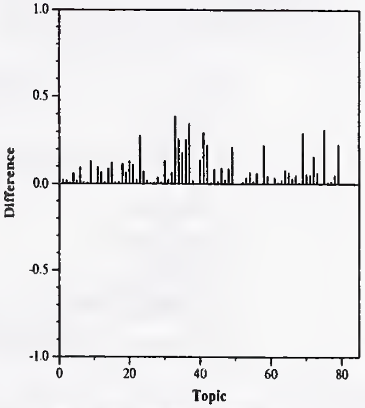
Summary Statistics	
Run ID:	DLewis01rFa
Subtask	routing
Resources used:	none
Optimized for	F-beta
Number of Topics:	84
Retrieved:	84000
Rel-ret:	66285
Topics with P@1000 > 0.9	46

Recall Level Averages	
Recall	Precision
0.00	0.9501
0.10	0.5262
0.20	0.2872
0.30	0.2115
0.40	0.1337
0.50	0.0623
0.60	0.0346
0.70	0.0093
0.80	0.0000
0.90	0.0000
1.00	0.0000
uninterp. MAP	0.1682

Document Level Averages	
	Precision
At 5 docs	0.8786
At 10 docs	0.8750
At 15 docs	0.8825
At 20 docs	0.8804
At 30 docs	0.8790
At 100 docs	0.8632
At 200 docs	0.8483
At 500 docs	0.8261
At 1000 docs	0.7891
Exact RPrec	0.1985



Recall-Precision Curve

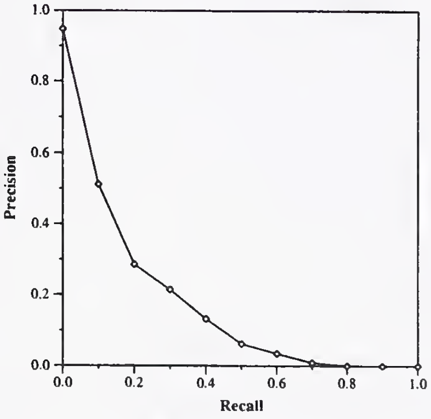


Difference from Median in Average Precision per Topic

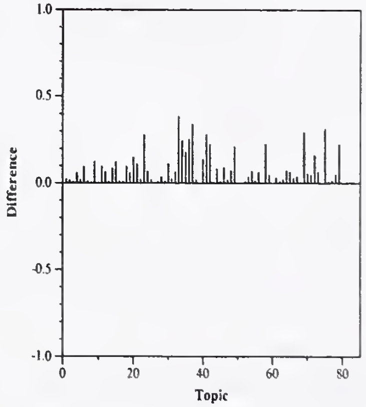
Summary Statistics	
Run ID:	DLewis01rUa
Subtask	routing
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	84000
Rel-ret:	65645
Topics with P@1000 > 0.9	40

Recall Level Averages	
Recall	Precision
0.00	0.9477
0.10	0.5116
0.20	0.2851
0.30	0.2141
0.40	0.1324
0.50	0.0621
0.60	0.0347
0.70	0.0093
0.80	0.0000
0.90	0.0000
1.00	0.0000
uninterp. MAP	0.1663

Document Level Averages	
	Precision
At 5 docs	0.8714
At 10 docs	0.8714
At 15 docs	0.8770
At 20 docs	0.8768
At 30 docs	0.8734
At 100 docs	0.8587
At 200 docs	0.8440
At 500 docs	0.8199
At 1000 docs	0.7815
Exact RPrec	0.1974



Recall-Precision Curve

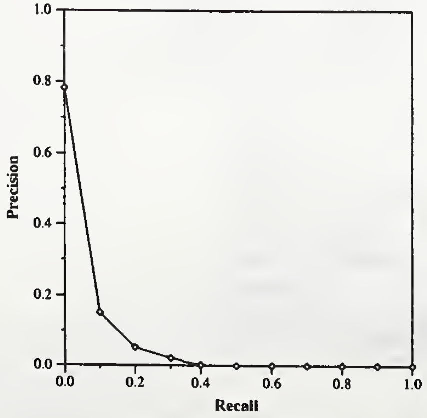


Difference from Median in Average Precision per Topic

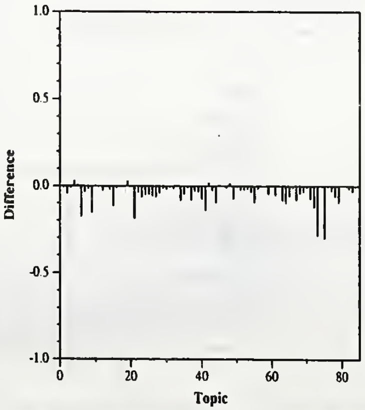
Summary Statistics	
Run ID:	MMAT10rout
Subtask	routing
Resources used:	other data
Optimized for	neither
Number of Topics:	84
Retrieved:	66050
Rel-ret:	33700
Topics with P@1000 > 0.9	6

R:call Level Averages	
Recall	Precision
0.00	0.7833
0.10	0.1512
0.20	0.0515
0.30	0.0211
0.40	0.0016
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
unin:erp. MAP	0.0451

Document Level Averages	
	Precision
At 5 docs	0.5643
At 10 docs	0.5476
At 15 docs	0.5302
At 20 docs	0.5179
At 30 docs	0.5187
At 100 docs	0.4732
At 200 docs	0.4532
At 500 docs	0.4154
At 1000 docs	0.4012
Exact RPrec	0.0729



Rec:ll-Precision Curve

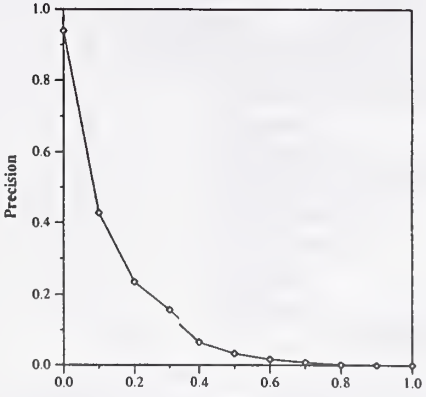


Difference from Median in Average Precision per Topic

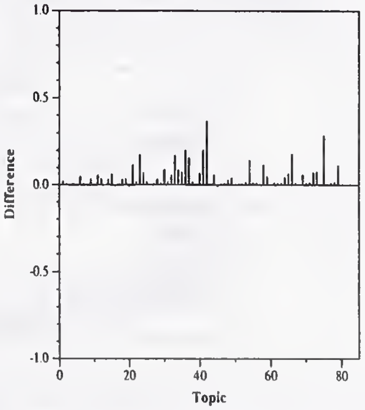
Summary Statistics	
Run ID:	KUNr1
Subtask	routing
Resources used:	none
Optimized for	neither
Number of Topics:	84
Retrieved:	84000
Rel-ret:	58915
Topics with P@1000 > 0.9	27

R:call Level Averages	
Recall	Precision
0.00	0.9393
0.10	0.4277
0.20	0.2354
0.30	0.1573
0.40	0.0652
0.50	0.0333
0.60	0.0168
0.70	0.0087
0.80	0.0012
0.90	0.0000
1.00	0.0000
unin:erp. MAP	0.1356

Document Level Averages	
	Precision
At 5 docs	0.8143
At 10 docs	0.8095
At 15 docs	0.8135
At 20 docs	0.8208
At 30 docs	0.8131
At 100 docs	0.7943
At 200 docs	0.7839
At 500 docs	0.7506
At 1000 docs	0.7014
Exact RPrec	0.1711



Reall-Precision Curve

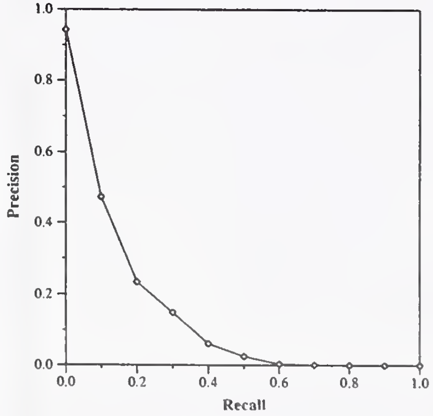


Difference from Median in Average Precision per Topic

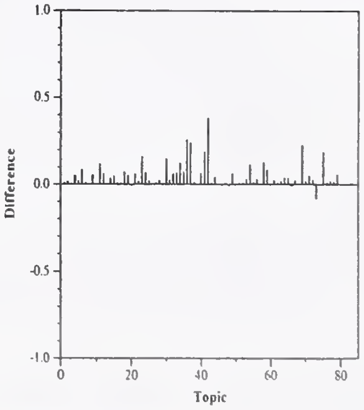
Summary Statistics	
Run ID:	KUNr2
Subtask	routing
Resources used:	none
Optimized for	neither
Number of Topics:	84
Retrieved:	84000
Rel-ret:	60564
Topics with P@1000 > 0.9	30

Recall Level Averages	
Recall	Precision
0.00	0.9429
0.10	0.4723
0.20	0.2346
0.30	0.1497
0.40	0.0610
0.50	0.0252
0.60	0.0037
0.70	0.0011
0.80	0.0000
0.90	0.0000
1.00	0.0000
uninterp. MAP	0.1369

Document Level Averages	
	Precision
At 5 docs	0.8881
At 10 docs	0.8786
At 15 docs	0.8714
At 20 docs	0.8673
At 30 docs	0.8627
At 100 docs	0.8367
At 200 docs	0.8184
At 500 docs	0.7724
At 1000 docs	0.7210
Exact RPrec	0.1662



Recall-Precision Curve



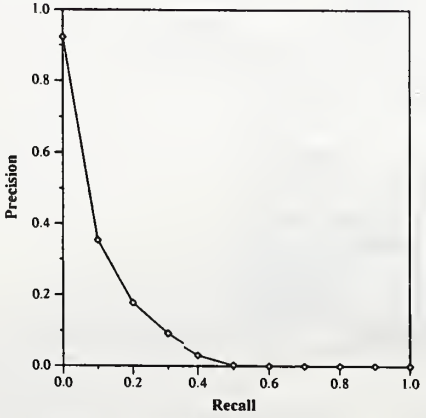
Difference from Median in Average Precision per Topic

Filtering - routing results — Oracle

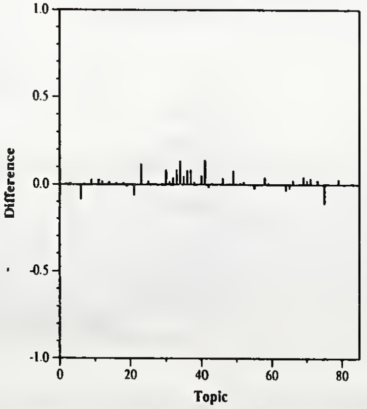
Summary Statistics	
Run ID:	oraRO082801
Subtask	routing
Resources used:	none
Optimized for	neither
Number of Topics:	84
Retrieved:	84000
Rel-ret:	54775
Topics with P@1000 > 0.9	21

Recall Level Averages	
Recall	Precision
0.00	0.9226
0.10	0.3532
0.20	0.1778
0.30	0.0921
0.40	0.0306
0.50	0.0026
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
uninterp. MAP	0.1040

Document Level Averages	
	Precision
At 5 docs	0.8000
At 10 docs	0.7857
At 15 docs	0.7722
At 20 docs	0.7720
At 30 docs	0.7694
At 100 docs	0.7405
At 200 docs	0.7267
At 500 docs	0.6968
At 1000 docs	0.6521
Exact RPrec	0.1431



Recall-Precision Curve

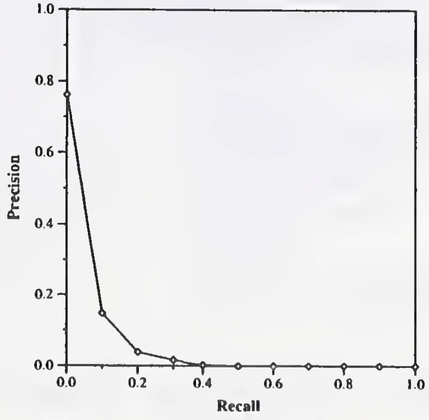


Difference from Median in Average Precision per Topic

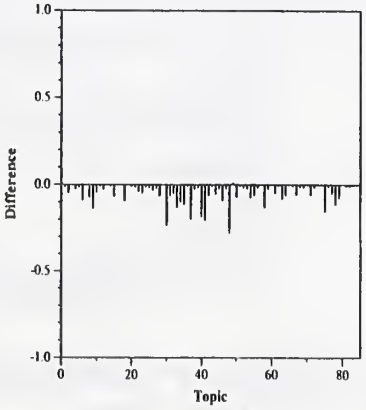
Summary Statistics	
Run ID:	RUGR01
Subtask	routing
Resources used:	none
Optimized for	T10U
Number of Topics:	84
Retrieved:	83038
Rel-ret:	37064
Topics with P@1000 > 0.9	4

Recall Level Averages	
Recall	Precision
0.00	0.7619
0.10	0.1472
0.20	0.0387
0.30	0.0165
0.40	0.0024
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
uninerp. MAP	0.0445

Document Level Averages	
	Precision
At 5 docs	0.5238
At 10 docs	0.5369
At 15 docs	0.5397
At 20 docs	0.5405
At 30 docs	0.5377
At 100 docs	0.5136
At 200 docs	0.4973
At 500 docs	0.4720
At 1000 docs	0.4412
Exact RPrec	0.0820



Recall-Precision Curve

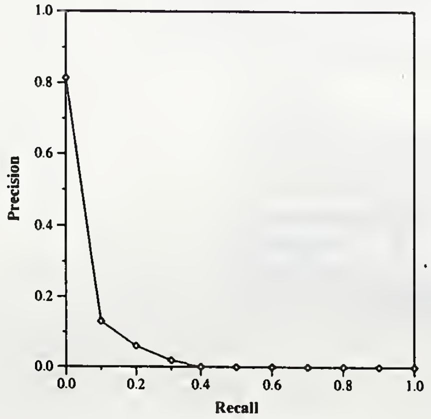


Difference from Median in Average Precision per Topic

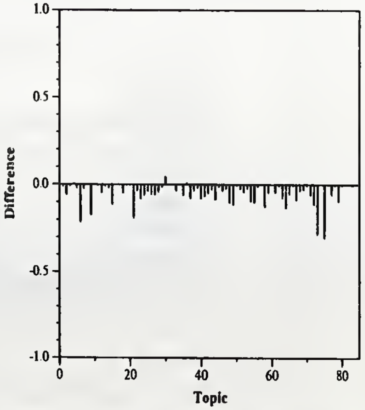
Summary Statistics	
Run ID:	serASSAT10ro
Subtask	routing
Resources used:	none
Optimized for	neither
Number of Topics:	84
Retrieved:	60634
Rel-ret:	31375
Topics with P@1000 > 0.9	7

Recall Level Averages	
Recall	Precision
0.00	0.8135
0.10	0.1300
0.20	0.0594
0.30	0.0181
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
uninterp. MAP	0.0440

Document Level Averages	
	Precision
At 5 docs	0.5561
At 10 docs	0.5500
At 15 docs	0.5333
At 20 docs	0.5341
At 30 docs	0.5256
At 100 docs	0.5054
At 200 docs	0.4820
At 500 docs	0.4342
At 1000 docs	0.3826
Exact RPrec	0.0649



Recall-Precision Curve

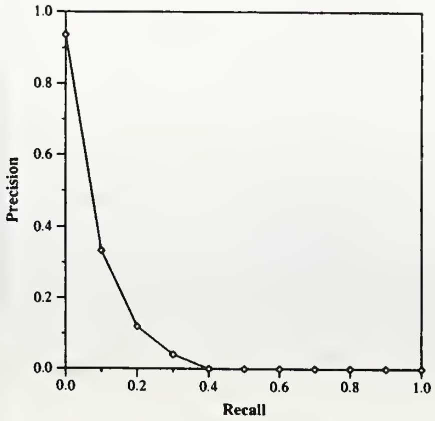


Difference from Median in Average Precision per Topic

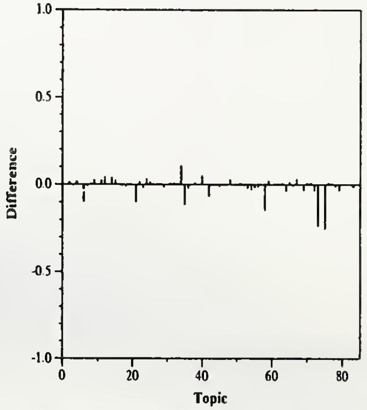
Summary Statistics	
Run ID:	serCLST10ro
Subtask	routing
Resources used:	none
Optimized for	neither
Number of Topics:	84
Retrieved:	66989
Rel-ret:	49796
Topics with P@1000 > 0.9	23

Recall Level Averages	
Recall	Precision
0.00	0.9365
0.10	0.3330
0.20	0.1193
0.30	0.0396
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
uninterp. MAP	0.0869

Document Level Averages	
	Precision
At 5 docs	0.8076
At 10 docs	0.7873
At 15 docs	0.7873
At 20 docs	0.7810
At 30 docs	0.7743
At 100 docs	0.7557
At 200 docs	0.7362
At 500 docs	0.6838
At 1000 docs	0.6303
Exact RPrec	0.1107



Recall-Precision Curve

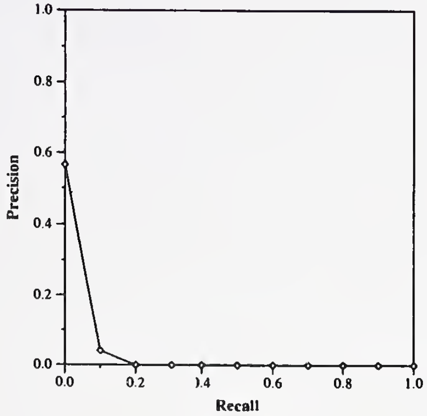


Difference from Median in Average Precision per Topic

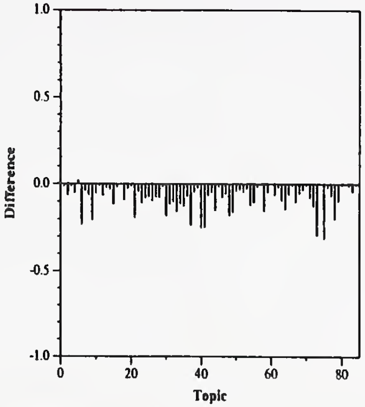
Summary Statistics	
Run ID:	VisaSent1T10
Subtask	routing
Resources used:	none
Optimized for	neither
Number of Topics:	84
Retrieved:	84000
Rel-ret:	13432
Topics with P@1000 > 0.9	3

Recall Level Averages	
Recall	Precision
).00	0.5657
).10	0.0398
).20	0.0000
).30	0.0000
).40	0.0000
).50	0.0000
).60	0.0000
).70	0.0000
).80	0.0000
).90	0.0000
l.00	0.0000
uninterp. MAP	0.0118

Document Level Averages	
	Precision
At 5 docs	0.4119
At 10 docs	0.3774
At 15 docs	0.3571
At 20 docs	0.3417
At 30 docs	0.3163
At 100 docs	0.2669
At 200 docs	0.2330
At 500 docs	0.1931
At 1000 docs	0.1599
Exact RPrec	0.0174



Recal-Precision Curve

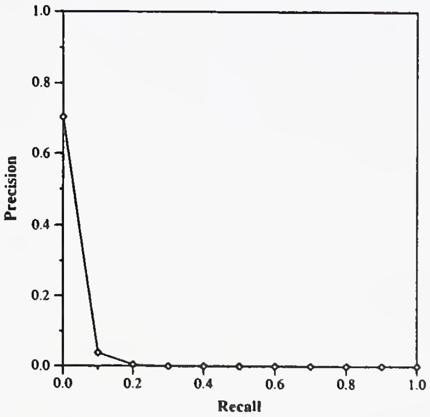


Difference from Median in Average Precision per Topic

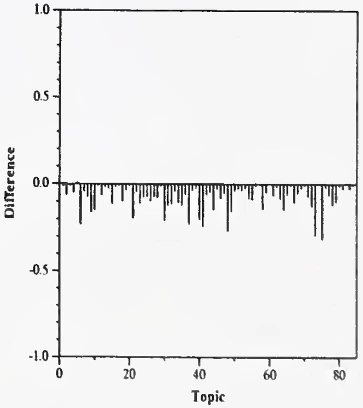
Summary Statistics	
Run ID:	VisaWordT10
Subtask	routing
Resources used:	none
Optimized for	neither
Number of Topics:	84
Retrieved:	82945
Rel-ret:	17048
Topics with P@1000 > 0.9	1

Recall Level Averages	
Recall	Precision
0.00	0.7033
0.10	0.0376
0.20	0.0042
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
uninterp. MAP	0.0144

Document Level Averages	
	Precision
At 5 docs	0.4762
At 10 docs	0.4333
At 15 docs	0.4135
At 20 docs	0.4054
At 30 docs	0.3877
At 100 docs	0.3239
At 200 docs	0.2827
At 500 docs	0.2332
At 1000 docs	0.2030
Exact RPrec	0.0338



Recall-Precision Curve



Difference from Median in Average Precision per Topic

clr01c1	
Question	Rank
1a	1
1b	Not found
1c	Not found
1d	1
1e	Not found
1f	1
2a	Not found
2b	3
2c	Not found
3a	Not found
3b	Not found
3c	Not found
3d	Not found
4a	Not found
4b	Not found
4c	Not found
5a	Not found
5b	Not found
5c	Not found
6a	Not found
6b	Not found
6c	Not found
7a	5
7b	Not found
7c	Not found
7d	Not found
7e	5
7f	Not found
7g	Not found
7h	1
7i	1
8a	Not found
8b	4
8c	Not found
8d	Not found
9a	Not found
9b	Not found
9c	Not found
9d	Not found
10a	1
10b	Not found
10c	2

KAISTQACTX	
Question	Rank
1a	Not found
1b	Not found
1c	Not found
1d	Not found
1e	Not found
1f	Not found
2a	Not found
2b	Not found
2c	Not found
3a	Not found
3b	Not found
3c	Not found
3d	Not found
4a	Not found
4b	Not found
4c	Not found
5a	Not found
5b	Not found
5c	Not found
6a	Not found
6b	Not found
6c	1
7a	3
7b	Not found
7c	2
7d	Not found
7e	Not found
7f	Not found
7g	Not found
7h	1
7i	3
8a	Not found
8b	1
8c	Not found
8d	Not found
9a	Not found
9b	Not found
9c	2
9d	Not found
10a	Not found
10b	Not found
10c	Not found

LCC3	
Question	Rank
1a	1
1b	1
1c	Not found
1d	1
1e	1
1f	Not found
2a	Not found
2b	3
2c	1
3a	Not found
3b	1
3c	1
3d	Not found
4a	1
4b	1
4c	1
5a	1
5b	1
5c	1
6a	Not found
6b	1
6c	1
7a	1
7b	Not found
7c	1
7d	1
7e	1
7f	1
7g	1
7h	1
7i	1
8a	Not found
8b	1
8c	1
8d	1
9a	2
9b	1
9c	1
9d	1
10a	1
10b	1
10c	1

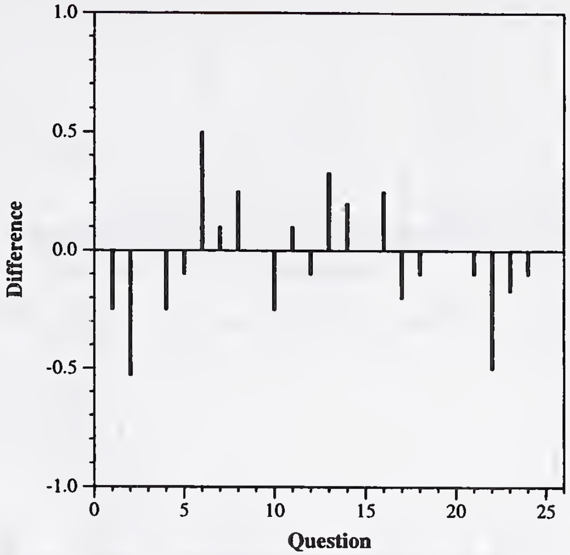
qntuacl	
Question	Rank
1a	2
1b	Not found
1c	Not found
1d	Not found
1e	Not found
1f	3
2a	Not found
2b	Not found
2c	Not found
3a	Not found
3b	3
3c	Not found
3d	Not found
4a	Not found
4b	Not found
4c	Not found
5a	Not found
5b	Not found
5c	Not found
6a	4
6b	1
6c	4
7a	2
7b	Not found
7c	Not found
7d	Not found
7e	Not found
7f	Not found
7g	3
7h	Not found
7i	1
8a	Not found
8b	Not found
8c	Not found
8d	Not found
9a	Not found
9b	Not found
9c	Not found
9d	Not found
10a	1
10b	3
10c	Not found

pir1Qctx2	
Question	Rank
1a	1
1b	Not found
1c	Not found
1d	1
1e	Not found
1f	5
2a	1
2b	Not found
2c	Not found
3a	Not found
3b	Not found
3c	Not found
3d	Not found
4a	Not found
4b	Not found
4c	1
5a	1
5b	2
5c	Not found
6a	4
6b	Not found
6c	Not found
7a	Not found
7b	Not found
7c	1
7d	Not found
7e	Not found
7f	Not found
7g	Not found
7h	1
7i	1
8a	1
8b	1
8c	Not found
8d	4
9a	Not found
9b	Not found
9c	Not found
9d	Not found
10a	1
10b	1
10c	Not found

pir1Qctx3	
Question	Rank
1a	1
1b	Not found
1c	3
1d	1
1e	Not found
1f	5
2a	1
2b	Not found
2c	Not found
3a	Not found
3b	Not found
3c	Not found
3d	Not found
4a	Not found
4b	Not found
4c	1
5a	1
5b	2
5c	Not found
6a	3
6b	Not found
6c	5
7a	Not found
7b	Not found
7c	1
7d	Not found
7e	Not found
7f	Not found
7g	Not found
7h	1
7i	1
8a	1
8b	1
8c	Not found
8d	4
9a	Not found
9b	Not found
9c	Not found
9d	Not found
10a	1
10b	1
10c	Not found

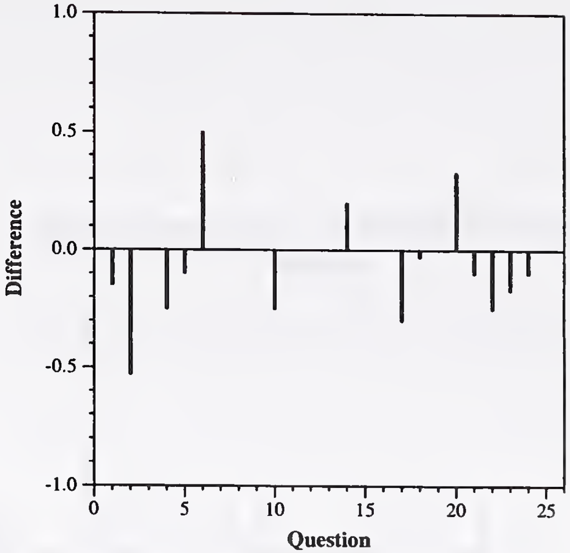
uwmtac0	
Question	Rank
1a	1
1b	Not found
1c	Not found
1d	Not found
1e	Not found
1f	Not found
2a	Not found
2b	Not found
2c	Not found
3a	Not found
3b	2
3c	Not found
3d	Not found
4a	4
4b	Not found
4c	Not found
5a	1
5b	Not found
5c	Not found
6a	1
6b	Not found
6c	Not found
7a	1
7b	Not found
7c	Not found
7d	Not found
7e	Not found
7f	Not found
7g	Not found
7h	Not found
7i	1
8a	5
8b	1
8c	Not found
8d	2
9a	Not found
9b	1
9c	Not found
9d	Not found
10a	1
10b	2
10c	Not found

Summary Statistics	
Run ID	clr0111
Num questions	25
Average accuracy	0.13



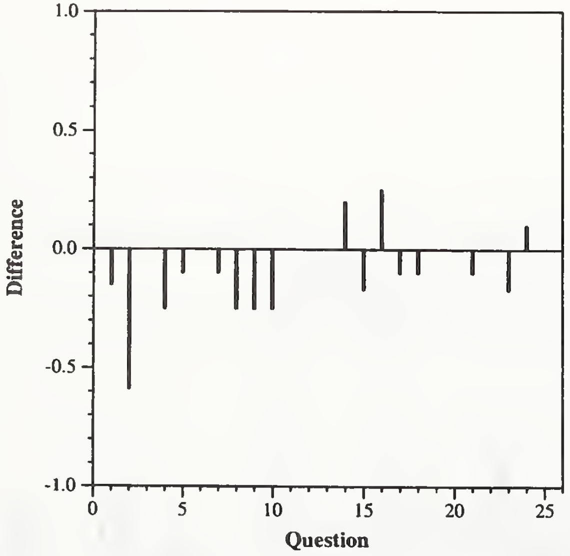
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	clr0112
Num questions	25
Average accuracy	0.12



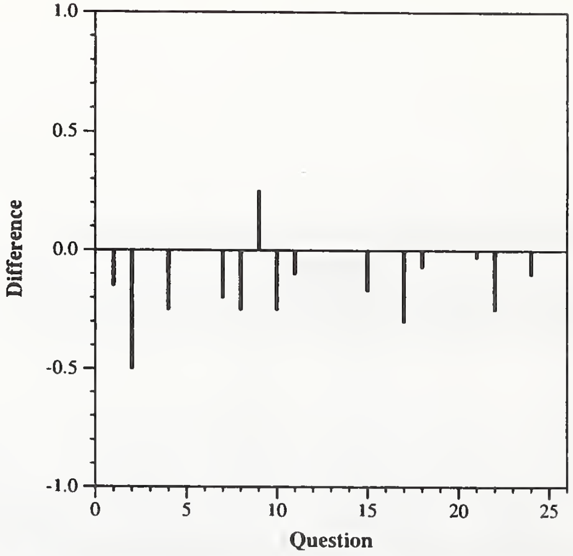
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	KAISTQALIST1
Num questions	25
Average accuracy	0.08



Difference from Median in average accuracy per question

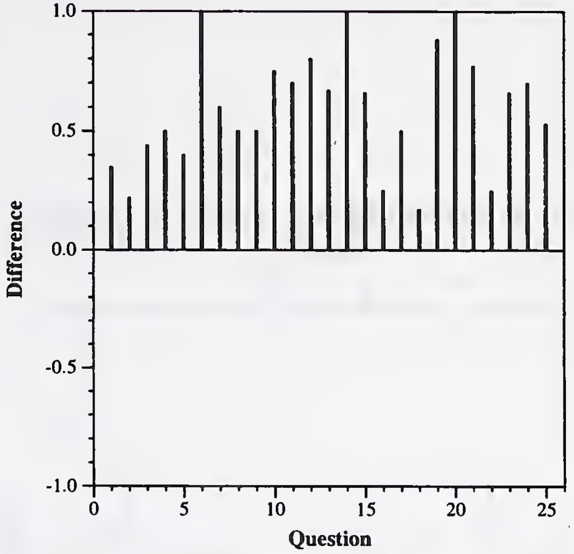
Summary Statistics	
Run ID	KAISTQALIST2
Num questions	25
Average accuracy	0.07



Difference from Median in average accuracy per question

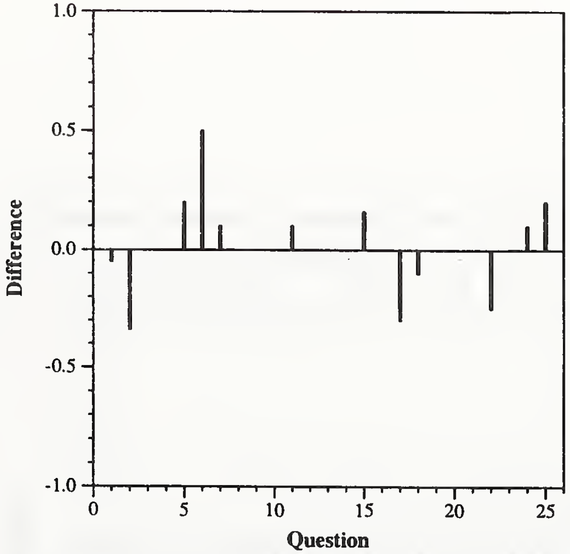
Question Answering track, List task results — Language Computer Corporation (LCC)

Summary Statistics	
Run ID	LCC2
Num questions	25
Average accuracy	0.76



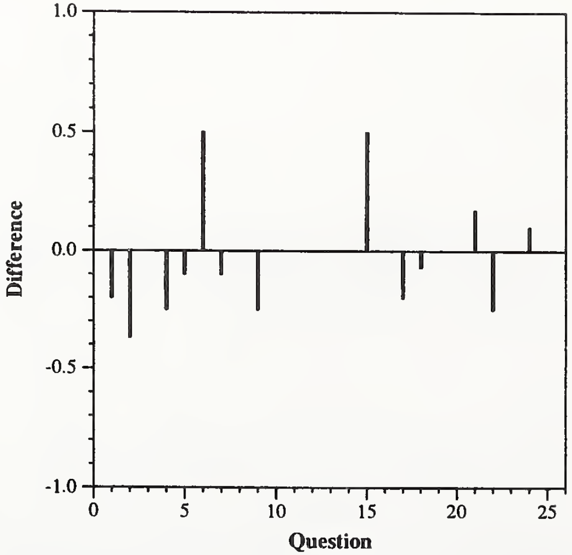
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	qntual1
Num questions	25
Average accuracy	0.18



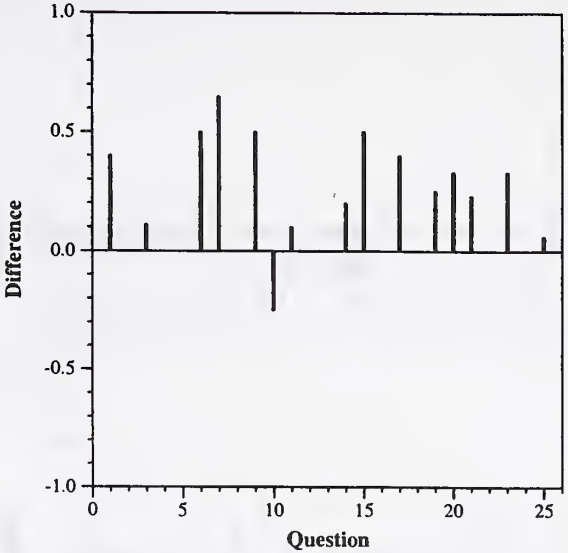
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	qntual2
Num questions	25
Average accuracy	0.14



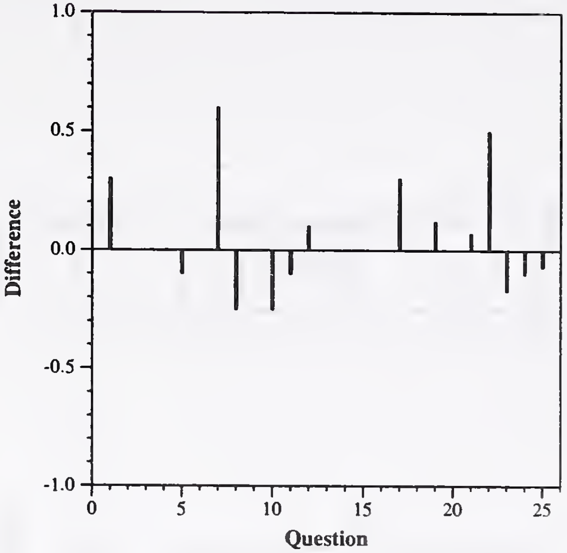
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	pir1Qli1
Num questions	25
Average accuracy	0.34



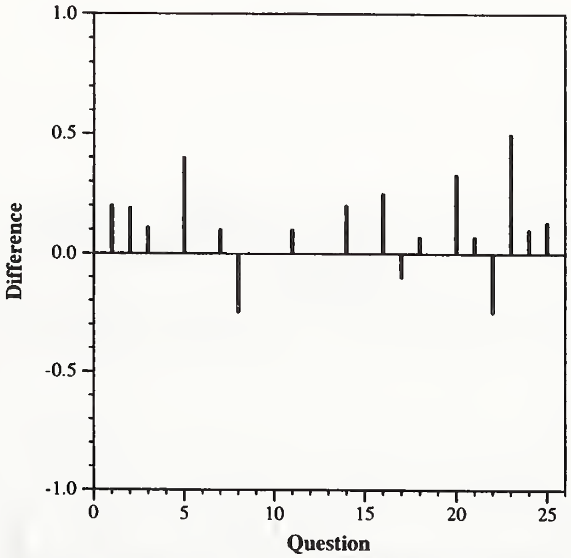
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	pir1Qli2
Num questions	25
Average accuracy	0.20



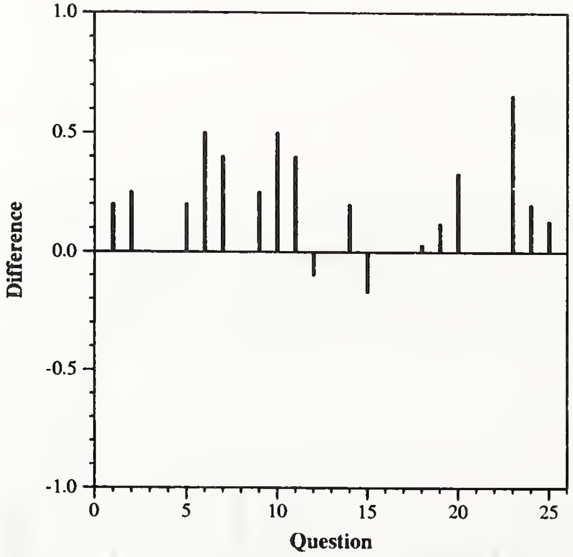
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	SUT10DOCLT
Num questions	25
Average accuracy	0.25



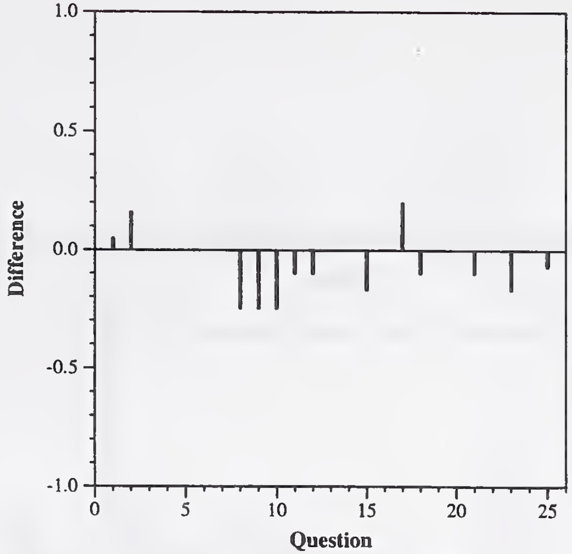
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	SUT10PARLT
Num questions	25
Average accuracy	0.33



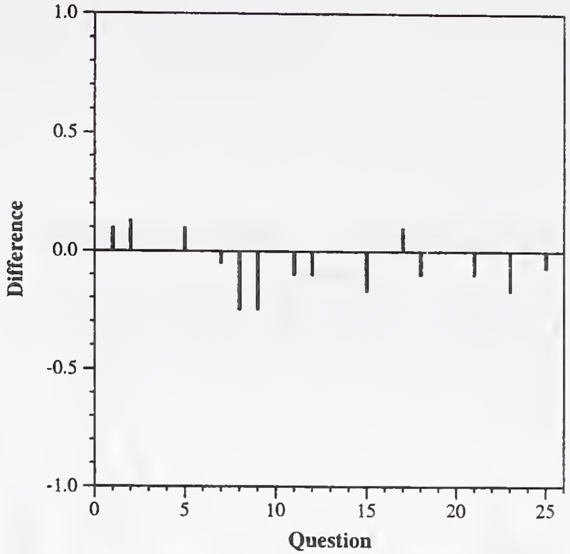
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	UAmsT10qaL1
Num questions	25
Average accuracy	0.12



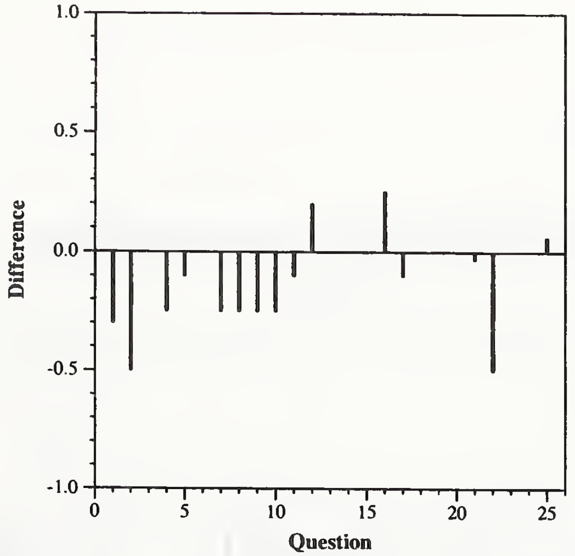
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	UAmsT10qaL2
Num questions	25
Average accuracy	0.13



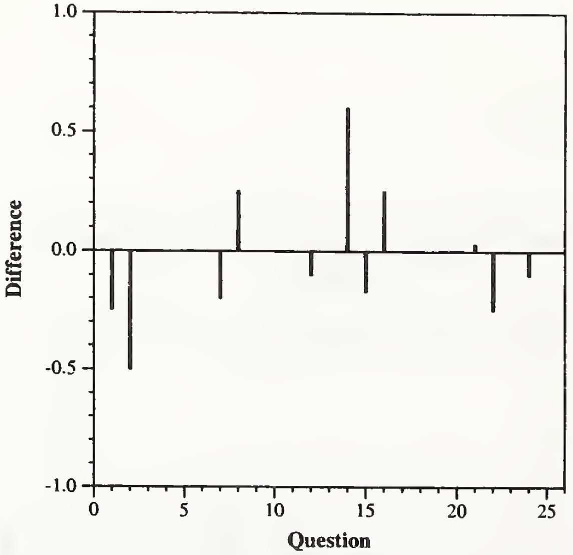
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	UdeMlistB
Num questions	25
Average accuracy	0.07



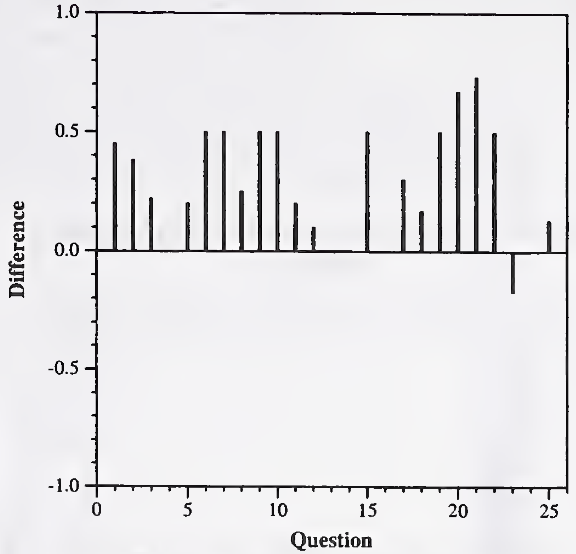
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	UdeMlistP
Num questions	25
Average accuracy	0.15



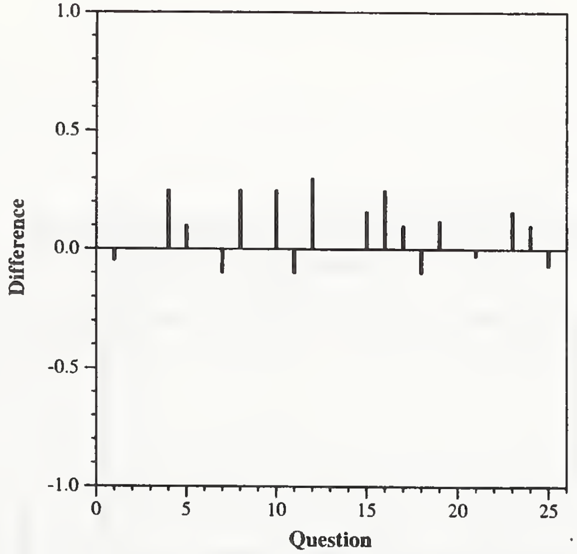
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	isi1l50
Num questions	25
Average accuracy	0.45



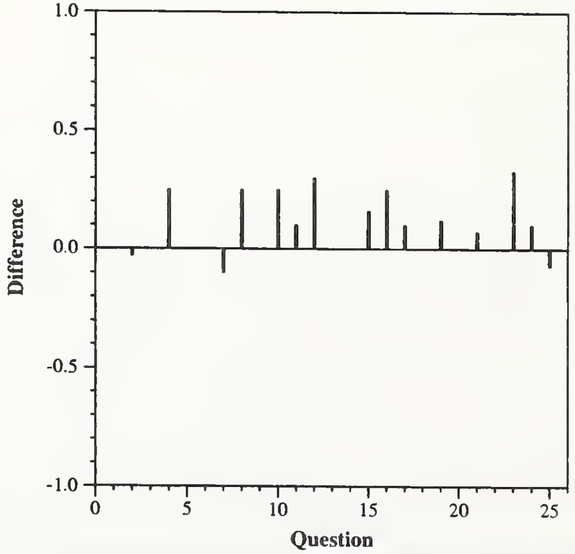
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	uwmtal0
Num questions	25
Average accuracy	0.23



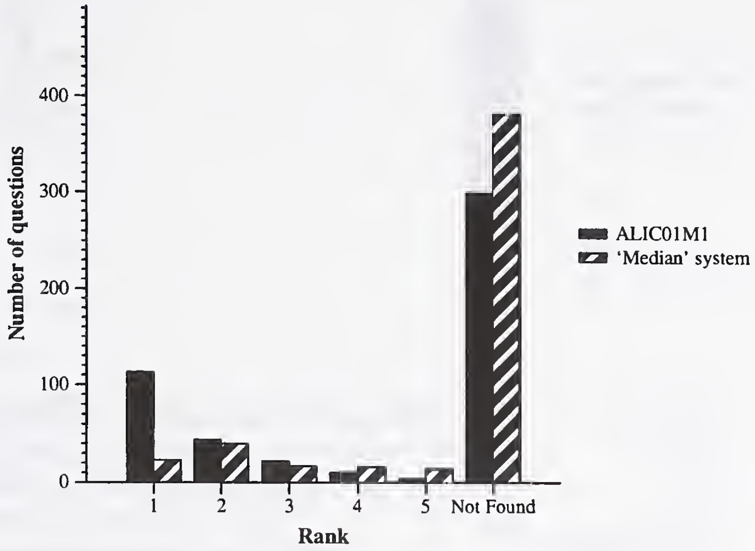
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	uwmtal1
Num questions	25
Average accuracy	0.25



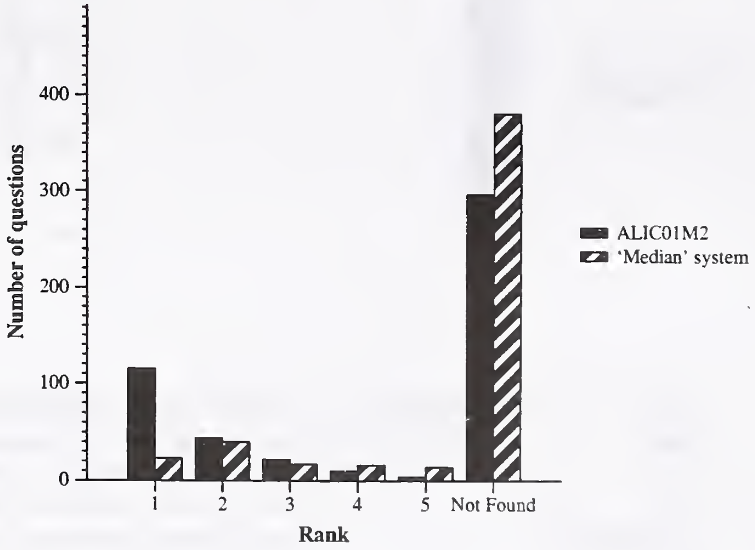
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	ALIC01M1
Num questions	492
Mean reciprocal rank (strict)	0.296
Mean reciprocal rank (lenient)	0.302
Num answers not found (strict)	299 (60.8%)
Num answers not found (lenient)	295 (60.0%)
Number of times NIL returned	4
Number of times NIL correctly returned	0
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	22%



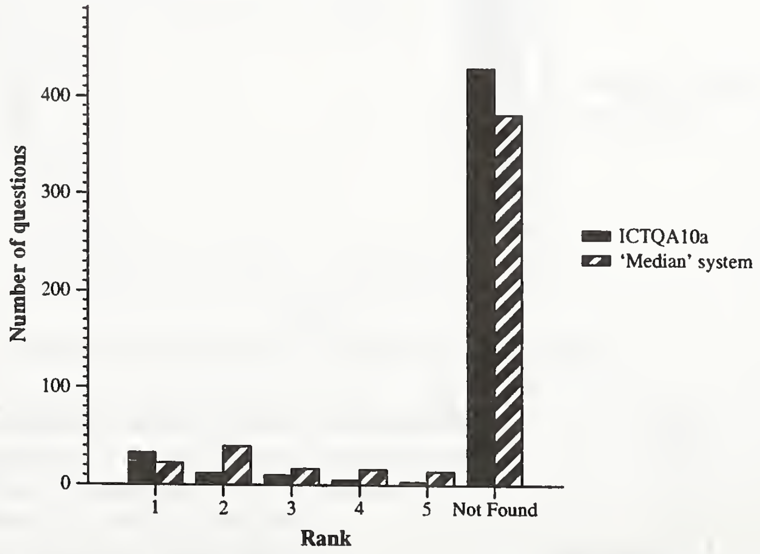
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	ALIC01M2
Num questions	492
Mean reciprocal rank (strict)	0.300
Mean reciprocal rank (lenient)	0.306
Num answers not found (strict)	297 (60.4%)
Num answers not found (lenient)	293 (59.6%)
Number of times NIL returned	4
Number of times NIL correctly returned	0
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	23%



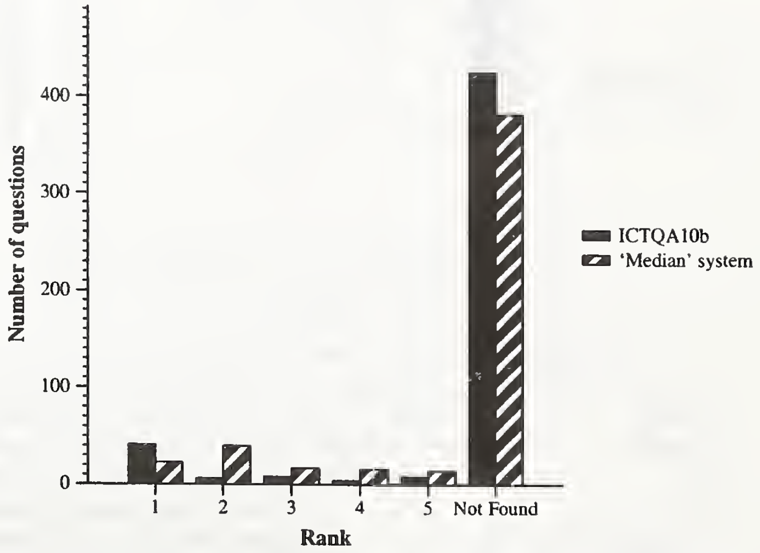
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	ICTQA10a
Num questions	492
Mean reciprocal rank (strict)	0.090
Mean reciprocal rank (lenient)	0.102
Num answers not found (strict)	429 (87.2%)
Num answers not found (lenient)	418 (85.0%)
Number of times NIL returned	35
Number of times NIL correctly returned	10
Percentage of answers system confident about	65%
Percentage of confident answers that were correct	8%



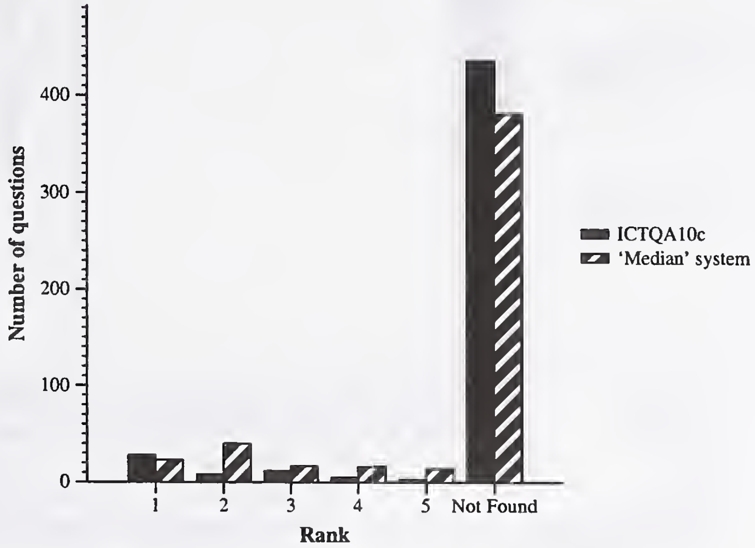
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	ICTQA10b
Num questions	492
Mean reciprocal rank (strict)	0.100
Mean reciprocal rank (lenient)	0.109
Num answers not found (strict)	425 (86.4%)
Num answers not found (lenient)	416 (84.6%)
Number of times NIL returned	55
Number of times NIL correctly returned	15
Percentage of answers system confident about	65%
Percentage of confident answers that were correct	10%



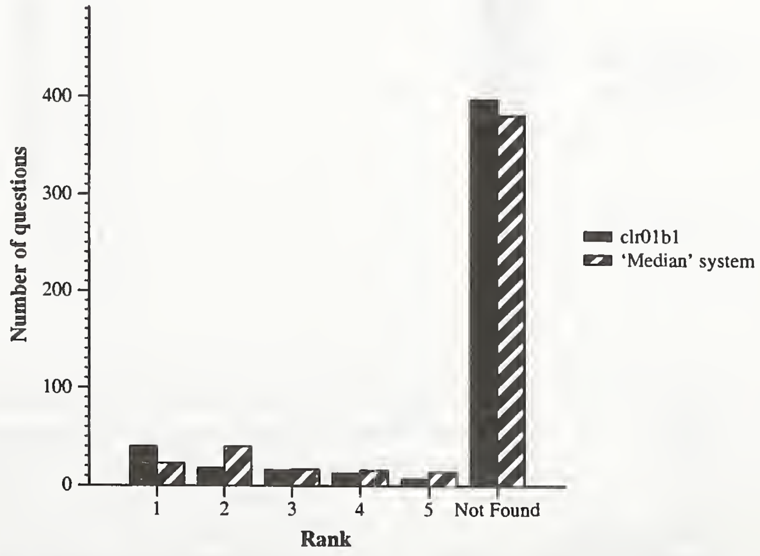
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	ICTQA10c
Num questions	492
Mean reciprocal rank (strict)	0.077
Mean reciprocal rank (lenient)	0.089
Num answers not found (strict)	436 (88.6%)
Num answers not found (lenient)	426 (86.6%)
Number of times NIL returned	60
Number of times NIL correctly returned	11
Percentage of answers system confident about	65%
Percentage of confident answers that were correct	6%



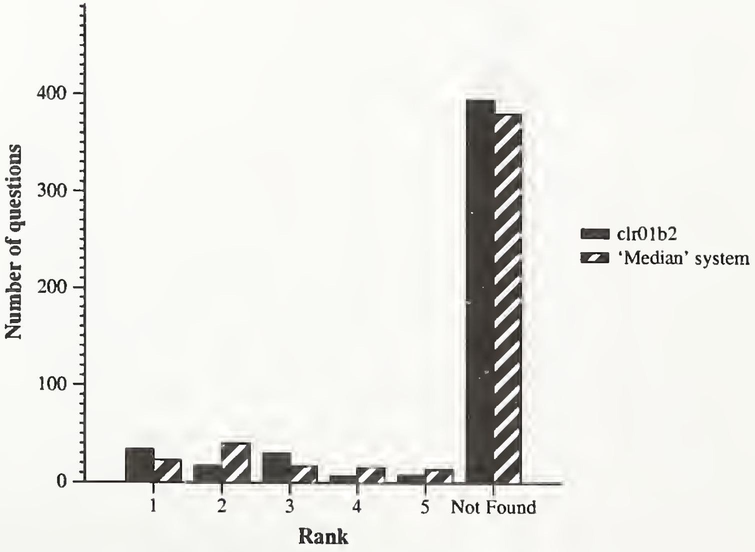
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	clr01b1
Num questions	492
Mean reciprocal rank (strict)	0.120
Mean reciprocal rank (lenient)	0.130
Num answers not found (strict)	398 (80.9%)
Num answers not found (lenient)	392 (79.7%)
Number of times NIL returned	60
Number of times NIL correctly returned	5
Percentage of answers system confident about	96%
Percentage of confident answers that were correct	8%



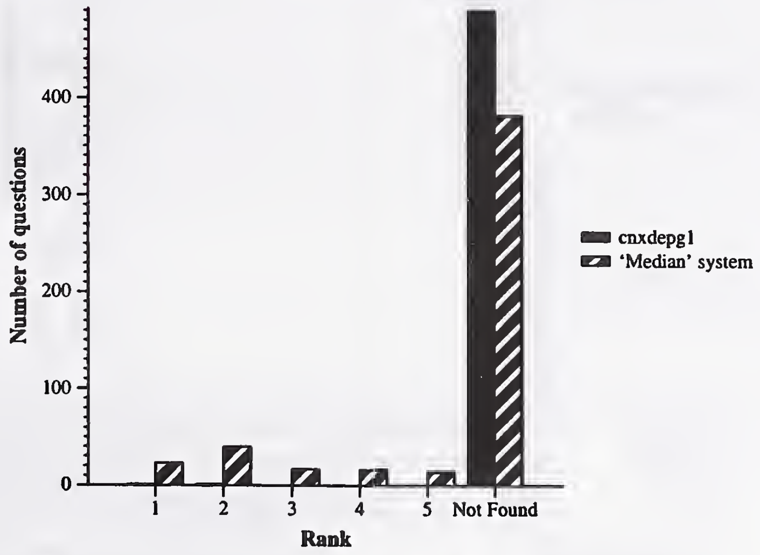
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	clr01b2
Num questions	492
Mean reciprocal rank (strict)	0.114
Mean reciprocal rank (lenient)	0.123
Num answers not found (strict)	396 (80.5%)
Num answers not found (lenient)	390 (79.3%)
Number of times NIL returned	227
Number of times NIL correctly returned	24
Percentage of answers system confident about	97%
Percentage of confident answers that were correct	7%



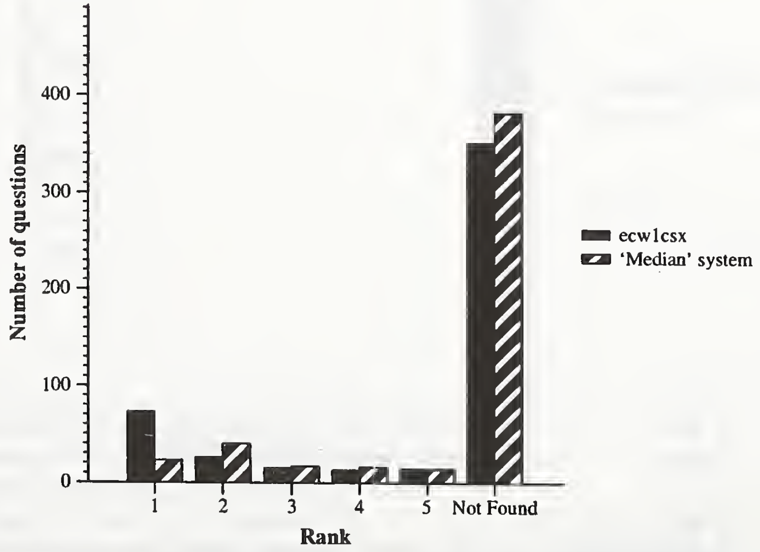
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	cnxdepg1
Num questions	492
Mean reciprocal rank (strict)	0.003
Mean reciprocal rank (lenient)	0.003
Num answers not found (strict)	490 (99.6%)
Num answers not found (lenient)	490 (99.6%)
Number of times NIL returned	26
Number of times NIL correctly returned	1
Percentage of answers system confident about	94%
Percentage of confident answers that were correct	0%



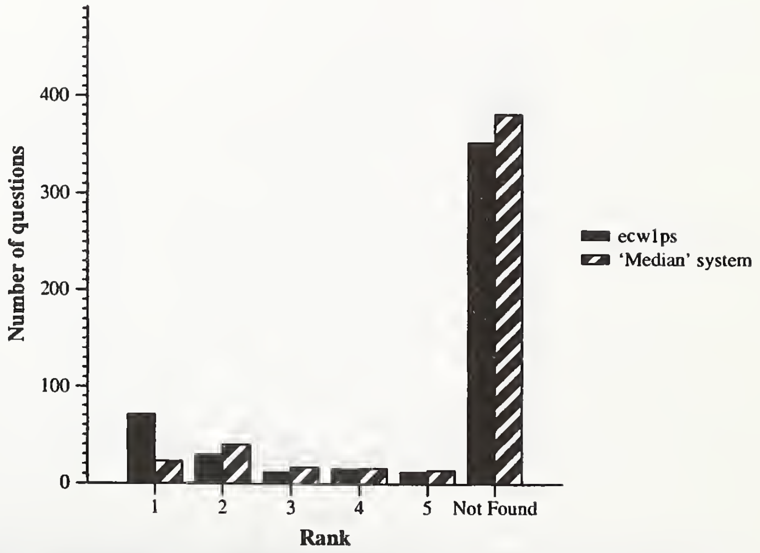
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	ecw1csx
Num questions	492
Mean reciprocal rank (strict)	0.197
Mean reciprocal rank (lenient)	0.204
Num answers not found (strict)	351 (71.3%)
Num answers not found (lenient)	346 (70.3%)
Number of times NIL returned	86
Number of times NIL correctly returned	9
Percentage of answers system confident about	32%
Percentage of confident answers that were correct	19%



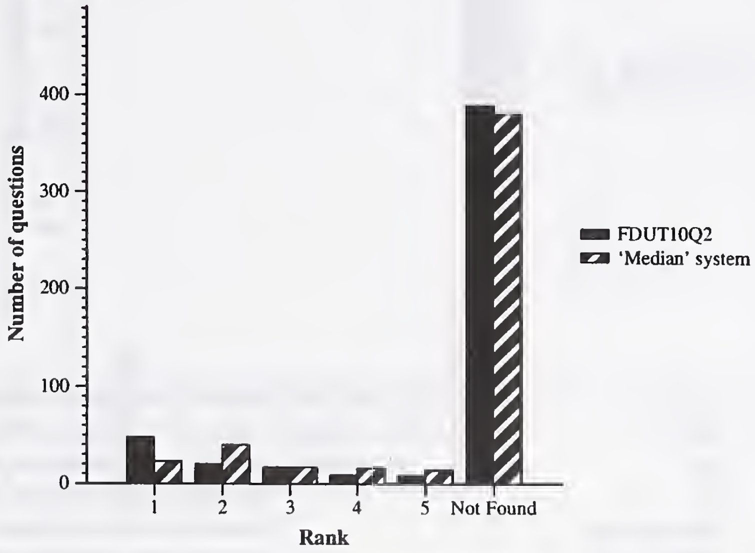
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	ecw1ps
Num questions	492
Mean reciprocal rank (strict)	0.194
Mean reciprocal rank (lenient)	0.194
Num answers not found (strict)	353 (71.7%)
Num answers not found (lenient)	353 (71.7%)
Number of times NIL returned	107
Number of times NIL correctly returned	14
Percentage of answers system confident about	31%
Percentage of confident answers that were correct	19%



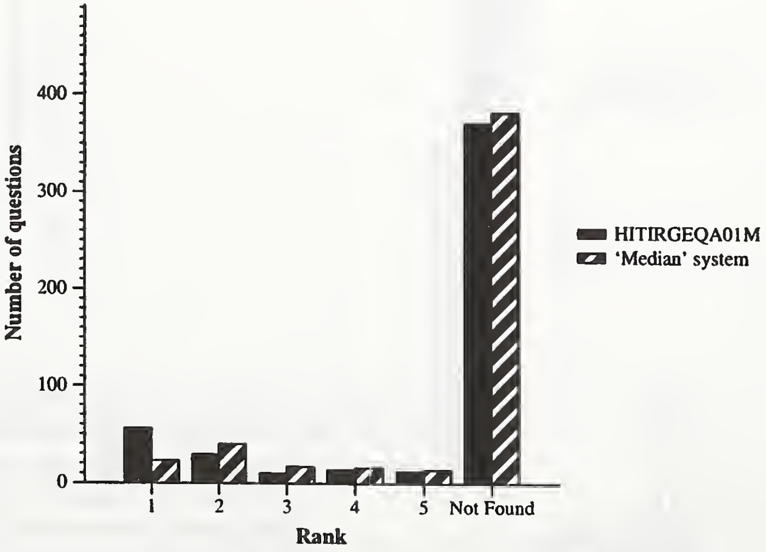
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	FDUT10Q2
Num questions	492
Mean reciprocal rank (strict)	0.137
Mean reciprocal rank (lenient)	0.145
Num answers not found (strict)	390 (79.3%)
Num answers not found (lenient)	383 (77.8%)
Number of times NIL returned	175
Number of times NIL correctly returned	19
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	9%



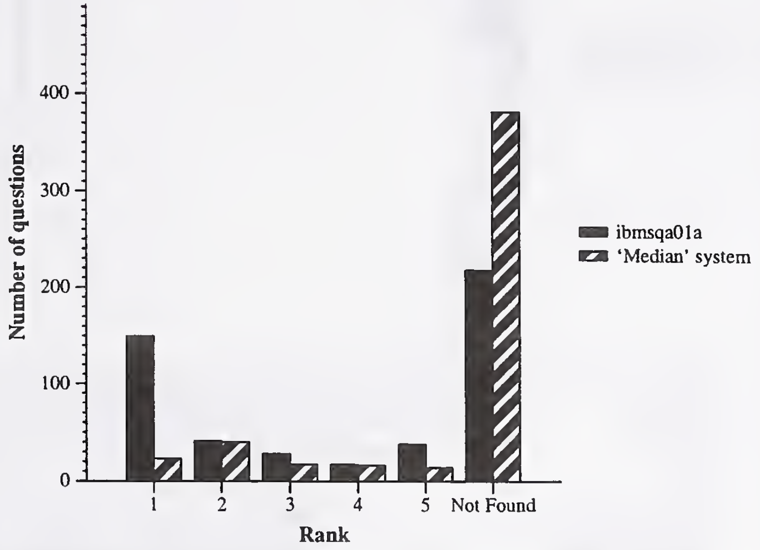
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	HITIRGEQA01M
Num questions	492
Mean reciprocal rank (strict)	0.162
Mean reciprocal rank (lenient)	0.166
Num answers not found (strict)	371 (75.4%)
Num answers not found (lenient)	368 (74.8%)
Number of times NIL returned	74
Number of times NIL correctly returned	10
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	11%



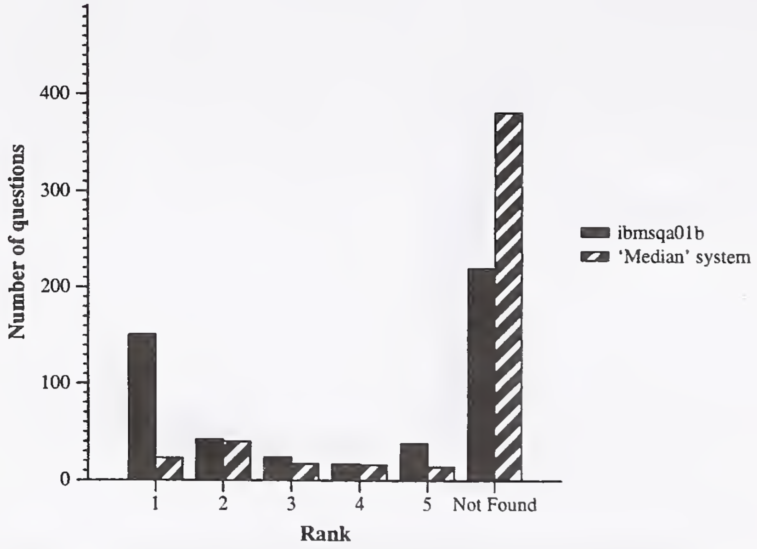
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	ibmsqa01a
Num questions	492
Mean reciprocal rank (strict)	0.390
Mean reciprocal rank (lenient)	0.403
Num answers not found (strict)	218 (44.3%)
Num answers not found (lenient)	212 (43.1%)
Number of times NIL returned	192
Number of times NIL correctly returned	28
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	30%



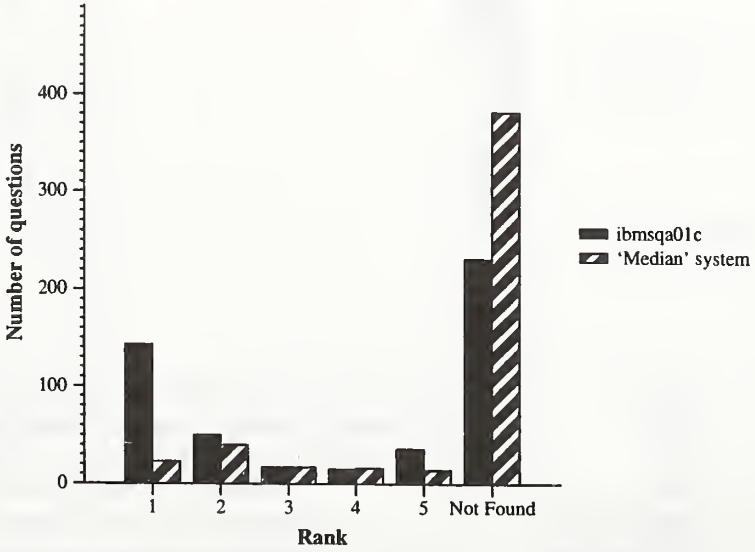
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	ibmsqa01b
Num questions	492
Mean reciprocal rank (strict)	0.390
Mean reciprocal rank (lenient)	0.403
Num answers not found (strict)	220 (44.7%)
Num answers not found (lenient)	215 (43.7%)
Number of times NIL returned	193
Number of times NIL correctly returned	30
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	30%



Number of questions for which the first correct response was at a given rank

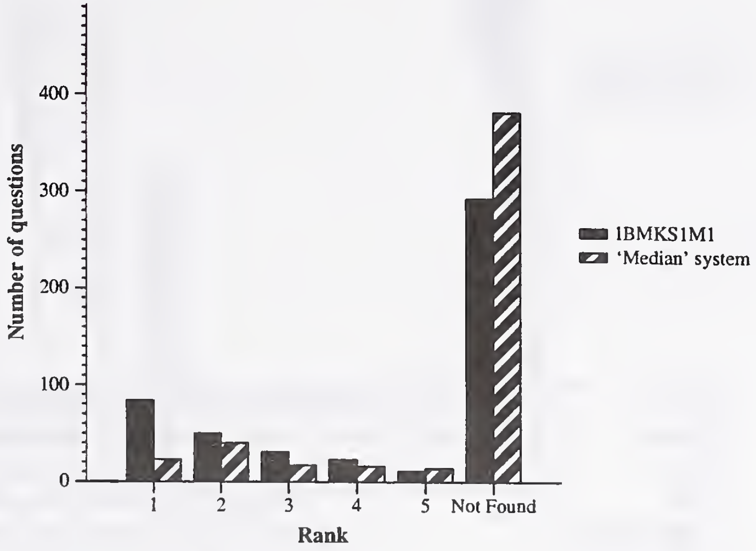
Summary Statistics	
Run ID	ibmsqa01c
Num questions	492
Mean reciprocal rank (strict)	0.375
Mean reciprocal rank (lenient)	0.388
Num answers not found (strict)	231 (47.0%)
Num answers not found (lenient)	224 (45.5%)
Number of times NIL returned	209
Number of times NIL correctly returned	30
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	29%



Number of questions for which the first correct response was at a given rank

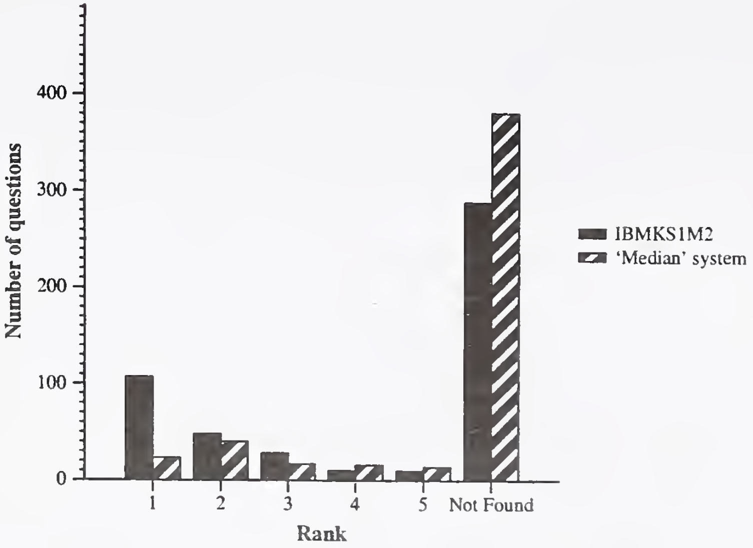
Question Answering track, Main task results — IBM T.J. Watson Research Center (Prager)

Summary Statistics	
Run ID	IBMKS1M1
Num questions	492
Mean reciprocal rank (strict)	0.259
Mean reciprocal rank (lenient)	0.270
Num answers not found (strict)	293 (59.6%)
Num answers not found (lenient)	286 (58.1%)
Number of times NIL returned	289
Number of times NIL correctly returned	36
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	17%



Number of questions for which the first correct response was at a given rank

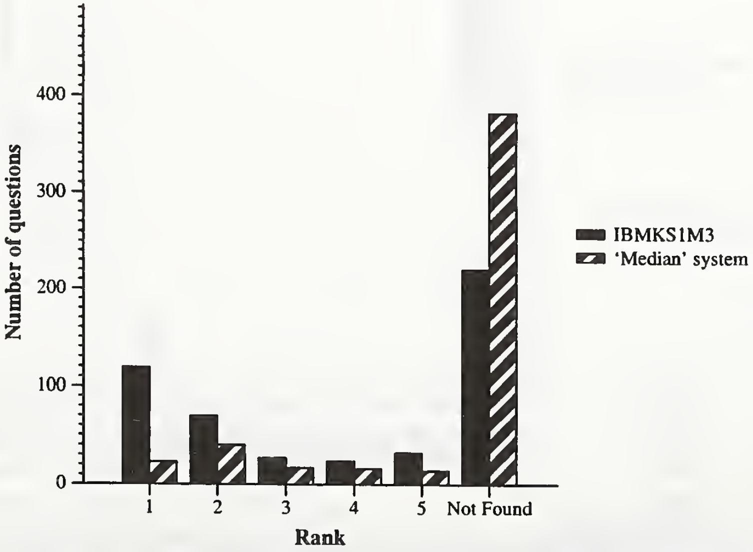
Summary Statistics	
Run ID	IBMKS1M2
Num questions	492
Mean reciprocal rank (strict)	0.294
Mean reciprocal rank (lenient)	0.299
Num answers not found (strict)	289 (58.7%)
Num answers not found (lenient)	282 (57.3%)
Number of times NIL returned	119
Number of times NIL correctly returned	11
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	21%



Number of questions for which the first correct response was at a given rank

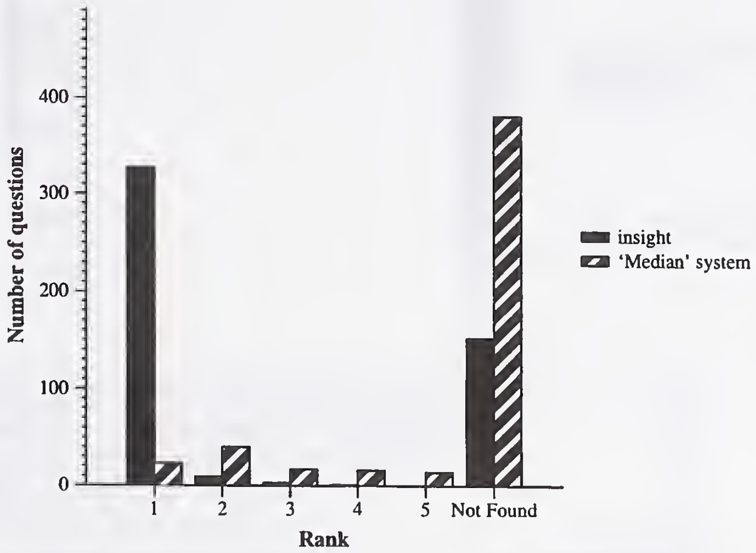
Question Answering track, Main task results — IBM T.J. Watson Research Center (Prager)

Summary Statistics	
Run ID	IBMKS1M3
Num questions	492
Mean reciprocal rank (strict)	0.357
Mean reciprocal rank (lenient)	0.365
Num answers not found (strict)	220 (44.7%)
Num answers not found (lenient)	211 (42.9%)
Number of times NIL returned	206
Number of times NIL correctly returned	27
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	24%



Number of questions for which the first correct response was at a given rank

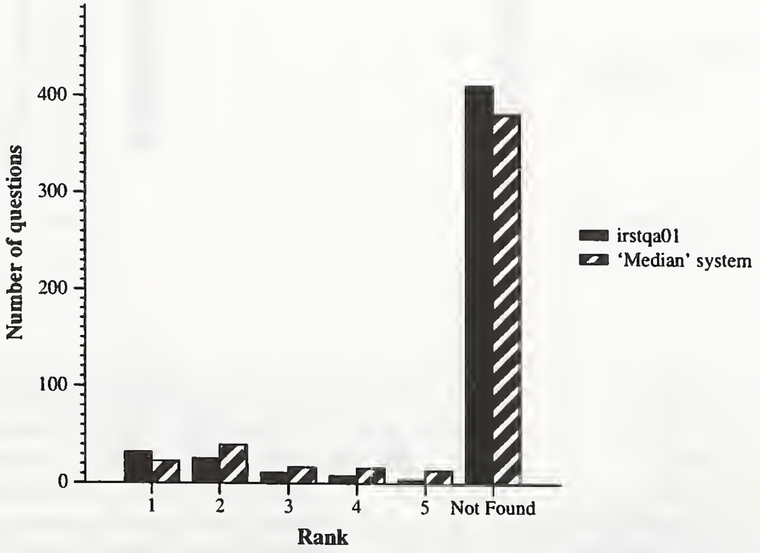
Summary Statistics	
Run ID	insight
Num questions	492
Mean reciprocal rank (strict)	0.676
Mean reciprocal rank (lenient)	0.686
Num answers not found (strict)	152 (30.9%)
Num answers not found (lenient)	147 (29.9%)
Number of times NIL returned	120
Number of times NIL correctly returned	38
Percentage of answers system confident about	75%
Percentage of confident answers that were correct	77%



Number of questions for which the first correct response was at a given rank

Question Answering track, Main task results — ITC-irst, Instituto per la Ricerca Scientifica e Tecnologica

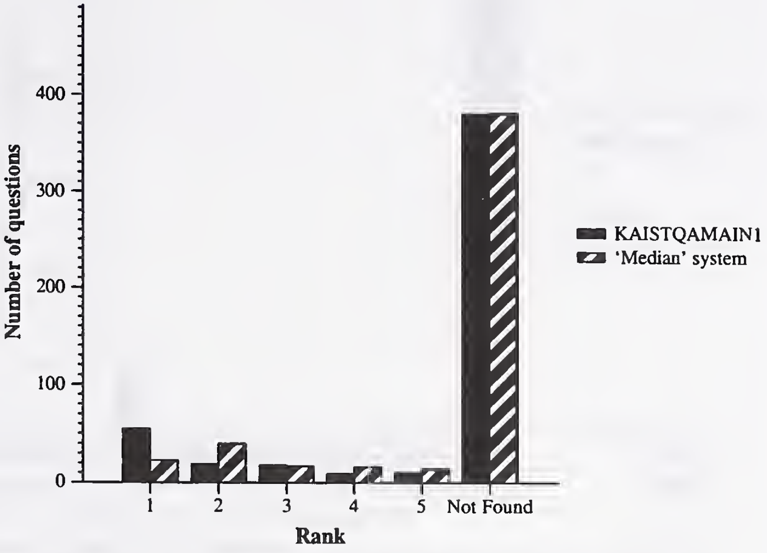
Summary Statistics	
Run ID	irstqa01
Num questions	492
Mean reciprocal rank (strict)	0.105
Mean reciprocal rank (lenient)	0.110
Num answers not found (strict)	411 (83.5%)
Num answers not found (lenient)	408 (82.9%)
Number of times NIL returned	47
Number of times NIL correctly returned	6
Percentage of answers system confident about	60%
Percentage of confident answers that were correct	5%



Number of questions for which the first correct response was at a given rank

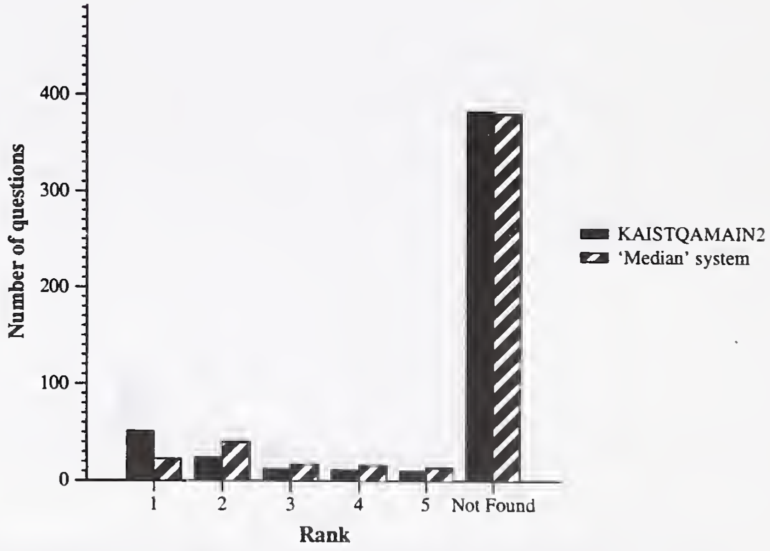
Question Answering track, Main task results — KAIST (Korea Advanced Institute of Science and Technology)

Summary Statistics	
Run ID	KAISTQAMAIN1
Num questions	492
Mean reciprocal rank (strict)	0.152
Mean reciprocal rank (lenient)	0.159
Num answers not found (strict)	381 (77.4%)
Num answers not found (lenient)	376 (76.4%)
Number of times NIL returned	6
Number of times NIL correctly returned	0
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	11%



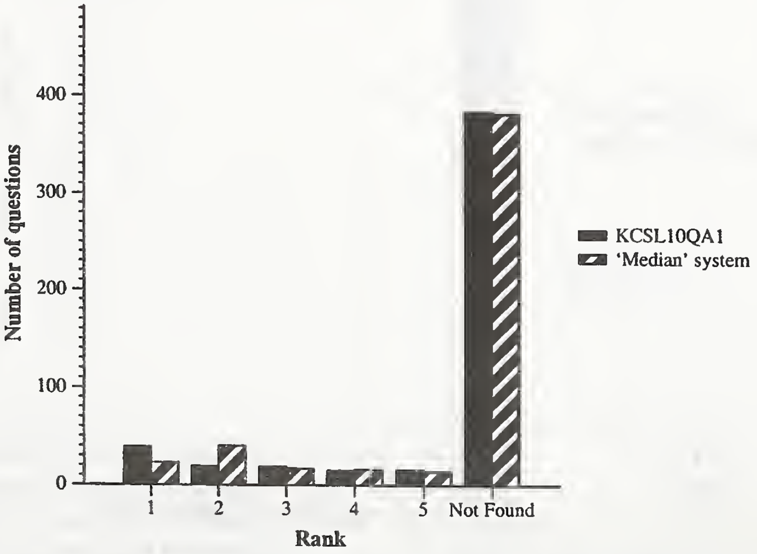
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	KAISTQAMAIN2
Num questions	492
Mean reciprocal rank (strict)	0.146
Mean reciprocal rank (lenient)	0.152
Num answers not found (strict)	384 (78.0%)
Num answers not found (lenient)	377 (76.6%)
Number of times NIL returned	5
Number of times NIL correctly returned	0
Percentage of answers system confident about	98%
Percentage of confident answers that were correct	10%



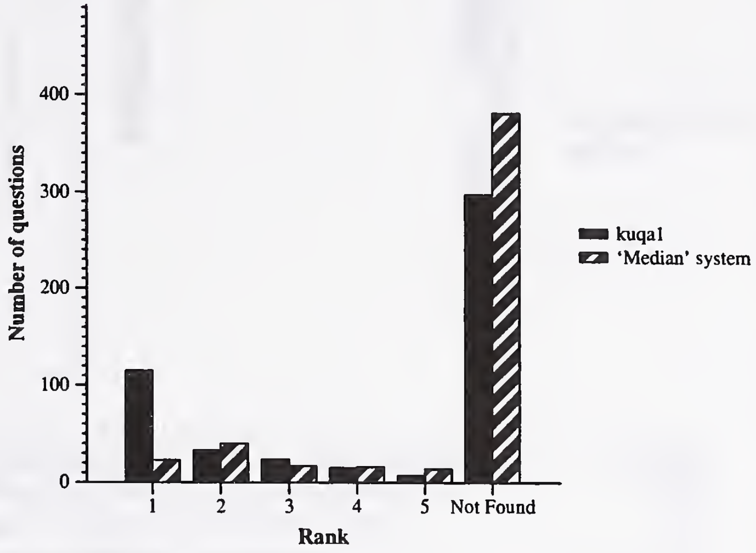
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	KCSL10QA1
Num questions	492
Mean reciprocal rank (strict)	0.126
Mean reciprocal rank (lenient)	0.131
Num answers not found (strict)	384 (78.0%)
Num answers not found (lenient)	380 (77.2%)
Number of times NIL returned	0
Number of times NIL correctly returned	0
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	7%



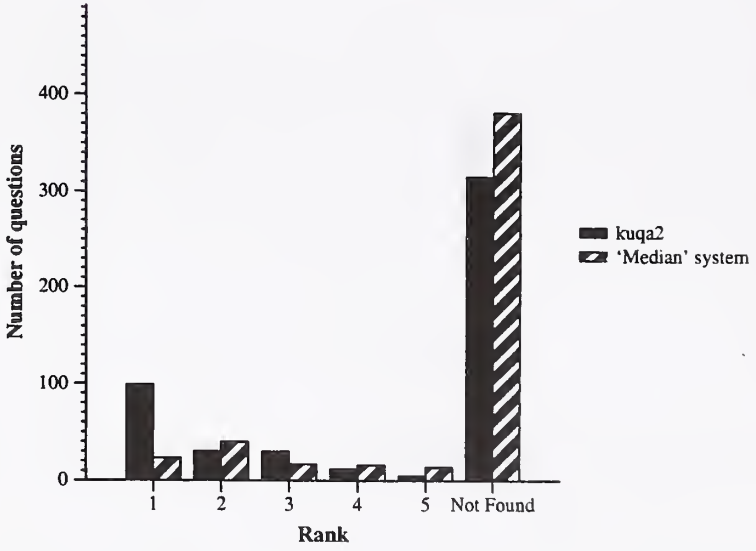
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	kuqa1
Num questions	492
Mean reciprocal rank (strict)	0.294
Mean reciprocal rank (lenient)	0.298
Num answers not found (strict)	298 (60.6%)
Num answers not found (lenient)	295 (60.0%)
Number of times NIL returned	6
Number of times NIL correctly returned	0
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	23%



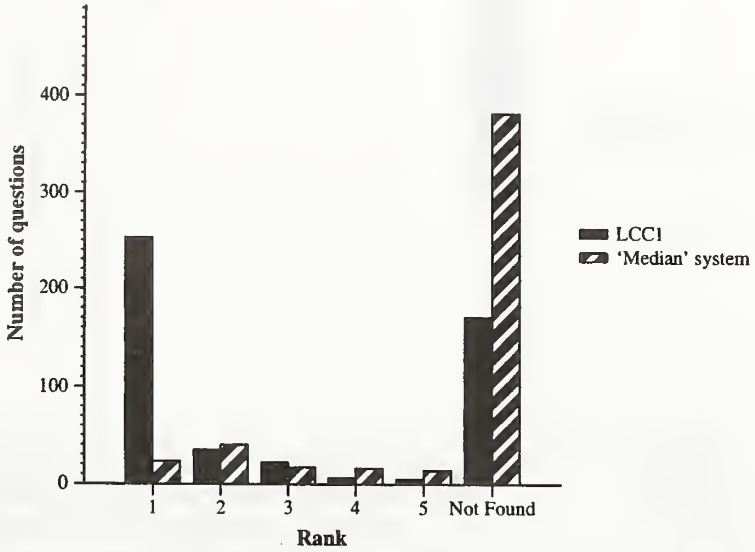
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	kuqa2
Num questions	492
Mean reciprocal rank (strict)	0.260
Mean reciprocal rank (lenient)	0.265
Num answers not found (strict)	316 (64.2%)
Num answers not found (lenient)	313 (63.6%)
Number of times NIL returned	7
Number of times NIL correctly returned	0
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	20%



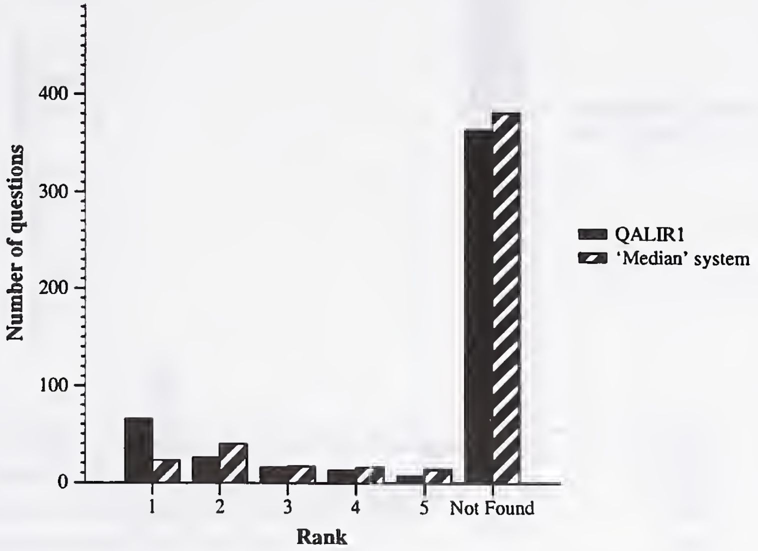
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	LCC1
Num questions	492
Mean reciprocal rank (strict)	0.570
Mean reciprocal rank (lenient)	0.587
Num answers not found (strict)	171 (34.8%)
Num answers not found (lenient)	159 (32.3%)
Number of times NIL returned	41
Number of times NIL correctly returned	31
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	51%



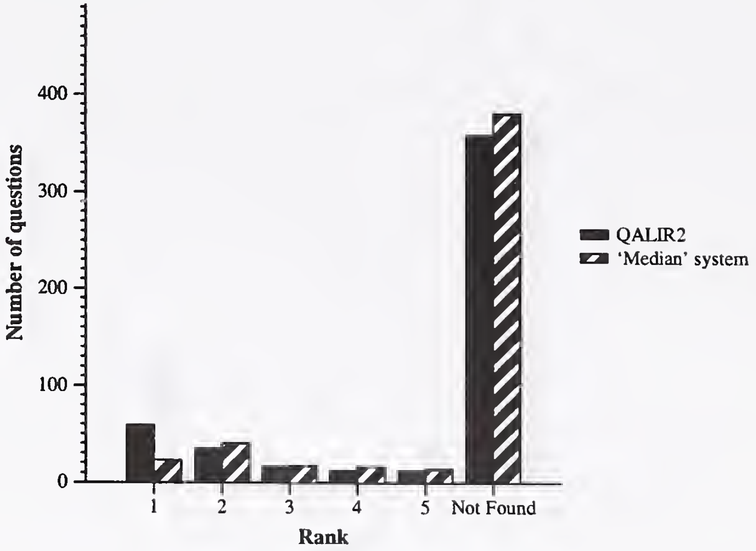
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	QALIR1
Num questions	492
Mean reciprocal rank (strict)	0.181
Mean reciprocal rank (lenient)	0.192
Num answers not found (strict)	364 (74.0%)
Num answers not found (lenient)	357 (72.6%)
Number of times NIL returned	22
Number of times NIL correctly returned	5
Percentage of answers system confident about	76%
Percentage of confident answers that were correct	12%



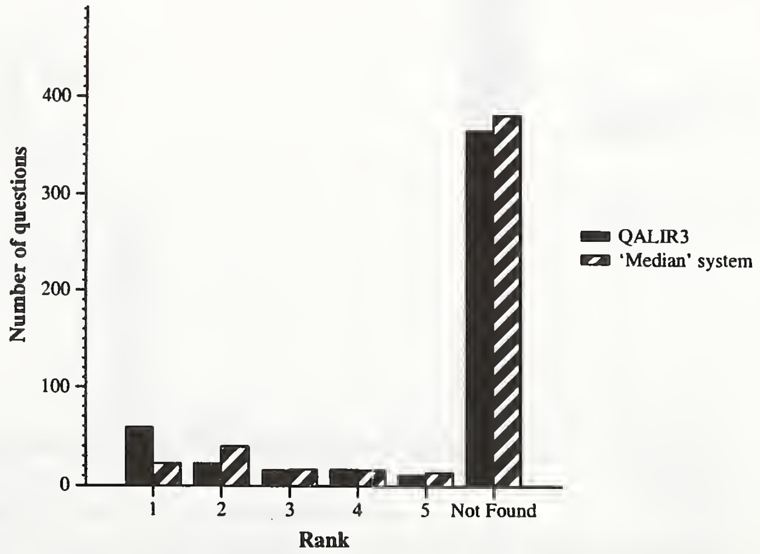
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	QALIR2
Num questions	492
Mean reciprocal rank (strict)	0.176
Mean reciprocal rank (lenient)	0.188
Num answers not found (strict)	359 (73.0%)
Num answers not found (lenient)	350 (71.1%)
Number of times NIL returned	22
Number of times NIL correctly returned	5
Percentage of answers system confident about	76%
Percentage of confident answers that were correct	11%



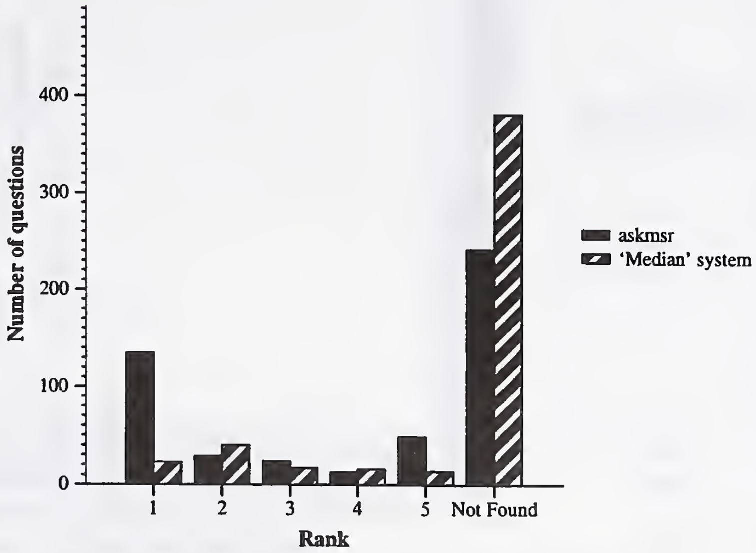
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	QALIR3
Num questions	492
Mean reciprocal rank (strict)	0.167
Mean reciprocal rank (lenient)	0.179
Num answers not found (strict)	366 (74.4%)
Num answers not found (lenient)	357 (72.6%)
Number of times NIL returned	22
Number of times NIL correctly returned	5
Percentage of answers system confident about	76%
Percentage of confident answers that were correct	11%



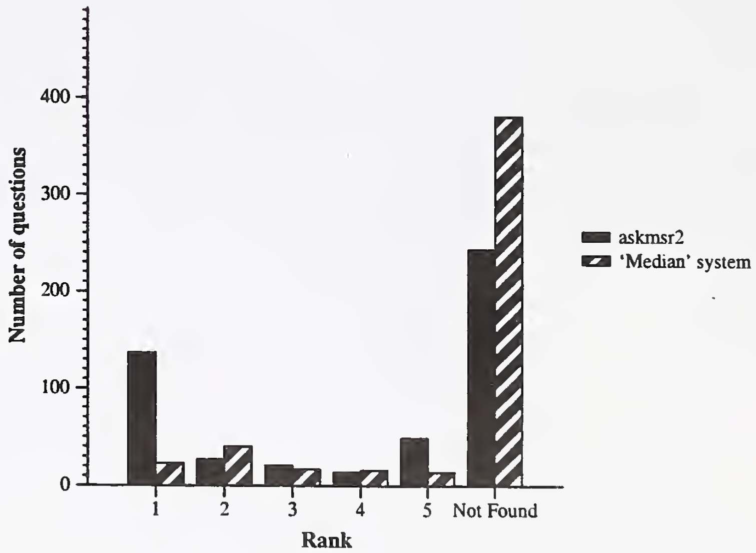
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	askmsr
Num questions	492
Mean reciprocal rank (strict)	0.347
Mean reciprocal rank (lenient)	0.434
Num answers not found (strict)	242 (49.2%)
Num answers not found (lenient)	197 (40.0%)
Number of times NIL returned	491
Number of times NIL correctly returned	49
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	27%



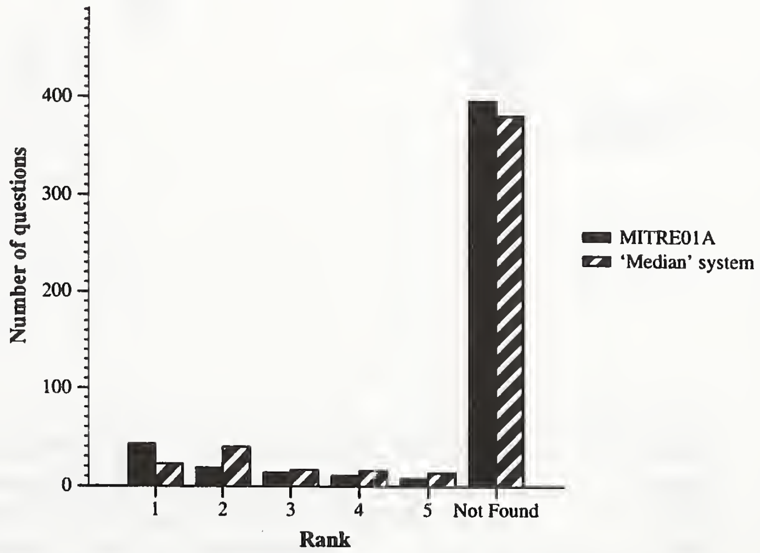
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	askmsr2
Num questions	492
Mean reciprocal rank (strict)	0.347
Mean reciprocal rank (lenient)	0.437
Num answers not found (strict)	244 (49.6%)
Num answers not found (lenient)	195 (39.6%)
Number of times NIL returned	492
Number of times NIL correctly returned	49
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	27%



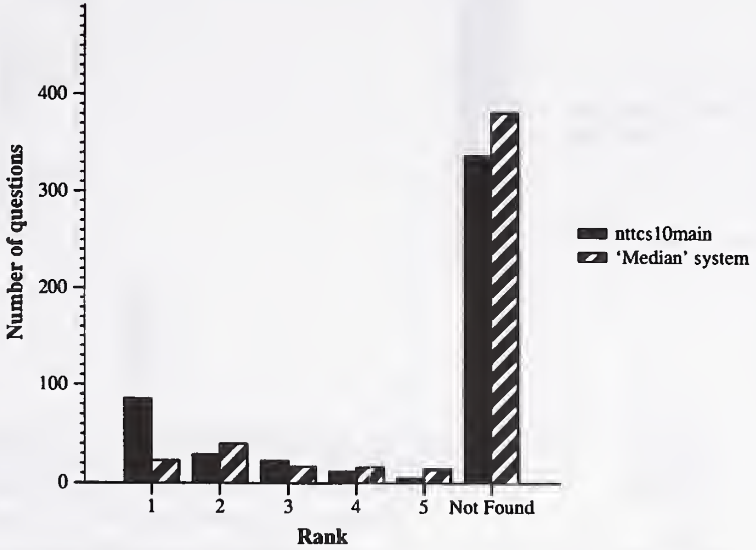
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	MITRE01A
Num questions	492
Mean reciprocal rank (strict)	0.125
Mean reciprocal rank (lenient)	0.131
Num answers not found (strict)	397 (80.7%)
Num answers not found (lenient)	395 (80.3%)
Number of times NIL returned	39
Number of times NIL correctly returned	5
Percentage of answers system confident about	92%
Percentage of confident answers that were correct	8%



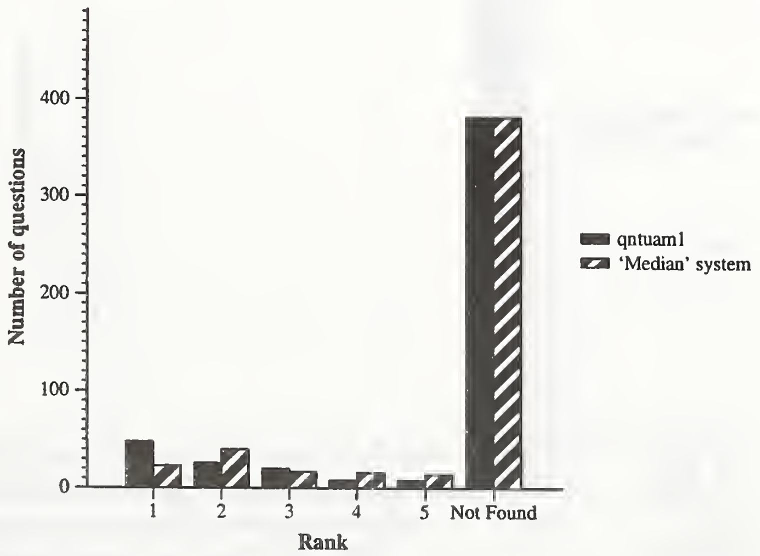
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	nttcs10main
Num questions	492
Mean reciprocal rank (strict)	0.228
Mean reciprocal rank (lenient)	0.231
Num answers not found (strict)	337 (68.5%)
Num answers not found (lenient)	335 (68.1%)
Number of times NIL returned	232
Number of times NIL correctly returned	23
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	17%



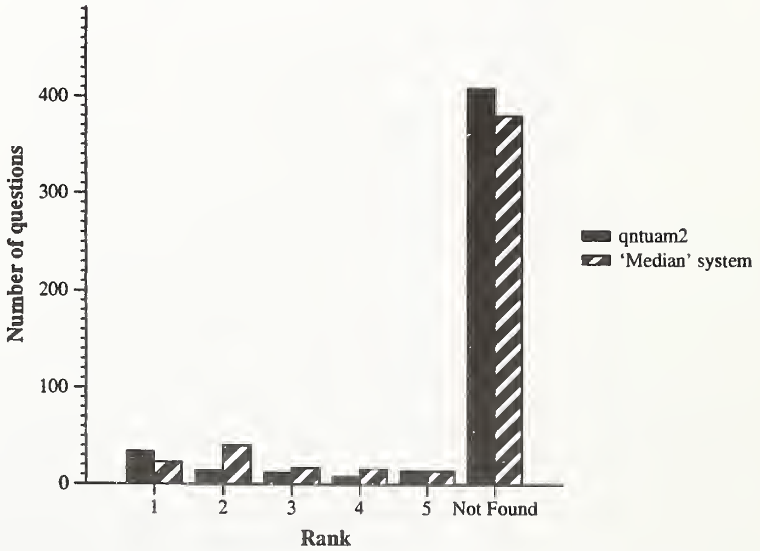
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	qntuam1
Num questions	492
Mean reciprocal rank (strict)	0.145
Mean reciprocal rank (lenient)	0.146
Num answers not found (strict)	382 (77.6%)
Num answers not found (lenient)	380 (77.2%)
Number of times NIL returned	56
Number of times NIL correctly returned	9
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	9%



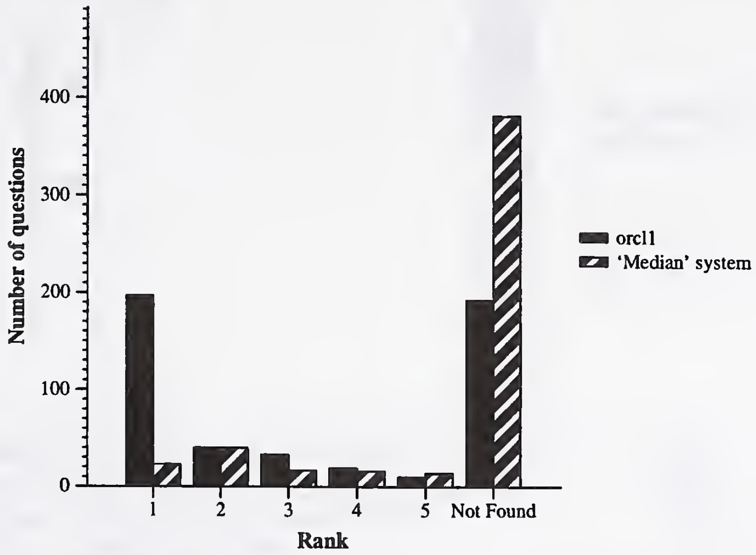
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	qntuam2
Num questions	492
Mean reciprocal rank (strict)	0.101
Mean reciprocal rank (lenient)	0.103
Num answers not found (strict)	410 (83.3%)
Num answers not found (lenient)	408 (82.9%)
Number of times NIL returned	14
Number of times NIL correctly returned	1
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	6%



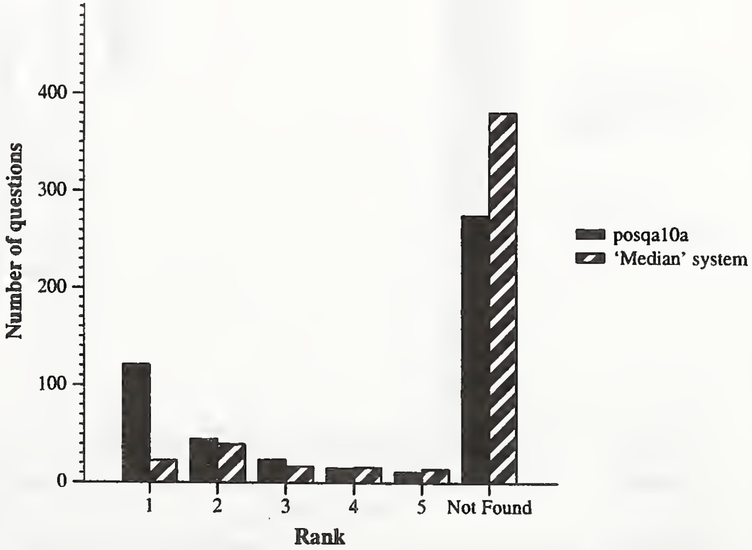
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	orcl1
Num questions	492
Mean reciprocal rank (strict)	0.477
Mean reciprocal rank (lenient)	0.491
Num answers not found (strict)	193 (39.2%)
Num answers not found (lenient)	184 (37.4%)
Number of times NIL returned	82
Number of times NIL correctly returned	35
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	40%



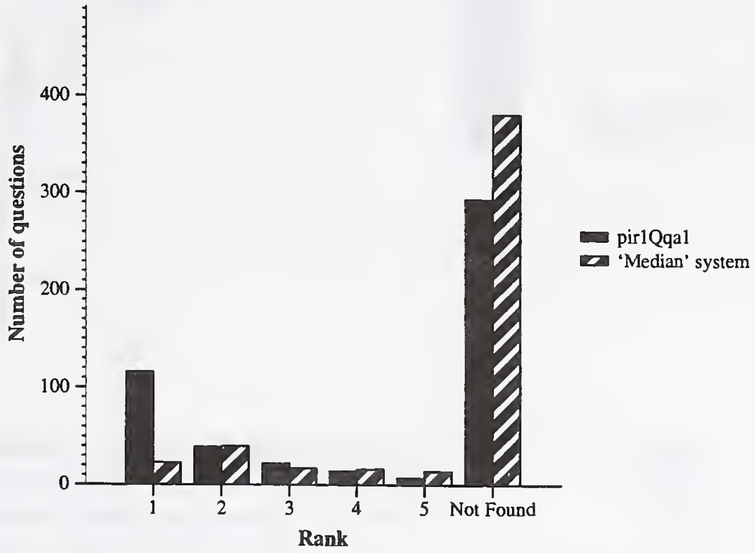
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	posqa10a
Num questions	492
Mean reciprocal rank (strict)	0.320
Mean reciprocal rank (lenient)	0.335
Num answers not found (strict)	276 (56.1%)
Num answers not found (lenient)	260 (52.8%)
Number of times NIL returned	13
Number of times NIL correctly returned	3
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	24%



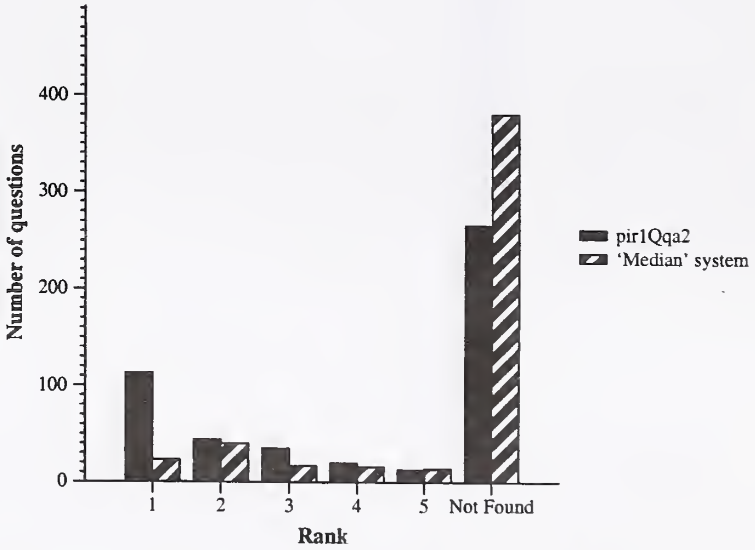
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	pir1Qqa1
Num questions	492
Mean reciprocal rank (strict)	0.300
Mean reciprocal rank (lenient)	0.310
Num answers not found (strict)	294 (59.8%)
Num answers not found (lenient)	287 (58.3%)
Number of times NIL returned	8
Number of times NIL correctly returned	0
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	23%



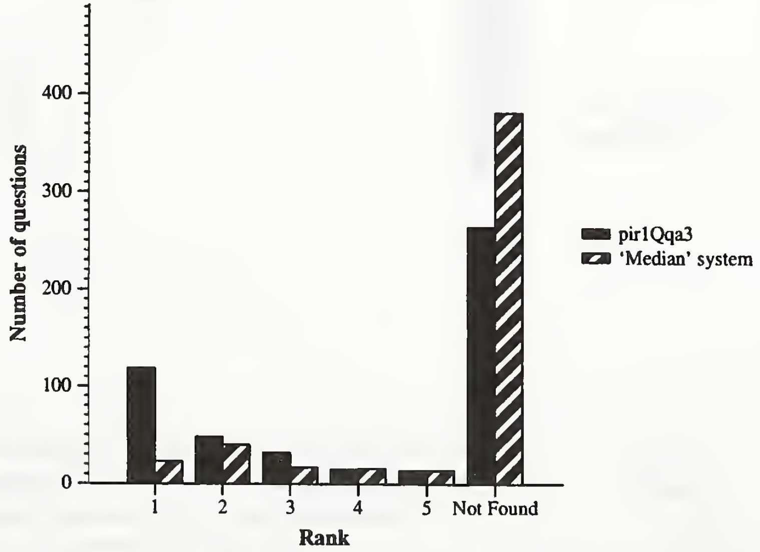
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	pir1Qqa2
Num questions	492
Mean reciprocal rank (strict)	0.314
Mean reciprocal rank (lenient)	0.319
Num answers not found (strict)	267 (54.3%)
Num answers not found (lenient)	265 (53.9%)
Number of times NIL returned	8
Number of times NIL correctly returned	0
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	22%



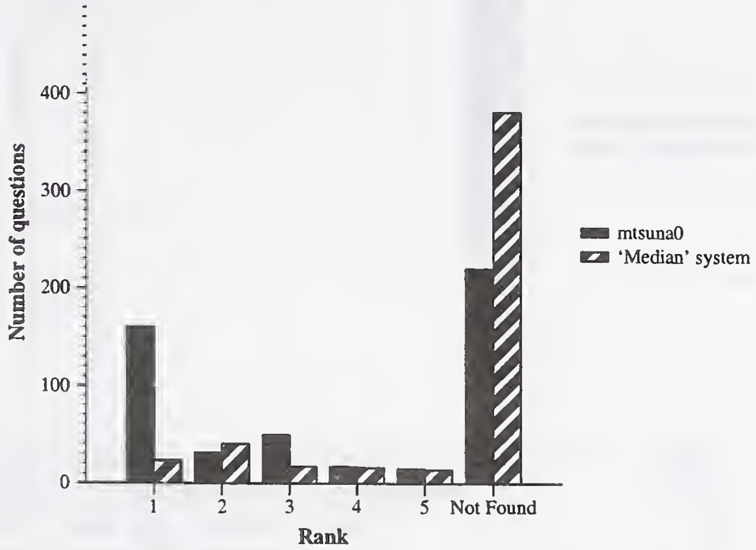
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	pir1Qqa3
Num questions	492
Mean reciprocal rank (strict)	0.326
Mean reciprocal rank (lenient)	0.331
Num answers not found (strict)	264 (53.7%)
Num answers not found (lenient)	260 (52.8%)
Number of times NIL returned	5
Number of times NIL correctly returned	0
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	24%



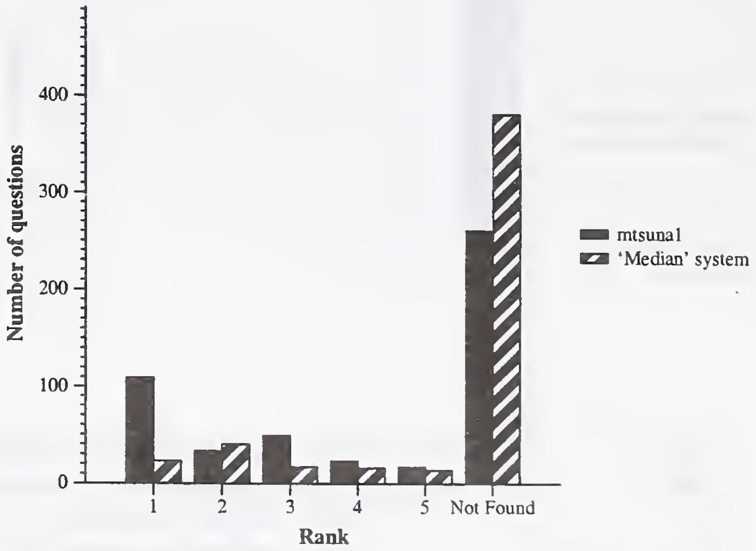
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	mtsuna0
Num questions	492
Mean reciprocal rank (strict)	0.405
Mean reciprocal rank (lenient)	0.418
Num answers not found (strict)	220 (44.7%)
Num answers not found (lenient)	213 (43.3%)
Number of times NIL returned	492
Number of times NIL correctly returned	49
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	32%



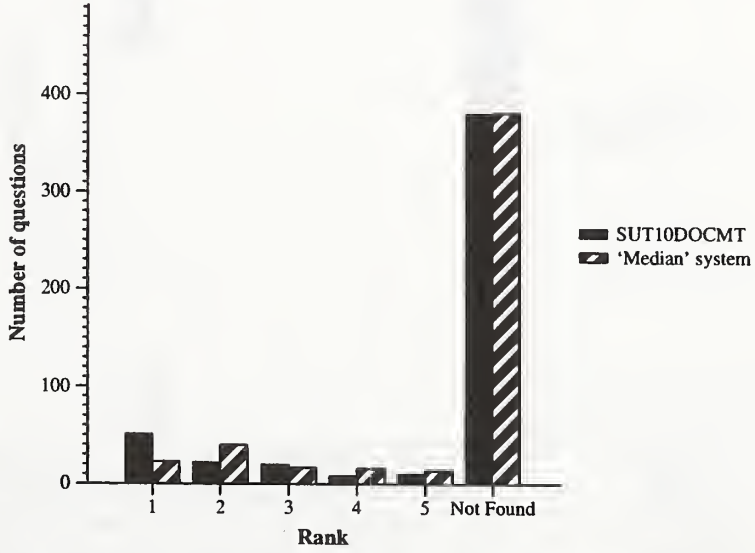
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	mtsuna1
Num questions	492
Mean reciprocal rank (strict)	0.307
Mean reciprocal rank (lenient)	0.322
Num answers not found (strict)	261 (53.0%)
Num answers not found (lenient)	252 (51.2%)
Number of times NIL returned	492
Number of times NIL correctly returned	49
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	22%



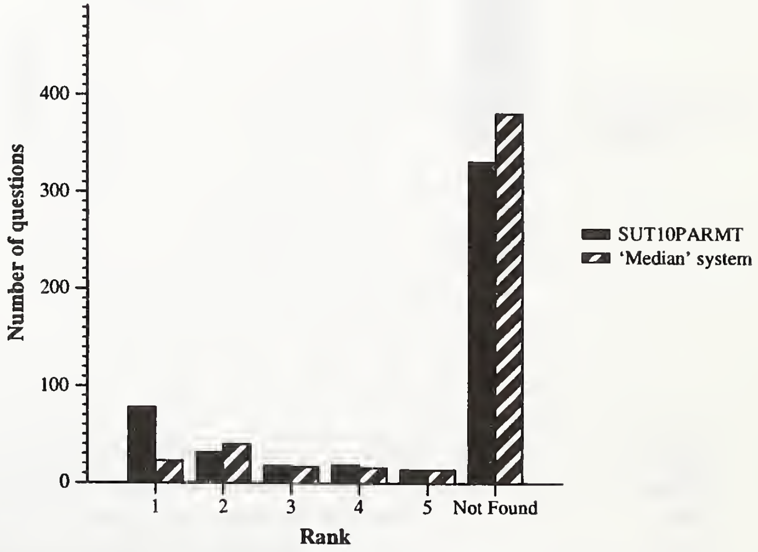
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	SUT10DOCMT
Num questions	492
Mean reciprocal rank (strict)	0.148
Mean reciprocal rank (lenient)	0.162
Num answers not found (strict)	381 (77.4%)
Num answers not found (lenient)	369 (75.0%)
Number of times NIL returned	3
Number of times NIL correctly returned	0
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	10%



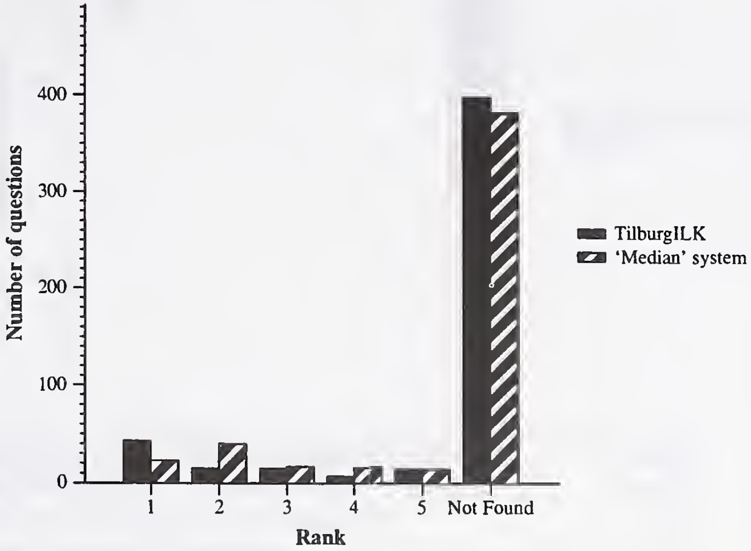
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	SUT10PARMT
Num questions	492
Mean reciprocal rank (strict)	0.218
Mean reciprocal rank (lenient)	0.230
Num answers not found (strict)	332 (67.5%)
Num answers not found (lenient)	326 (66.3%)
Number of times NIL returned	3
Number of times NIL correctly returned	0
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	15%



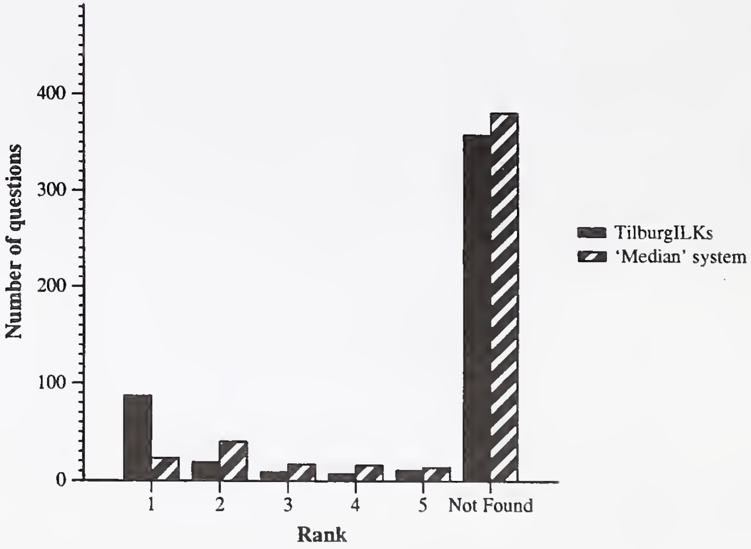
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	TilburgILK
Num questions	492
Mean reciprocal rank (strict)	0.122
Mean reciprocal rank (lenient)	0.128
Num answers not found (strict)	398 (80.9%)
Num answers not found (lenient)	394 (80.1%)
Number of times NIL returned	9
Number of times NIL correctly returned	0
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	8%



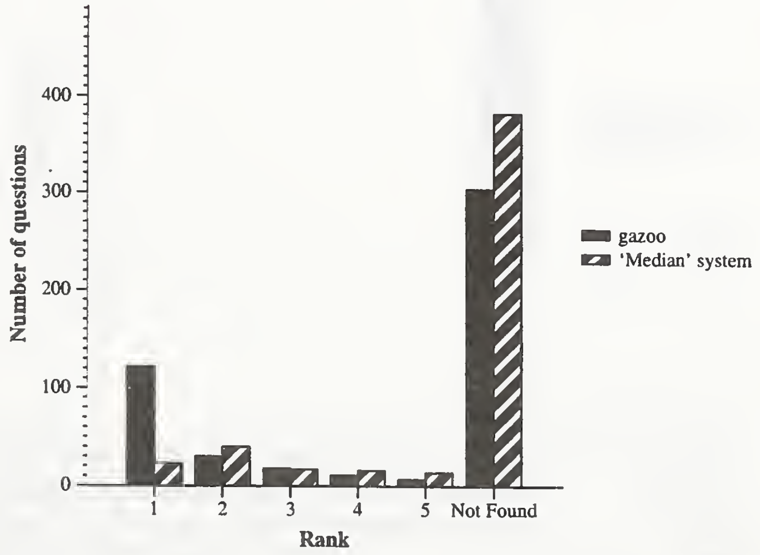
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	TilburgILKs
Num questions	492
Mean reciprocal rank (strict)	0.210
Mean reciprocal rank (lenient)	0.234
Num answers not found (strict)	359 (73.0%)
Num answers not found (lenient)	347 (70.5%)
Number of times NIL returned	9
Number of times NIL correctly returned	0
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	17%



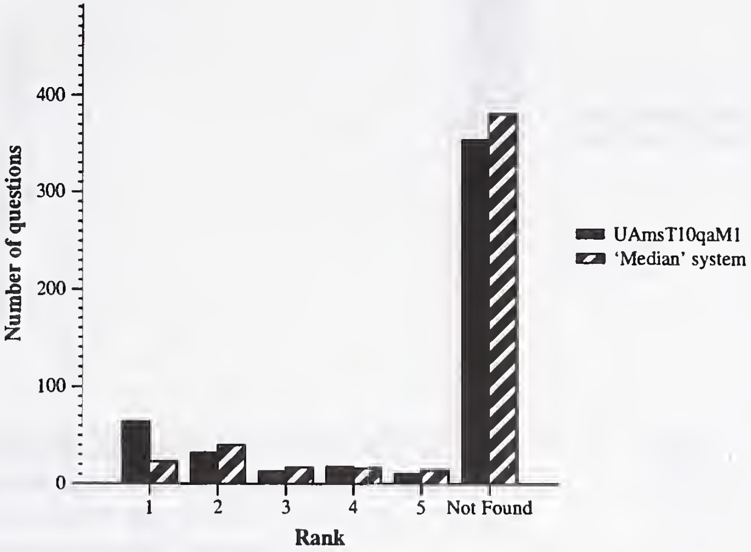
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	gazoo
Num questions	492
Mean reciprocal rank (strict)	0.299
Mean reciprocal rank (lenient)	0.311
Num answers not found (strict)	304 (61.8%)
Num answers not found (lenient)	300 (61.0%)
Number of times NIL returned	11
Number of times NIL correctly returned	0
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	24%



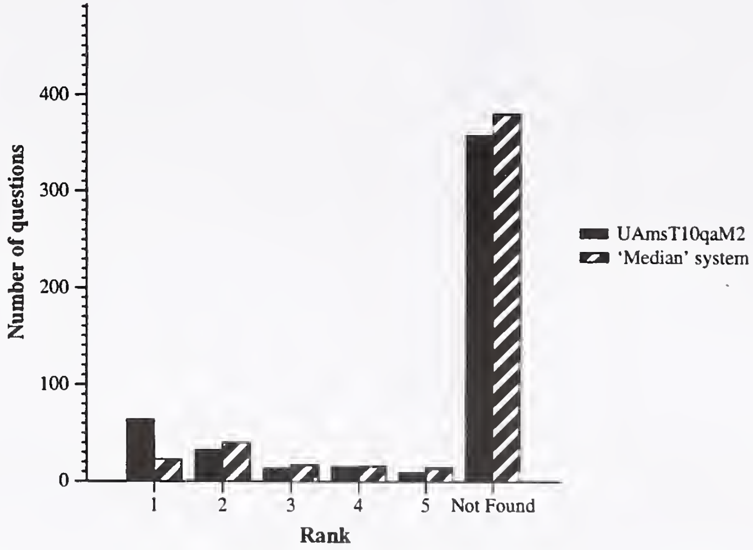
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	UAmsT10qaM1
Num questions	492
Mean reciprocal rank (strict)	0.185
Mean reciprocal rank (lenient)	0.197
Num answers not found (strict)	355 (72.2%)
Num answers not found (lenient)	346 (70.3%)
Number of times NIL returned	79
Number of times NIL correctly returned	12
Percentage of answers system confident about	59%
Percentage of confident answers that were correct	14%



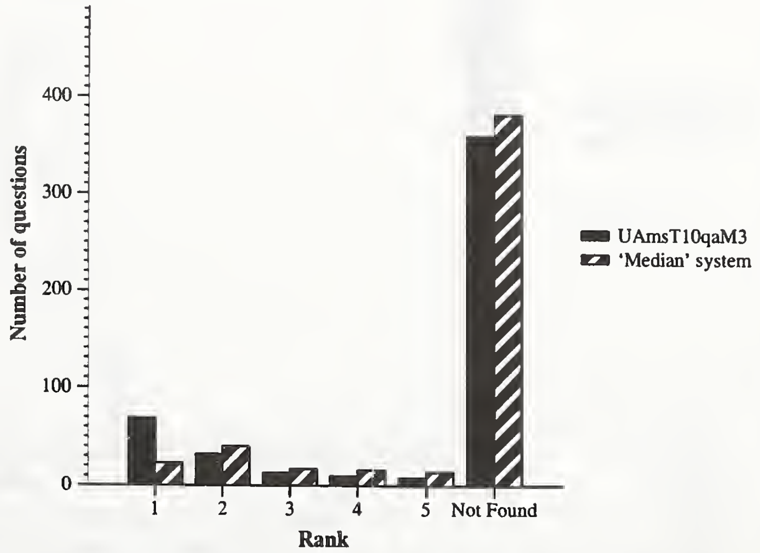
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	UAmsT10qaM2
Num questions	492
Mean reciprocal rank (strict)	0.183
Mean reciprocal rank (lenient)	0.196
Num answers not found (strict)	359 (73.0%)
Num answers not found (lenient)	349 (70.9%)
Number of times NIL returned	78
Number of times NIL correctly returned	12
Percentage of answers system confident about	67%
Percentage of confident answers that were correct	14%



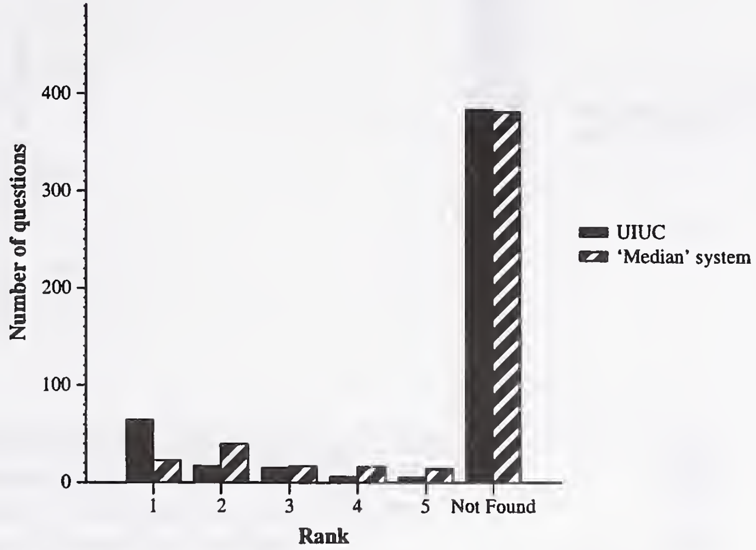
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	UAmsT10qaM3
Num questions	492
Mean reciprocal rank (strict)	0.190
Mean reciprocal rank (lenient)	0.203
Num answers not found (strict)	360 (73.2%)
Num answers not found (lenient)	351 (71.3%)
Number of times NIL returned	81
Number of times NIL correctly returned	17
Percentage of answers system confident about	71%
Percentage of confident answers that were correct	13%



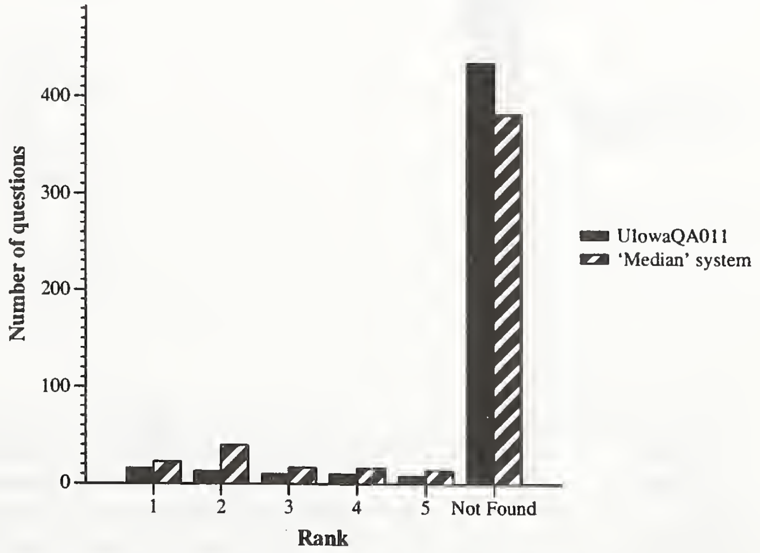
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	UIUC
Num questions	492
Mean reciprocal rank (strict)	0.165
Mean reciprocal rank (lenient)	0.193
Num answers not found (strict)	384 (78.0%)
Num answers not found (lenient)	367 (74.6%)
Number of times NIL returned	180
Number of times NIL correctly returned	19
Percentage of answers system confident about	63%
Percentage of confident answers that were correct	14%



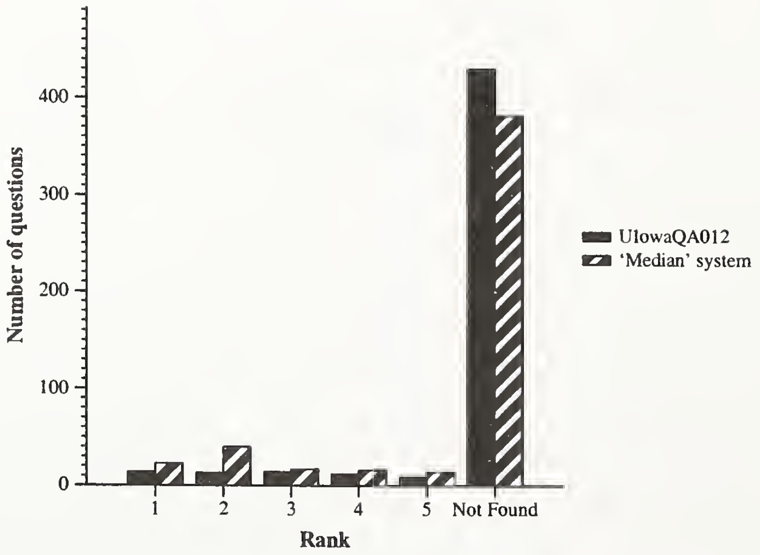
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	UIowaQA011
Num questions	492
Mean reciprocal rank (strict)	0.061
Mean reciprocal rank (lenient)	0.062
Num answers not found (strict)	435 (88.4%)
Num answers not found (lenient)	434 (88.2%)
Number of times NIL returned	19
Number of times NIL correctly returned	1
Percentage of answers system confident about	55%
Percentage of confident answers that were correct	2%



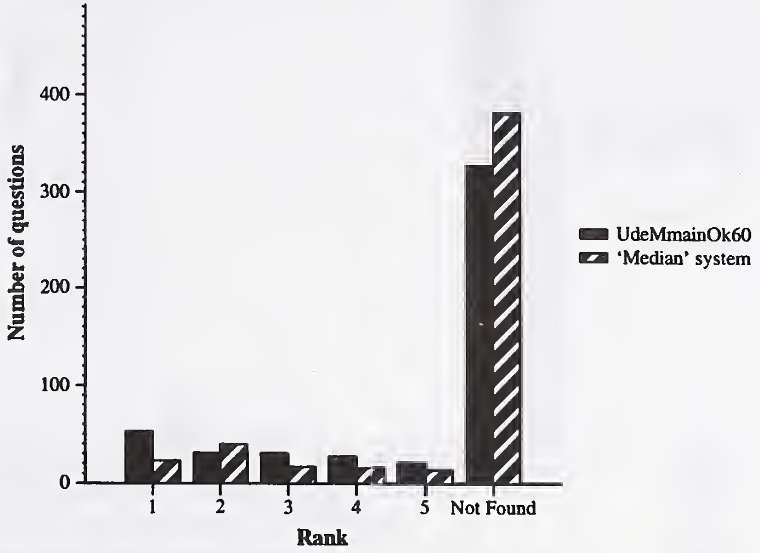
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	UIowaQA012
Num questions	492
Mean reciprocal rank (strict)	0.061
Mean reciprocal rank (lenient)	0.064
Num answers not found (strict)	430 (87.4%)
Num answers not found (lenient)	427 (86.8%)
Number of times NIL returned	19
Number of times NIL correctly returned	1
Percentage of answers system confident about	19%
Percentage of confident answers that were correct	3%



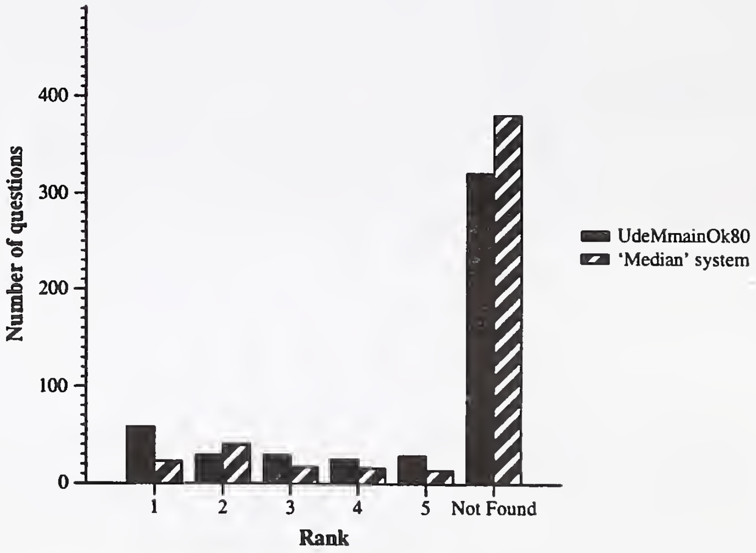
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	UdeMmainOk60
Num questions	492
Mean reciprocal rank (strict)	0.183
Mean reciprocal rank (lenient)	0.189
Num answers not found (strict)	328 (66.7%)
Num answers not found (lenient)	323 (65.7%)
Number of times NIL returned	422
Number of times NIL correctly returned	43
Percentage of answers system confident about	92%
Percentage of confident answers that were correct	11%



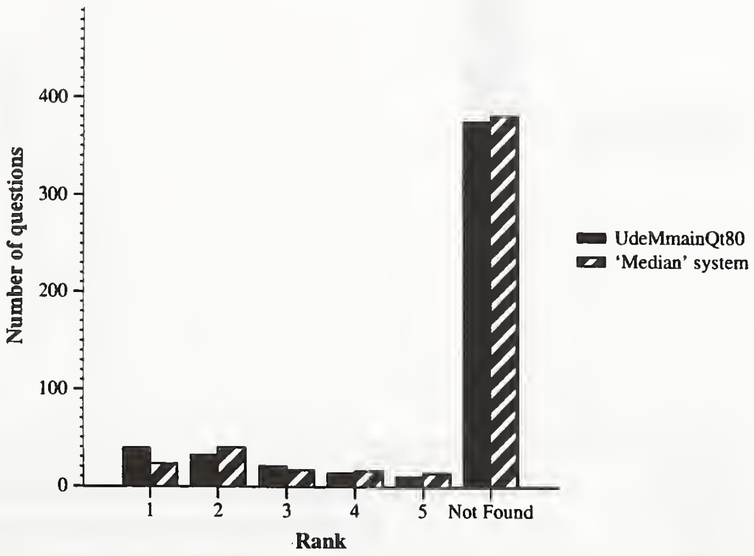
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	UdeMmainOk80
Num questions	492
Mean reciprocal rank (strict)	0.191
Mean reciprocal rank (lenient)	0.197
Num answers not found (strict)	322 (65.4%)
Num answers not found (lenient)	317 (64.4%)
Number of times NIL returned	420
Number of times NIL correctly returned	43
Percentage of answers system confident about	92%
Percentage of confident answers that were correct	12%



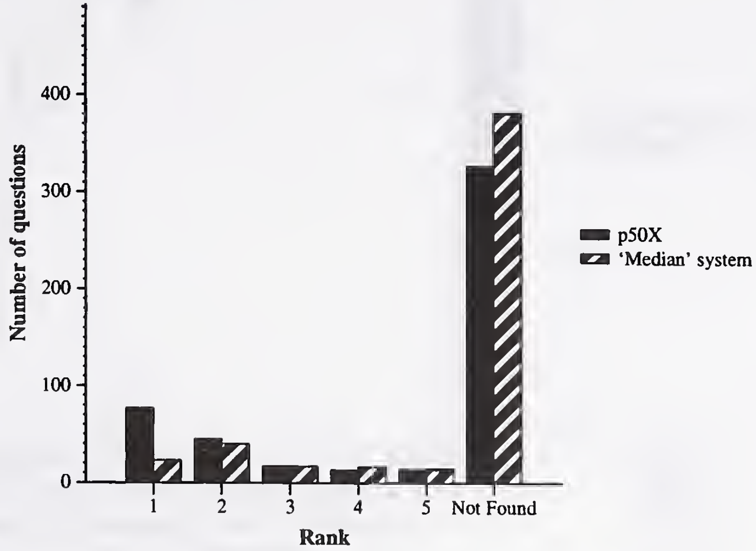
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	UdeMmainQt80
Num questions	492
Mean reciprocal rank (strict)	0.137
Mean reciprocal rank (lenient)	0.145
Num answers not found (strict)	376 (76.4%)
Num answers not found (lenient)	369 (75.0%)
Number of times NIL returned	240
Number of times NIL correctly returned	23
Percentage of answers system confident about	92%
Percentage of confident answers that were correct	8%



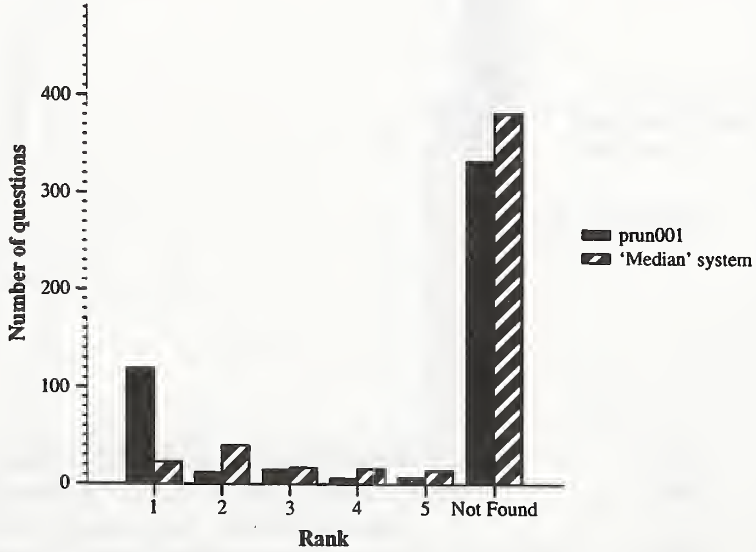
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	p50X
Num questions	492
Mean reciprocal rank (strict)	0.226
Mean reciprocal rank (lenient)	0.235
Num answers not found (strict)	327 (66.5%)
Num answers not found (lenient)	321 (65.2%)
Number of times NIL returned	43
Number of times NIL correctly returned	1
Percentage of answers system confident about	9%
Percentage of confident answers that were correct	40%



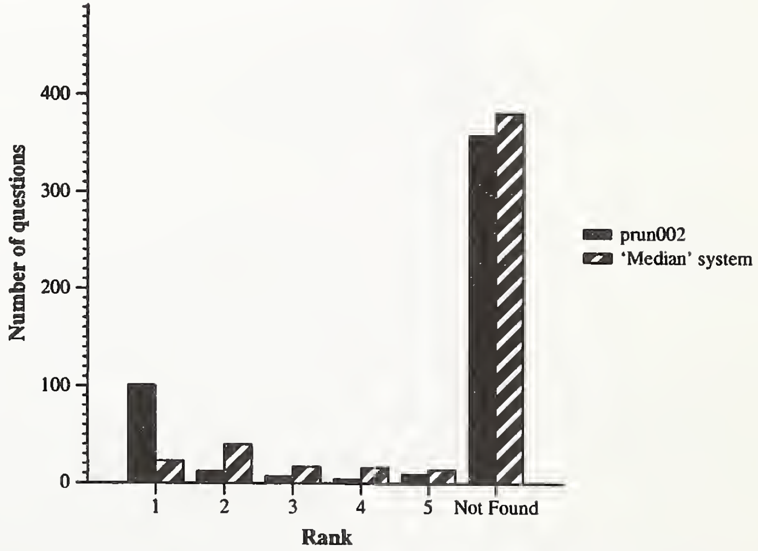
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	prun001
Num questions	492
Mean reciprocal rank (strict)	0.270
Mean reciprocal rank (lenient)	0.271
Num answers not found (strict)	333 (67.7%)
Num answers not found (lenient)	332 (67.5%)
Number of times NIL returned	201
Number of times NIL correctly returned	38
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	24%



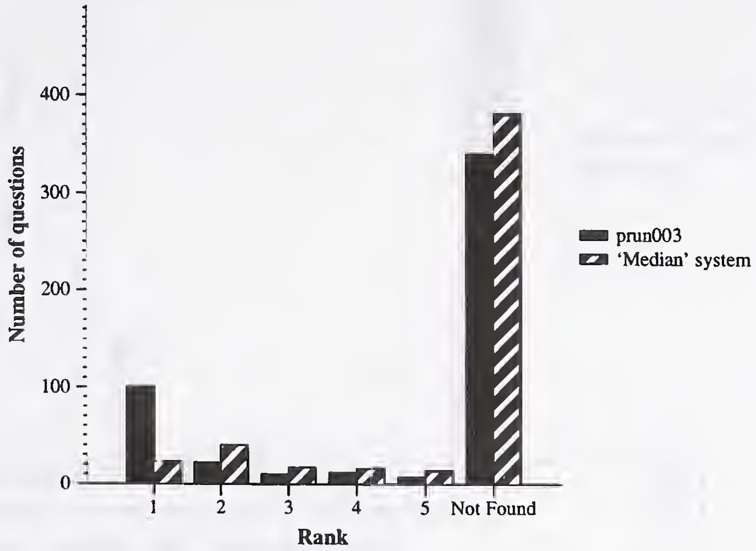
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	prun002
Num questions	492
Mean reciprocal rank (strict)	0.228
Mean reciprocal rank (lenient)	0.230
Num answers not found (strict)	359 (73.0%)
Num answers not found (lenient)	358 (72.8%)
Number of times NIL returned	221
Number of times NIL correctly returned	33
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	20%



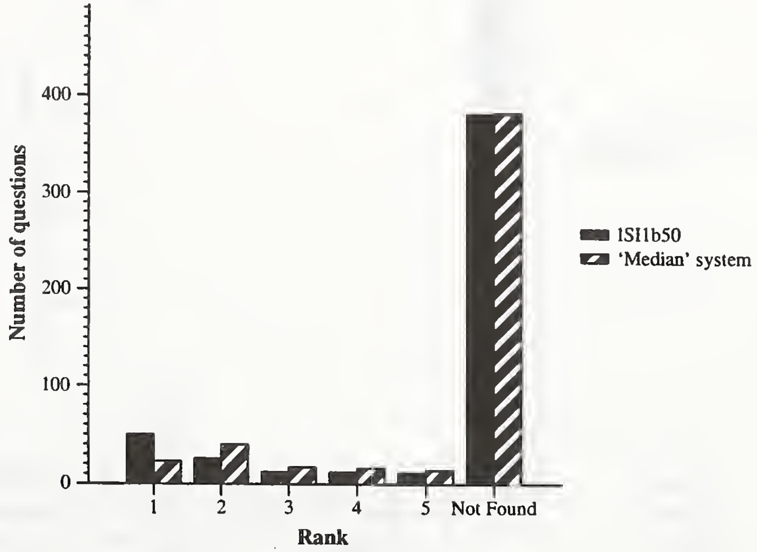
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	prun003
Num questions	492
Mean reciprocal rank (strict)	0.241
Mean reciprocal rank (lenient)	0.243
Num answers not found (strict)	341 (69.3%)
Num answers not found (lenient)	339 (68.9%)
Number of times NIL returned	125
Number of times NIL correctly returned	24
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	20%



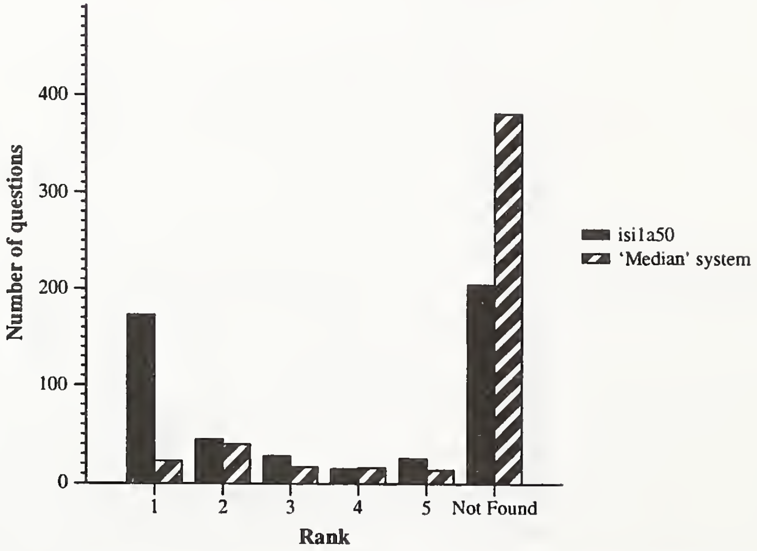
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	ISI1b50
Num questions	492
Mean reciprocal rank (strict)	0.147
Mean reciprocal rank (lenient)	0.152
Num answers not found (strict)	381 (77.4%)
Num answers not found (lenient)	377 (76.6%)
Number of times NIL returned	0
Number of times NIL correctly returned	0
Percentage cf answers system confident about	0%
Percentage cf confident answers that were correct	%



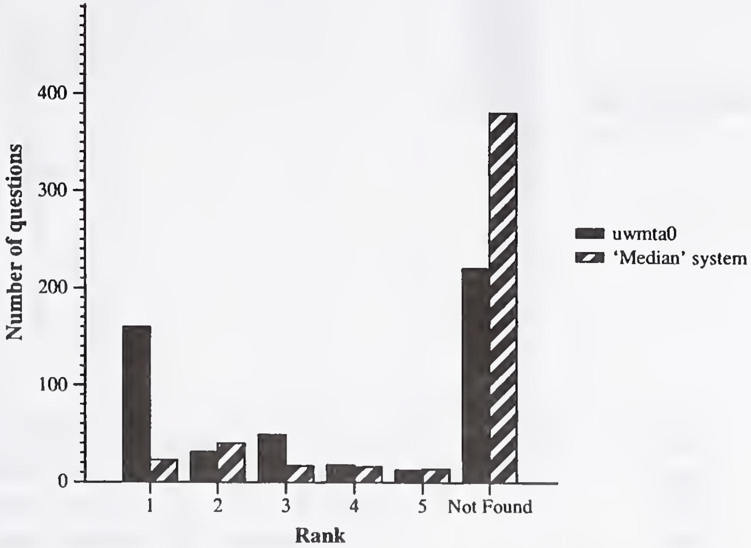
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	isi1a50
Num questions	492
Mean reciprocal rank (strict)	0.435
Mean reciprocal rank (lenient)	0.451
Num answers not found (strict)	205 (41.7%)
Num answers not found (lenient)	196 (39.8%)
Number of times NIL returned	407
Number of times NIL correctly returned	33
Percentage of answers system confident about	80%
Percentage of confident answers that were correct	38%



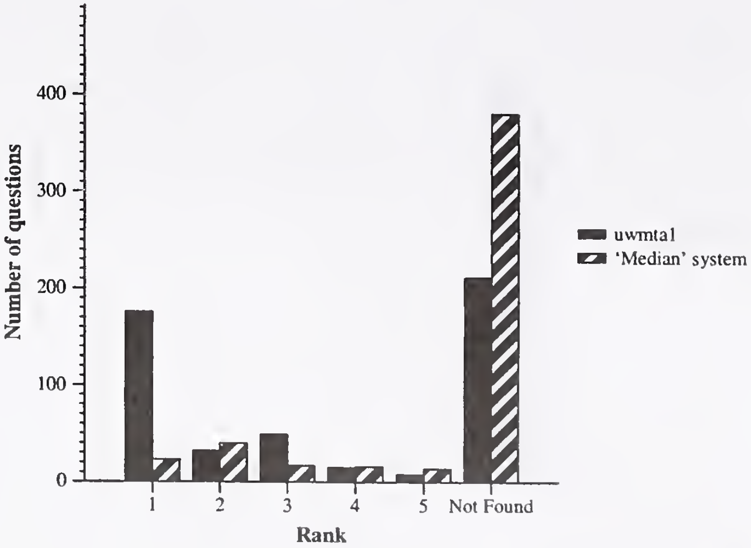
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	uwmta0
Num questions	492
Mean reciprocal rank (strict)	0.404
Mean reciprocal rank (lenient)	0.450
Num answers not found (strict)	221 (44.9%)
Num answers not found (lenient)	193 (39.2%)
Number of times NIL returned	492
Number of times NIL correctly returned	49
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	32%



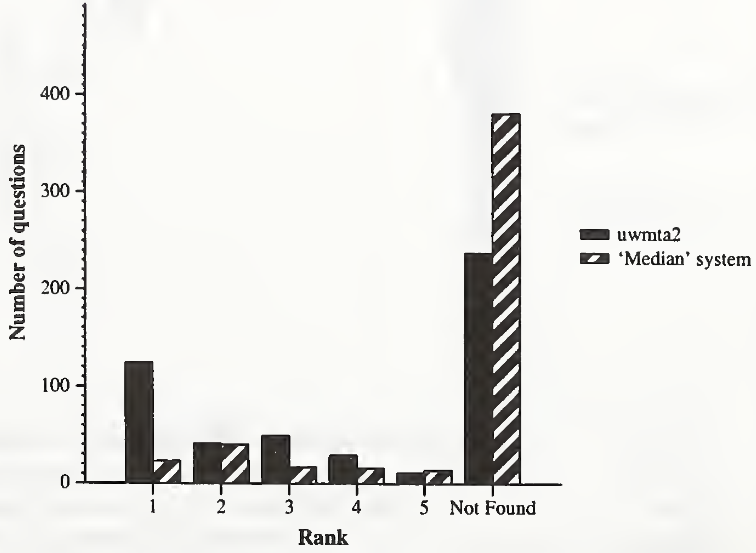
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	uwmta1
Num questions	492
Mean reciprocal rank (strict)	0.434
Mean reciprocal rank (lenient)	0.457
Num answers not found (strict)	212 (43.1%)
Num answers not found (lenient)	200 (40.7%)
Number of times NIL returned	492
Number of times NIL correctly returned	49
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	35%



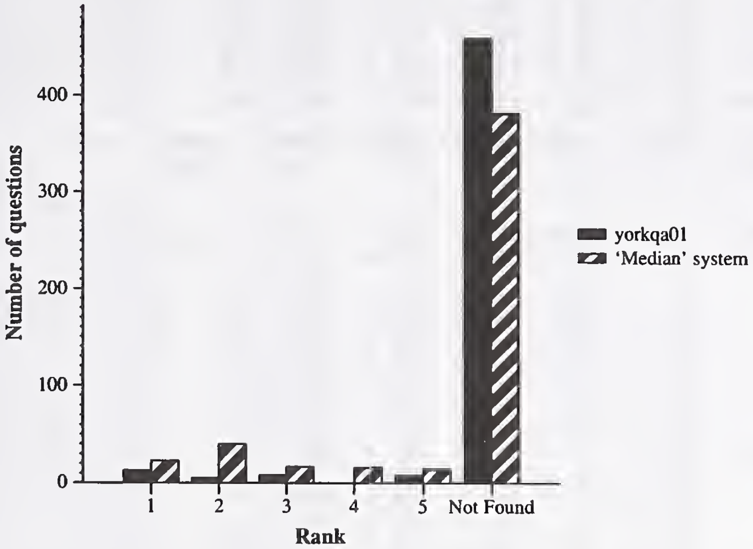
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	uwmta2
Num questions	492
Mean reciprocal rank (strict)	0.346
Mean reciprocal rank (lenient)	0.365
Num answers not found (strict)	238 (48.4%)
Num answers not found (lenient)	227 (46.1%)
Number of times NIL returned	492
Number of times NIL correctly returned	49
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	25%



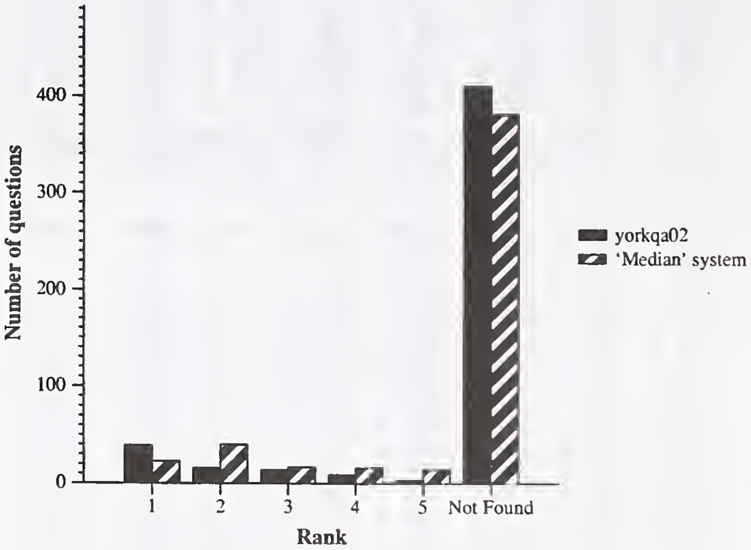
Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	yorkqa01
Num questions	492
Mean reciprocal rank (strict)	0.040
Mean reciprocal rank (lenient)	0.048
Num answers not found (strict)	459 (93.3%)
Num answers not found (lenient)	454 (92.3%)
Number of times NIL returned	124
Number of times NIL correctly returned	7
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	2%



Number of questions for which the first correct response was at a given rank

Summary Statistics	
Run ID	yorkqa02
Num questions	492
Mean reciprocal rank (strict)	0.111
Mean reciprocal rank (lenient)	0.121
Num answers not found (strict)	411 (83.5%)
Num answers not found (lenient)	403 (81.9%)
Number of times NIL returned	37
Number of times NIL correctly returned	4
Percentage of answers system confident about	100%
Percentage of confident answers that were correct	7%



Number of questions for which the first correct response was at a given rank

TREC-2001 Video Track Shot Boundary Determination Results - Precision and Recall by Topic and System for Gradual Transitions

Precision

	Ref. Trans.	Row Mean	Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15
ahf1.mpg	44	0.700	0.614	0.614	0.871	0.891	1.000	0.241	0.689	0.579	0.625	0.256	1.000	0.826	0.703	0.909	0.687
anni005.mpg	27	0.679	0.730	0.730	0.863	0.857			0.777	0.564	0.666	0.333	0.000	0.782	0.875	0.950	0.705
anni009.mpg	65	0.669	0.807	0.807	0.860	0.829	0.000		0.784	0.700	0.809	0.508	0.000	0.733	0.836	0.926	0.769
bor03.mpg	11	0.283	0.212	0.212	0.307	0.333	1.000	0.014	0.185	0.146	0.166	0.023	0.000	0.281	0.375	0.727	0.257
bor08.mpg	151	0.794	0.904	0.904	0.770	0.807			0.790	0.805	0.741	0.376	1.000	0.815	0.796	0.961	0.647
bor10.mpg	150	0.743	0.658	0.658	0.862	0.869			0.901	0.890	0.866	0.472	0.000	0.906	0.837	0.958	0.780
bor12.mpg	136	0.705	0.610	0.610	0.842	0.818			0.656	0.744	0.825	0.737	0.000	0.814	0.832	0.914	0.765
bor17.mpg	119	0.678	0.625	0.636	0.802	0.802			0.686	0.616	0.584	0.220	1.000	0.573	0.711	1.000	0.560
eall.mpg	20	0.531	0.551	0.551	0.666	0.857			0.600	0.619	0.600	0.516	0.000	0.166	0.538	0.818	0.423
nad28.mpg	116	0.555	0.538	0.538	0.636	0.636	1.000	0.078	0.615	0.556	0.588	0.221	0.000	0.713	0.744	0.924	0.531
nad31.mpg	55	0.436	0.342	0.345	0.551	0.533			0.364	0.292	0.353	0.198	1.000	0.420	0.417	0.535	0.319
nad33.mpg	26	0.389	0.280	0.280	0.422	0.425			0.347	0.315	0.206	0.060	1.000	0.382	0.307	0.555	0.480
nad53.mpg	76	0.575	0.527	0.527	0.737	0.766			0.630	0.572	0.761	0.422	0.000	0.693	0.655	0.700	0.484
nad57.mpg	23	0.620	0.653	0.653	0.680	0.730			0.580	0.500	0.612	0.419	1.000	0.593	0.562	0.647	0.434
pfm1.mpg	21	0.499	0.461	0.461	0.727	0.736			0.461	0.377	0.608	0.228	0.000	0.533	0.545	0.687	0.666
senses111.mpg	16	0.298	0.092	0.092	0.461	0.545			0.454	0.400	0.068	0.126	0.000	0.125	0.375	1.000	0.137
ydh1.mpg	52	0.620	0.593	0.593	0.675	0.694			0.400	0.436	0.536	0.347	1.000	0.660	0.698	0.812	
weighted column mean ->		0.648	0.623	0.625	0.753	0.763	0.725	0.116	0.669	0.641	0.670	0.381	0.424	0.703	0.723	0.879	0.616
			Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15

Recall

	Ref. Trans.	Row Mean	Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15
ahf1.mpg	44	0.683	0.795	0.795	0.772	0.750	0.636	0.636	0.909	0.909	0.681	0.431	0.022	0.863	0.863	0.681	0.500
anni005.mpg	27	0.609	0.703	0.703	0.703	0.666			0.777	0.814	0.444	0.518	0.000	0.666	0.777	0.703	0.444
anni009.mpg	65	0.501	0.707	0.707	0.569	0.523	0.000		0.615	0.646	0.523	0.461	0.000	0.507	0.707	0.584	0.461
bor03.mpg	11	0.660	0.909	0.909	0.727	0.727	0.363	0.363	0.909	1.000	0.545	0.272	0.000	0.818	0.818	0.727	0.818
bor08.mpg	151	0.633	0.688	0.688	0.688	0.695			0.874	0.933	0.741	0.503	0.013	0.761	0.880	0.331	0.437
bor10.mpg	150	0.687	0.706	0.706	0.753	0.713			0.920	0.920	0.866	0.506	0.000	0.906	0.893	0.613	0.426
bor12.mpg	136	0.556	0.794	0.794	0.470	0.397			0.647	0.727	0.625	0.764	0.000	0.647	0.764	0.235	0.360
bor17.mpg	119	0.511	0.504	0.529	0.647	0.647			0.495	0.512	0.697	0.436	0.268	0.394	0.705	0.453	0.352
eall.mpg	20	0.573	0.800	0.800	0.600	0.600			0.600	0.650	0.600	0.800	0.000	0.300	0.700	0.450	0.550
nad28.mpg	116	0.603	0.793	0.793	0.543	0.543	0.500	0.500	0.870	0.896	0.543	0.336	0.000	0.836	0.827	0.629	0.431
nad31.mpg	55	0.478	0.709	0.709	0.490	0.436			0.563	0.600	0.418	0.563	0.054	0.527	0.600	0.272	0.272
nad33.mpg	26	0.535	0.538	0.538	0.730	0.653			0.615	0.692	0.500	0.500	0.038	0.692	0.615	0.384	0.461
nad53.mpg	76	0.596	0.750	0.750	0.592	0.605			0.763	0.776	0.631	0.500	0.000	0.802	0.802	0.368	0.407
nad57.mpg	23	0.659	0.739	0.739	0.739	0.826			0.782	0.782	0.826	0.565	0.043	0.826	0.782	0.478	0.434
pfm1.mpg	21	0.630	0.857	0.857	0.761	0.666			0.857	0.809	0.666	0.380	0.000	0.761	0.571	0.523	0.476
senses111.mpg	16	0.336	0.437	0.437	0.375	0.375			0.312	0.375	0.187	0.812	0.000	0.375	0.375	0.062	0.250
ydh1.mpg	52	0.492	0.673	0.673	0.519	0.480			0.461	0.596	0.423	0.461	0.019	0.634	0.711	0.250	
weighted column mean ->		0.585	0.706	0.709	0.621	0.597	0.381	0.526	0.732	0.769	0.640	0.513	0.037	0.694	0.777	0.445	0.413
			Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15

TREC-2001 Video Track   Shot Boundary Determination Results - Insertion Counts by Topic and System for Cuts and Gradual Transitions

C u t s	Ref Trans.	RowMean	Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15
ancc836y_s1_01_002.mpg	0	1.000	1	1	1	1			1	1	0		1	1	1	1	2
ancc836y_s1_01_004.mpg	0	0.083	0	0	0	0			0	0	0		0	1	0	0	0
bor10.mpg	0	8.231	0	2	0	0			0	0	11	0	48	0	10	7	29
bor12.mpg	0	11.231	2	7	0	0			1	1	2	0	49	0	21	8	55
esbt501d_s1_01_001.mpg	0	0.000	0	0	0	0			0	0	0		0	0	0	0	0
esbt501d_s1_01_003.mpg	0	0.167	0	0	0	0			0	0	0		0	1	1	0	0
esbt501d_s1_01_005.mpg	0	0.167	0	0	0	0			0	0	0		1	1	0	0	0
esbt501d_s1_01_007.mpg	0	0.083	0	0	0	0			0	0	0		1	0	0	0	0
1doi909j_s1_01_001.mpg	0	0.083	0	0	0	0			0	0	0		0	1	0	0	0
1doi909j_s1_01_003.mpg	0	0.083	0	0	0	0			0	0	0		0	1	0	0	0
1doi909j_s1_07_002.mpg	0	0.333	0	0	0	0			0	0	0		1	0	1	2	0
1doi909j_s1_07_004.mpg	0	0.000	0	0	0	0			0	0	0		0	0	0	0	0
1doi909j_s1_07_006.mpg	0	0.000	0	0	0	0			0	0	0		0	0	0	0	0
1doi909j_s1_07_008.mpg	0	0.583	1	1	2	2			0	0	0		0	1	0	0	0
1doi909j_s1_07_010.mpg	0	0.667	0	0	0	0			0	0	0		8	0	0	0	0
1doi909j_s2_15_003.mpg	0	0.083	0	0	0	0			0	0	0		0	1	0	0	0
1doi915y_s1_01_001.mpg	0	1.333	0	0	2	2			0	0	0		7	2	1	0	2
1doi915y_s1_01_003.mpg	0	0.583	0	0	0	0			0	0	0		4	3	0	0	0
1doi915y_s1_01_005.mpg	0	2.917	3	5	2	2			2	2	1		12	2	2	2	0
1doi915y_s1_01_007.mpg	0	0.083	0	0	0	0			0	0	0		0	1	0	0	0
1doi915y_s1_01_009.mpg	0	0.917	0	0	0	0			0	0	0		6	1	0	0	4
nbmw628d_s1_01_002.mpg	0	5.083	2	2	3	3			2	2	4		25	8	5	3	2
weighted column mean ->		1.532	0.409	0.818	0.455	0.455			0.273	0.273	0.818	0.000	7.409	1.136	1.909	1.045	4.273
			Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15
G r a d u a l																	
ancc836y_s1_01_002.mpg	0	0.083	0	0	0	0			0	0	0		0	0	0	0	1
ancc836y_s1_01_004.mpg	0	0.250	0	0	0	0			0	0	0		0	0	0	0	3
ancc836y_s1_01_006.mpg	0	0.500	2	2	0	0			0	0	1		0	0	0	0	1
esbt501d_s1_01_001.mpg	0	0.000	0	0	0	0			0	0	0		0	0	0	0	0
esbt501d_s1_01_003.mpg	0	0.333	0	0	0	0			0	1	0		0	0	0	0	3
esbt501d_s1_01_005.mpg	0	0.000	0	0	0	0			0	0	0		0	0	0	0	0
esbt501d_s1_01_007.mpg	0	0.667	0	0	1	0			1	1	0		0	1	0	0	4
1doi874a_s1_02_002.mpg	0	2.417	0	0	3	2			2	6	10		0	0	2	0	4
1doi874a_s1_02_004.mpg	0	5.214	0	0	6	4	0	0	11	18	13		0	6	10	0	5
1doi909j_s1_01_001.mpg	0	0.083	0	0	0	0			0	0	0		0	0	0	0	1
1doi909j_s1_01_003.mpg	0	0.000	0	0	0	0			0	0	0		0	0	0	0	0
1doi909j_s1_07_002.mpg	0	0.083	0	0	0	0			0	1	0		0	0	0	0	0
1doi909j_s1_07_004.mpg	0	0.000	0	0	0	0			0	0	0		0	0	0	0	0
1doi909j_s1_07_006.mpg	0	0.000	0	0	0	0			0	0	0		0	0	0	0	0
1doi909j_s1_07_008.mpg	0	0.333	0	0	0	0			1	2	0		0	0	0	0	1
1doi909j_s1_07_010.mpg	0	2.333	0	0	2	2			4	5	3		0	4	4	0	4
1doi909j_s2_15_001.mpg	0	0.250	0	0	0	0			1	1	0		0	0	0	0	1
1doi909j_s2_15_003.mpg	0	0.250	0	0	0	0			0	1	0		0	0	0	0	2
1doi915y_s1_01_001.mpg	0	1.333	0	0	0	0			2	4	2		0	1	6	0	1
1doi915y_s1_01_003.mpg	0	1.250	0	0	2	2			1	2	3		0	1	2	0	2
1doi915y_s1_01_005.mpg	0	2.417	0	0	1	1			2	5	2		0	3	6	0	9
1doi915y_s1_01_007.mpg	0	0.167	0	0	0	0			0	0	0		0	0	0	0	2
1doi915y_s1_01_009.mpg	0	0.500	0	0	0	0			1	1	1		0	0	2	0	1
nbmw628d_s1_01_002.mpg	0	4.333	0	0	0	0			2	5	7		0	5	31	0	2
nbmw628d_s1_01_004.mpg	0	0.167	0	0	0	0			0	0	0		0	0	0	0	2
weighted column mean ->		0.919	0.080	0.080	0.600	0.440	0.000	0.000	1.120	2.120	1.680		0.000	0.840	2.520	0.000	1.960
			Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15

TREC-2001 Video Track   Shot Boundary Determination Results - Precision and Recall by Topic and System for Cuts

Precision

	Ref. Trans.	Row Mean	Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15
ahf1.mpg	63	0.919	1.000	1.000	0.983	0.983	0.887	0.887	1.000	1.000	0.921	1.000	0.710	0.852	0.787	0.926	0.852
ancc836y_s1_01_006.mpg	2	0.889	1.000	1.000	1.000	1.000			1.000	1.000	1.000		1.000	0.666	1.000	1.000	0.000
anni005.mpg	38	0.699	0.487	0.258	0.950	0.950			0.972	0.971	0.770	0.942	0.297	0.948	0.244	0.566	0.729
anni009.mpg	38	0.735	0.822	0.666	0.947	0.947	0.967		0.923	0.923	0.697	0.958	0.205	0.888	0.208	0.447	0.697
bor03.mpg	230	0.920	0.933	0.808	0.961	0.961	0.899	0.899	0.956	0.964	0.953	0.974	0.815	0.972	0.865	0.942	0.905
bor08.mpg	379	0.894	0.910	0.749	0.971	0.971			0.971	0.973	0.945	0.988	0.598	0.888	0.832	0.958	0.873
bor17.mpg	127	0.843	0.787	0.514	0.984	0.984			0.976	0.967	0.975	0.981	0.584	0.983	0.863	0.932	0.433
eal1.mpg	61	0.943	1.000	0.983	1.000	1.000			1.000	1.000	0.921	1.000	0.439	1.000	0.938	0.983	1.000
1doi874a_s1_02_002.mpg	8	0.776	1.000	1.000	1.000	1.000			0.888	0.888	1.000		0.666	0.857	0.444	0.571	0.000
1doi874a_s1_02_004.mpg	7	0.688	0.875	0.875	0.875	0.875	0.875	0.875	1.000	0.875	0.625		0.333	0.777	0.272	0.500	0.000
1doi909j_s2_15_001.mpg	1	0.292	0.000	0.000	1.000	1.000			0.000	0.000	0.000		0.500	1.000	0.000	0.000	0.000
nad28.mpg	181	0.853	0.899	0.873	0.930	0.930	0.915	0.915	0.945	0.949	0.762	0.933	0.721	0.910	0.881	0.936	0.302
nad31.mpg	187	0.834	0.947	0.866	0.939	0.939			0.962	0.967	0.900	0.960	0.607	0.944	0.698	0.839	0.273
nad33.mpg	189	0.882	0.917	0.782	0.974	0.974			0.954	0.943	0.918	0.977	0.815	0.968	0.806	0.917	0.517
nad53.mpg	83	0.876	0.965	0.965	0.987	0.987			0.987	0.987	0.797	0.946	0.562	0.987	0.922	0.965	0.331
nad57.mpg	44	0.880	0.956	0.897	0.956	0.956			0.977	0.977	0.860	0.975	0.750	0.971	0.830	0.977	0.361
nbmw628d_s1_01_004.mpg	1	0.958	1.000	1.000	1.000	1.000			1.000	1.000	1.000		1.000	0.500	1.000	1.000	1.000
pfm1.mpg	61	0.851	0.909	0.845	0.921	0.921			0.909	0.909	0.746	0.879	0.786	0.900	0.722	0.800	0.822
senses111.mpg	292	0.909	1.000	0.963	0.996	0.996			0.996	0.996	1.000	0.996	0.688	0.992	0.738	0.988	0.473
ydh1.mpg	69	0.839	0.827	0.761	0.905	0.905			0.891	0.890	0.881	0.917	0.626	0.848	0.761	0.858	
weighted column mean ->		0.876	0.917	0.816	0.965	0.965	0.908	0.903	0.965	0.965	0.907	0.970	0.660	0.940	0.788	0.915	0.602
			Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15

Recall

	Ref. Trans.	Row Mean	Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15
ahf1.mpg	63	0.961	1.000	1.000	0.968	0.968	1.000	1.000	1.000	0.968	0.936	0.888	0.936	0.920	1.000	1.000	0.825
ancc836y_s1_01_006.mpg	2	0.875	1.000	1.000	1.000	1.000			1.000	1.000	0.500		1.000	1.000	1.000	1.000	0.000
anni005.mpg	38	0.929	1.000	1.000	1.000	1.000			0.947	0.894	0.973	0.868	0.736	0.973	0.868	0.894	0.921
anni009.mpg	38	0.870	0.973	1.000	0.947	0.947	0.789		0.947	0.947	0.789	0.605	0.815	0.842	0.894	0.894	0.789
bor03.mpg	230	0.856	0.978	0.973	0.982	0.982	0.426	0.426	0.965	0.939	0.982	0.843	0.747	0.939	0.947	0.926	0.786
bor08.mpg	379	0.920	0.989	0.986	0.978	0.978			0.984	0.978	0.873	0.894	0.746	0.963	0.981	0.968	0.638
bor17.mpg	127	0.905	0.992	1.000	0.992	0.992			0.968	0.944	0.952	0.818	0.488	0.913	0.992	0.976	0.740
eal1.mpg	61	0.956	1.000	1.000	1.000	1.000			0.983	0.983	0.967	0.934	0.836	0.868	1.000	0.983	0.868
1doi874a_s1_02_002.mpg	8	0.896	1.000	1.000	1.000	1.000			1.000	1.000	1.000		1.000	0.750	1.000	1.000	0.000
1doi874a_s1_02_004.mpg	7	0.888	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.714		1.000	1.000	0.857	0.857	0.000
1doi909j_s2_15_001.mpg	1	0.333	0.000	0.000	1.000	1.000			0.000	0.000	0.000		1.000	1.000	0.000	0.000	0.000
nad28.mpg	181	0.917	0.988	0.994	0.883	0.883	0.955	0.955	0.961	0.939	0.939	0.696	0.828	0.839	0.983	0.977	0.928
nad31.mpg	187	0.892	0.967	0.973	0.909	0.909			0.973	0.941	0.866	0.775	0.754	0.903	0.951	0.925	0.748
nad33.mpg	189	0.951	0.994	0.989	0.994	0.994			0.989	0.978	0.952	0.910	0.867	0.989	0.994	0.994	0.714
nad53.mpg	83	0.962	1.000	1.000	0.987	0.987			0.987	0.975	0.951	0.855	0.915	0.975	1.000	1.000	0.879
nad57.mpg	44	0.956	1.000	1.000	1.000	1.000			1.000	1.000	0.977	0.909	0.886	0.772	1.000	1.000	0.886
nbmw628d_s1_01_004.mpg	1	1.000	1.000	1.000	1.000	1.000			1.000	1.000	1.000		1.000	1.000	1.000	1.000	1.000
pfm1.mpg	61	0.948	0.983	0.983	0.967	0.967			0.983	0.983	0.967	0.836	0.967	0.885	0.983	0.983	0.836
senses111.mpg	292	0.902	1.000	1.000	0.996	0.996			1.000	1.000	0.989	0.863	0.976	0.938	0.784	0.578	0.609
ydh1.mpg	69	0.961	0.971	0.971	0.971	0.971			0.956	0.942	0.971	0.971	0.898	0.971	0.971	0.971	
weighted column mean ->		0.915	0.988	0.988	0.969	0.969	0.714	0.709	0.979	0.965	0.935	0.847	0.815	0.927	0.946	0.908	0.738
			Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15

TREC-2001 Video Track - Tabular Overview of Topics

Each row provides: the topic's number (#); indications of whether the topic is intended for interactive (I) . automatic (A), and/or known-item (K) processing; a brief text description of the need; the number of video (Vid), image (Im), audio (Aud) examples; and the number of known-items (K-I) which satisfy the topic's information need.

#	I	A	K	Text description of needed information/clip	Vid	Im	Aud	K-I
1	i	-	k	Statue of Liberty spikes	0	1	0	10
2	-	a	-	liftoff of the Space Shuttle	4	0	0	0
3	-	a	k	vehicle traveling on the moon	1	0	0	2
4	-	a	k	mountains as prominent scenery	1	0	0	8
5	-	a	k	water skiing	1	0	0	5
6	-	a	k	scenes with a yellow boat	1	0	0	4
7	-	a	k	pink flower	0	1	0	1
8	i	a	k	the planet Jupiter	0	2	0	6
9	-	a	-	people who are water skiing	1	0	0	0
10	-	a	-	swimming pools	1	0	0	0
11	-	a	-	people on the beach	1	0	0	0
12	-	a	k	surface of Mars	1	0	1	4
13	-	a	k	speaker talking in front of the US flag	2	0	2	2
14	i	a	k	astronaut driving lunar rover over lunar surface	2	0	0	5
15	i	a	k	corn on the cob	0	1	0	4
16	i	a	k	deer with its antlers	1	1	0	2
17	i	a	k	airliner landing	0	1	0	3
18	i	a	k	John Deere tractor	0	2	0	2
19	i	a	k	lunar rover from Apollo missions	0	2	0	5
20	i	a	k	pictures of Ron Vaughn, President of Vaughncraft	0	0	0	1
21	i	a	k	pictures of Ronald Reagan speaking	0	3	3	1
22	i	a	k	pictures of Harry Hertz	0	2	0	5
23	i	a	k	images of Lou Gossett, Jr.	0	3	0	2
24	i	a	-	all other pictures of R. Lynn Bonderant	1	0	0	0
25	i	a	k	scene from Star-Wars with R2D2 and 3CPO	0	2	0	1
26	i	a	k	given summary, find the full scene sequence	1	0	0	1
27	i	a	k	biplane flying over a field	1	1	0	4
28	i	a	k	sailing boat on a beach	0	1	0	2
29	i	a	k	hot air balloon in the sky	0	1	0	5
30	i	a	k	governmental buildings looking like Capitol	1	0	0	4
31	i	a	k	waterskier behind a speed boat	0	2	0	7
32	i	a	k	chopper landing	0	0	3	1
33	i	a	k	additional shots of white fort	1	0	0	1
34	i	a	k	Ronald Reagan reading speech about Space Shuttle	0	1	0	1
35	i	a	k	Where else does this person appear?	1	0	0	11
36	i	a	k	Where else does this person appear?	1	0	0	7
37	i	-	-	other examples of rocket and shuttle launches	7	0	7	0
38	i	-	-	other examples of fires	4	0	0	0
39	i	-	-	other examples of airplanes taking off	3	0	3	0
40	-	a	-	all monologue shots	2	0	0	0
41	-	a	-	all shots with at least 8 people	2	0	0	0
42	-	a	-	all shots with David J. Nash	1	0	0	0
43	-	a	-	all shots with a specific landscape: grassland	1	0	0	0
44	-	a	-	all shots with specific camera technique: pan & tilt	1	0	0	0
45	-	a	-	other shots of cityscapes	1	0	0	0

46	-	a	-	other shots of sailing boats	1	0	0	0
47	-	a	-	clips that deal with floods	1	0	0	0
48	-	a	-	overhead zooming-in views of canyons...	8	0	0	0
49	-	a	-	Other clips from the lecture showing/explaining example graphic	9	0	0	0
50	-	a	-	Other examples of natural outdoors scenes with birds	8	0	10	0
51	-	a	-	Other examples of splashing water in natural outdoors environment	7	0	10	0
52	i	a	-	space shuttle on launch pad	6	2	0	0
53	i	a	-	pictures of the Perseus high altitude plane	0	3	0	0
54	i	a	-	clips showing Glen Canyon dam	1	0	0	0
55	i	a	-	pictures of Hoover Dam	1	0	0	0
56	i	a	-	clips of rockets taking off	2	0	0	0
57	i	a	-	footage of explosions, blasting of hillsides	1	0	0	0
58	i	a	-	additional shots of Lynn Bonderant	1	0	0	0
59	i	a	-	launch of the Space Shuttle	3	1	0	0
60	i	a	k	explosions in progress	0	1	0	60
61	i	a	-	environmental degradation	3	1	1	0
62	i	a	k	how long has Baldrige Award existed	0	0	0	3
63	-	a	-	clips of different interviewees	7	0	0	0
64	-	a	-	clips of different male interviewees	4	0	3	0
65	-	a	-	gradual shot changes	1	0	0	0
66	i	a	-	clips talking about water project	1	0	0	0
67	i	a	k	segments of aircraft X-29	2	5	0	10
68	i	a	k	segment with a(n expert) person showing the X-29	2	5	0	1
69	i	a	k	logo of Northwest Airlines	0	5	0	2
70	i	a	k	identify the producer of each item	0	0	0	3
71	i	a	k	scenes with street traffic (cars, trucks, maybe people)	0	1	0	18
72	i	a	-	other similar clips containing a rocket launch	2	0	0	0
73	-	a	-	all shots with a specific landscape: lake	2	0	0	0
74	-	a	-	all shots with specific camera technique: zoom-out	1	0	0	0

General and known-item search - groups, system ids, and systems

General search

- University of Maryland, USA
  - Sys01 - CMRS_UMD
  - Sys02 - CMRS_UMD2
- Carnegie Mellon University, USA
  - Sys03 - CMU_Aut
  - Sys04 - CMU_Int
  - Sys05 - CMUsr_Aut
  - Sys06 - CMUsr_Int
- Dublin City University,Ireland
  - Sys07 - DCUKI2001
- Fudan University, China
  - Sys08 - FDUSys1
- IBM Almaden/Watson Research Centers, USA
  - Sys09 - IBM_A_ASR
  - Sys10 - IBM_A_C+S
  - Sys11 - IBM_A_CBR
  - Sys12 - IBM_I_ASR
  - Sys13 - IBM_I_C+S
  - Sys14 - IBM_I_CBR
- Johns Hopkins University, USA
  - Sys15 - JHUAP
- CWI / Univ. of Amsterdam / Univ. of Twente / TNO, the Netherlands
  - Sys16 - Lowlands_1
  - Sys17 - Lowlands_2
  - Sys18 - Lowlands_3
  - Sys19 - Lowlands_4
  - Sys20 - Lowlands_5
- University of North Texas
  - Sys21 - UNT1

Known-item search - same as above except different numbering and sysids for CMU systems:

- Carnegie Mellon University, USA
  - Sys03 - CMU_AUT_NO_SR
  - Sys04 - CMU_AUT_SR
  - Sys05 - CMU_INT_NO_SR
  - Sys06 - CMU_INT_SR

TREC-2001 Video Track General Search Results - Partial Average Precision by Topic and System

Topic groupings: i = interactive only a+i = automatic and interactive a = automatic only

	Rel	Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15	Sys16	Sys17	Sys18	Sys19	Sys20	Sys21		
vt37	13	--	0.00	--	0.22	--	0.22	--	--	--	--	--	0.00	0.03	0.03	--	0.08	0.10	0.01	--	--	--	vt37	\
vt38	26	--	0.00	--	0.32	--	0.32	0.05	--	--	--	--	0.00	0.00	0.00	--	0.04	0.12	0.42	0.17	0.17	--	vt38	i
vt39	12	--	0.00	--	0.35	--	0.35	0.01	--	--	--	--	0.00	0.04	0.04	--	0.00	0.00	0.00	0.10	0.10	--	vt39	/
vt24	8	--	0.34	0.12	0.26	0.12	0.26	0.03	0.50	0.00	0.17	--	--	0.00	0.06	0.14	0.00	0.00	0.25	0.25	0.12	0.12	vt24	\
vt52	2	0.00	0.00	0.00	0.04	0.00	0.04	--	--	0.00	0.00	--	--	0.00	0.00	0.00	0.00	0.12	0.05	--	--	0.00	vt52	
vt53	0																						vt53	
vt54	29	--	0.08	0.05	0.12	0.05	0.12	0.07	0.00	0.02	0.07	--	--	0.07	0.07	0.03	0.03	0.18	0.16	0.09	0.09	--	vt54	
vt55	0																						vt55	
vt56	4	--	0.12	0.00	0.48	0.00	0.48	--	--	0.00	0.11	--	--	0.10	0.10	0.00	0.00	0.03	0.29	--	--	0.00	vt56	a+i
vt57	0																						vt57	
vt58	7	--	0.00	0.03	0.39	0.03	0.39	--	0.57	0.07	0.20	--	--	0.00	0.07	0.00	0.00	0.00	0.00	--	--	--	vt58	
vt59	4	--	0.00	0.02	0.00	0.02	0.00	0.03	--	0.00	0.00	--	--	0.00	0.00	0.00	0.00	0.00	0.21	0.02	0.05	0.00	vt59	
vt61	15	0.04	0.00	0.00	0.00	0.00	0.00	--	--	--	0.05	--	--	0.03	0.03	0.00	0.00	0.00	0.15	--	--	0.03	vt61	
vt66	17	--	--	0.00	0.10	0.00	0.10	0.05	--	0.14	0.07	--	--	0.07	0.14	0.00	0.00	0.00	0.02	0.02	0.02	--	vt66	
vt72	13	--	--	0.00	0.42	0.00	0.42	0.02	--	0.00	0.03	--	--	0.03	0.03	0.00	0.00	0.00	0.00	0.03	0.03	--	vt72	/
vt02	5	--	0.00	0.00	--	0.00	--	--	--	0.00	0.00	0.00	--	--	--	0.00	0.00	0.00	0.18	--	--	0.33	vt02	\
vt09	0																						vt09	
vt10	1	--	--	0.25	--	0.00	--	--	--	0.00	0.25	--	--	--	--	0.00	0.00	--	0.00	--	--	--	vt10	
vt11	4	--	--	0.00	--	0.00	--	--	--	0.12	0.50	0.50	--	--	--	0.00	0.00	0.00	0.21	--	--	--	vt11	
vt40	24	--	0.11	0.00	--	0.00	--	--	--	--	0.00	0.00	--	--	--	0.00	0.42	0.25	0.13	--	--	--	vt40	
vt41	10	--	0.00	0.36	--	0.36	--	--	--	0.00	0.02	0.02	--	--	--	0.00	0.49	0.20	0.00	--	--	0.01	vt41	
vt42	14	--	0.00	0.04	--	0.04	--	--	0.25	0.07	0.43	0.43	--	--	--	0.19	0.38	0.26	0.32	--	--	--	vt42	
vt43	4	--	0.25	0.25	--	0.25	--	--	--	0.00	0.00	0.00	--	--	--	0.02	0.30	0.00	0.02	--	--	0.00	vt43	
vt44	17	--	0.02	0.00	--	0.00	--	--	0.05	--	0.03	0.06	--	--	--	0.00	0.28	0.28	0.00	--	--	--	vt44	
vt45	1	--	0.00	0.00	--	0.00	--	--	--	0.00	0.00	0.00	--	--	--	0.00	0.00	0.00	0.08	--	--	--	vt45	a
vt46	3	--	0.00	0.00	--	0.00	--	--	--	0.06	0.12	0.00	--	--	--	0.00	0.00	0.00	0.00	--	--	--	vt46	
vt47	5	--	--	0.20	--	0.20	--	--	--	0.09	0.00	0.00	--	--	--	0.00	0.00	0.03	0.40	--	--	--	vt47	
vt48	0																						vt48	
vt49	36	--	0.47	0.08	--	0.08	--	--	--	0.01	0.53	0.53	--	--	--	0.00	0.00	0.00	0.15	--	--	0.03	vt49	
vt50	1	--	0.00	0.00	--	0.00	--	--	--	0.33	0.00	0.06	--	--	--	0.00	0.00	0.00	0.00	--	--	0.00	vt50	
vt51	27	--	0.02	0.03	--	0.03	--	--	--	0.01	0.21	0.21	--	--	--	0.00	0.05	0.05	0.17	--	--	0.03	vt51	
vt63	20	--	0.00	0.20	--	0.20	--	--	--	0.08	0.01	0.09	--	--	--	0.10	0.39	--	0.00	--	--	--	vt63	
vt64	16	--	0.01	0.26	--	0.26	--	--	--	--	0.18	0.18	--	--	--	0.00	0.38	--	0.00	--	--	--	vt64	
vt65	8	--	0.00	0.04	--	0.03	--	--	0.00	--	0.01	0.01	--	--	--	0.02	0.00	0.00	0.01	--	--	--	vt65	
vt73	18	--	0.06	0.14	--	0.14	--	--	--	0.10	0.07	0.01	--	--	--	0.00	0.12	0.12	0.13	--	--	--	vt73	
vt74	13	--	0.04	0.00	--	0.00	--	--	0.01	--	0.04	0.03	--	--	--	0.00	0.11	0.11	0.02	--	--	--	vt74	/
		2	26	28	12	28	12	7	7	22	28	18	3	12	12	28	31	28	31	7	7	11		
		0.02	0.06	0.07	0.23	0.06	0.23	0.04	0.20	0.05	0.11	0.12	0.00	0.03	0.05	0.02	0.10	0.07	0.11	0.10	0.08	0.05		
		Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15	Sys16	Sys17	Sys18	Sys19	Sys20	Sys21		

TREC-2001 Video Track   General Search Results   -   Partial Recall by Topic and System

Topic groupings: i = interactive only   a+i = automatic and interactive   a = automatic only

	Rel	Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15	Sys16	Sys17	Sys18	Sys19	Sys20	Sys21		
vt37	13	--	0.00	--	0.62	--	0.62	--	--	--	--	--	0.00	0.15	0.15	--	0.08	0.23	0.08	--	--	--	vt37\	
vt38	26	--	0.04	--	0.35	--	0.35	0.12	--	--	--	--	0.00	0.00	0.00	--	0.04	0.19	0.46	0.42	0.42	--	vt38	i
vt39	12	--	0.00	--	0.75	--	0.75	0.08	--	--	--	--	0.00	0.08	0.08	--	0.00	0.08	0.00	0.17	0.17	--	vt39/	
vt24	8	--	0.38	0.25	0.50	0.25	0.50	0.12	0.50	0.00	0.25	--	--	0.00	0.12	0.25	0.00	0.00	0.25	0.25	0.25	0.12	vt24\	
vt52	2	0.00	0.00	0.00	0.50	0.00	0.50	--	--	0.00	0.00	--	--	0.00	0.00	0.00	0.00	0.50	0.50	--	--	0.00	vt52	
vt53	0																						vt53	
vt54	29	--	0.17	0.10	0.21	0.10	0.21	0.24	0.00	0.07	0.07	--	--	0.14	0.14	0.03	0.03	0.28	0.31	0.14	0.14	--	vt54	
vt55	0																						vt55	
vt56	4	--	0.25	0.00	1.00	0.00	1.00	--	--	0.00	0.50	--	--	0.50	0.50	0.00	0.00	0.25	0.50	--	--	0.00	vt56	a+1
vt57	0																						vt57	
vt58	7	--	0.00	0.14	0.57	0.14	0.57	--	0.57	0.14	0.29	--	--	0.00	0.14	0.00	0.00	0.00	0.00	--	--	--	vt58	
vt59	4	--	0.00	0.25	0.00	0.25	0.00	0.25	--	0.00	0.00	--	--	0.00	0.00	0.00	0.00	0.00	0.50	0.25	0.50	0.00	vt59	
vt61	15	0.20	0.00	0.00	0.00	0.00	0.00	--	--	--	0.20	--	--	0.07	0.07	0.00	0.00	0.00	0.47	--	--	0.13	vt61	
vt66	17	--	--	0.00	0.18	0.00	0.18	0.24	--	0.29	0.18	--	--	0.18	0.29	0.00	0.00	0.00	0.12	0.18	0.18	--	vt66	
vt72	13	--	--	0.00	0.54	0.00	0.54	0.31	--	0.00	0.15	--	--	0.15	0.15	0.00	0.00	0.00	0.00	0.15	0.15	--	vt72/	
vt02	5	--	0.00	0.00	--	0.00	--	--	--	0.00	0.00	0.00	--	--	--	0.00	0.00	0.00	0.60	--	--	0.60	vt02\	
vt09	0																						vt09	
vt10	1	--	--	1.00	--	0.00	--	--	--	0.00	1.00	--	--	--	--	0.00	0.00	--	0.00	--	--	--	vt10	
vt11	4	--	--	0.00	--	0.00	--	--	--	0.50	0.50	0.50	--	--	--	0.00	0.00	0.00	0.50	--	--	--	vt11	
vt40	24	--	0.12	0.00	--	0.00	--	--	--	--	0.00	0.00	--	--	--	0.00	0.58	0.33	0.25	--	--	--	vt40	
vt41	10	--	0.00	0.50	--	0.50	--	--	--	0.00	0.10	0.10	--	--	--	0.00	0.60	0.20	0.00	--	--	0.10	vt41	
vt42	14	--	0.00	0.29	--	0.29	--	--	0.43	0.07	0.57	0.57	--	--	--	0.36	0.57	0.43	0.57	--	--	--	vt42	
vt43	4	--	0.25	0.25	--	0.25	--	--	--	0.00	0.00	0.00	--	--	--	0.25	0.50	0.00	0.25	--	--	0.00	vt43	
vt44	17	--	0.06	0.00	--	0.00	--	--	0.12	--	0.06	0.12	--	--	--	0.06	0.59	0.59	0.06	--	--	--	vt44	
vt45	1	--	0.00	0.00	--	0.00	--	--	--	0.00	0.00	0.00	--	--	--	0.00	0.00	0.00	1.00	--	--	--	vt45	a
vt46	3	--	0.00	0.00	--	0.00	--	--	--	0.33	0.67	0.00	--	--	--	0.00	0.00	0.00	0.00	--	--	--	vt46	
vt47	5	--	--	0.20	--	0.20	--	--	--	0.40	0.00	0.00	--	--	--	0.00	0.00	0.20	0.40	--	--	--	vt47	
vt48	0																						vt48	
vt49	36	--	0.50	0.22	--	0.22	--	--	--	0.06	0.58	0.58	--	--	--	0.00	0.00	0.00	0.28	--	--	0.03	vt49	
vt50	1	--	0.00	0.00	--	0.00	--	--	--	1.00	0.00	1.00	--	--	--	0.00	0.00	0.00	0.00	--	--	0.00	vt50	
vt51	27	--	0.11	0.15	--	0.15	--	--	--	0.07	0.26	0.26	--	--	--	0.00	0.15	0.15	0.30	--	--	0.07	vt51	
vt63	20	--	0.05	0.30	--	0.30	--	--	--	0.10	0.10	0.15	--	--	--	0.25	0.60	--	0.00	--	--	--	vt63	
vt64	16	--	0.06	0.38	--	0.38	--	--	--	--	0.25	0.25	--	--	--	0.00	0.62	--	0.00	--	--	--	vt64	
vt65	8	--	0.00	0.38	--	0.25	--	--	0.00	--	0.25	0.25	--	--	--	0.25	0.00	0.00	0.12	--	--	--	vt65	
vt73	18	--	0.17	0.28	--	0.28	--	--	--	0.22	0.17	0.06	--	--	--	0.00	0.22	0.22	0.17	--	--	--	vt73	
vt74	13	--	0.08	0.08	--	0.00	--	--	0.08	--	0.23	0.15	--	--	--	0.00	0.38	0.38	0.15	--	--	--	vt74/	
		2	26	28	12	28	12	7	7	22	28	18	3	12	12	28	31	28	31	7	7	11		
		0.10	0.09	0.17	0.43	0.13	0.43	0.19	0.24	0.15	0.23	0.22	0.00	0.11	0.14	0.05	0.16	0.14	0.25	0.22	0.26	0.10		
		Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15	Sys16	Sys17	Sys18	Sys19	Sys20	Sys21		

TREC-2001 Video Track General Search Results - Precision at 20 by Topic and System

Topic groupings: i = interactive only a+i = automatic and interactive a = automatic only

	Rel	Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15	Sys16	Sys17	Sys18	Sys19	Sys20	Sys21	
vt37	13	--	0.05	--	0.90	--	0.90	--	--	--	--	--	0.15	0.45	0.45	--	0.05	0.35	0.75	--	--	--	vt37\
vt38	26	--	0.15	--	0.95	--	0.95	0.30	--	--	--	--	0.10	0.00	0.00	--	0.05	0.30	0.70	0.60	0.60	--	vt38 i
vt39	12	--	0.05	--	0.95	--	0.95	0.60	--	--	--	--	0.00	0.25	0.25	--	0.05	0.05	0.00	0.20	0.20	--	vt39/
vt24	8	--	0.15	0.25	0.30	0.25	0.30	0.20	0.30	0.00	0.10	--	--	0.15	0.20	0.10	0.00	0.00	0.15	0.10	0.25	0.05	vt24\
vt52	2	0.00	0.05	0.00	0.55	0.00	0.55	--	--	0.00	0.10	--	--	0.20	0.20	0.00	0.00	0.35	0.55	--	--	0.00	vt52
vt53	0																						vt53
vt54	29	--	0.25	0.20	0.40	0.20	0.40	0.70	0.00	0.10	0.10	--	--	0.25	0.30	0.10	0.05	0.40	0.35	0.20	0.20	--	vt54
vt55	0																						vt55
vt56	4	--	0.10	0.05	0.75	0.05	0.75	--	--	0.00	0.35	--	--	0.45	0.45	0.00	0.25	0.45	0.75	--	--	0.05	vt56 a+i
vt57	0																						vt57
vt58	7	--	0.10	0.20	0.25	0.20	0.25	--	0.30	0.05	0.10	--	--	0.15	0.20	0.10	0.10	0.00	0.00	--	--	--	vt58
vt59	4	--	0.05	0.05	0.65	0.05	0.65	0.50	--	0.05	0.30	--	--	0.25	0.30	0.00	0.10	0.15	0.80	0.15	0.20	0.05	vt59
vt61	15	0.30	0.40	0.25	0.60	0.25	0.60	--	--	--	0.25	--	--	0.20	0.20	0.00	0.00	0.15	1.00	--	--	0.40	vt61
vt66	17	--	--	0.20	0.25	0.20	0.25	0.50	--	0.55	0.15	--	--	0.15	0.50	0.05	0.10	0.70	0.55	0.55	0.55	--	vt66
vt72	13	--	--	0.00	0.70	0.05	0.65	0.95	--	0.00	0.45	--	--	0.45	0.45	0.00	0.00	0.00	0.10	0.10	0.10	--	vt72/
vt02	5	--	0.20	0.00	--	0.00	--	--	--	0.00	0.25	0.25	--	--	--	0.00	0.00	0.00	0.90	--	--	0.35	vt02\
vt09	0																						vt09
vt10	1	--	--	0.05	--	0.00	--	--	--	0.00	0.05	--	--	--	--	0.00	0.00	--	0.00	--	--	--	vt10
vt11	4	--	--	0.00	--	0.00	--	--	--	0.10	0.10	0.10	--	--	--	0.00	0.00	0.00	0.10	--	--	--	vt11
vt40	24	--	0.25	0.45	--	0.45	--	--	--	--	0.15	0.15	--	--	--	0.00	1.00	0.65	0.40	--	--	--	vt40
vt41	10	--	0.10	0.45	--	0.45	--	--	--	0.00	0.30	0.30	--	--	--	0.00	0.35	0.10	0.15	--	--	0.45	vt41
vt42	14	--	0.00	0.35	--	0.35	--	--	0.25	0.05	0.45	0.45	--	--	--	0.25	0.50	0.35	0.50	--	--	--	vt42
vt43	4	--	0.15	0.45	--	0.45	--	--	--	0.00	0.40	0.40	--	--	--	0.10	0.20	0.00	0.05	--	--	0.10	vt43
vt44	17	--	0.30	0.05	--	0.10	--	--	0.80	--	0.10	0.15	--	--	--	0.15	0.90	0.85	0.25	--	--	--	vt44
vt45	1	--	0.15	0.25	--	0.25	--	--	--	0.00	0.15	0.15	--	--	--	0.25	0.10	0.15	0.25	--	--	--	vt45 a
vt46	3	--	0.00	0.00	--	0.00	--	--	--	0.10	0.15	0.15	--	--	--	0.00	0.00	0.00	0.00	--	--	--	vt46
vt47	5	--	--	0.30	--	0.30	--	--	--	0.45	0.05	0.05	--	--	--	0.00	0.20	0.05	0.20	--	--	--	vt47
vt48	0																						vt48
vt49	36	--	1.00	0.40	--	0.40	--	--	--	0.20	1.00	1.00	--	--	--	0.00	0.00	0.00	1.00	--	--	0.15	vt49
vt50	1	--	0.10	0.35	--	0.35	--	--	--	0.15	0.20	0.30	--	--	--	0.00	0.25	0.50	0.80	--	--	0.55	vt50
vt51	27	--	0.35	0.25	--	0.25	--	--	--	0.25	0.40	0.45	--	--	--	0.00	0.30	0.20	0.65	--	--	0.30	vt51
vt63	20	--	0.20	0.50	--	0.50	--	--	--	0.10	0.25	0.30	--	--	--	0.30	0.90	--	0.00	--	--	--	vt63
vt64	16	--	0.15	0.35	--	0.35	--	--	--	--	0.20	0.20	--	--	--	0.00	0.60	--	0.00	--	--	--	vt64
vt65	8	--	0.10	0.35	--	0.30	--	--	0.65	--	0.10	0.10	--	--	--	0.05	0.25	0.05	0.15	--	--	--	vt65
vt73	18	--	0.25	0.25	--	0.30	--	--	--	0.20	0.30	0.30	--	--	--	0.00	0.25	0.25	0.20	--	--	--	vt73
vt74	13	--	0.25	0.25	--	0.20	--	--	0.95	--	0.20	0.15	--	--	--	0.10	1.00	1.00	0.15	--	--	--	vt74/
		2	25	28	12	28	12	7	7	22	28	18	3	12	12	28	31	28	31	7	7	11	
		0.15	0.19	0.22	0.60	0.22	0.60	0.54	0.46	0.11	0.24	0.27	0.08	0.25	0.29	0.06	0.24	0.25	0.37	0.27	0.30	0.22	
		Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15	Sys16	Sys17	Sys18	Sys19	Sys20	Sys21	

TREC-2001 Video Track General Search Results - Precision@ResultSetSize by Topic and System

Topic groupings: i = interactive only a+i = automatic and interactive a = automatic only

	Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15	Sys16	Sys17	Sys18	Sys19	Sys20	Sys21		
vt37	--	--	--	1.00	--	1.00	--	--	--	--	--	0.23	0.45	0.45	--	0.05	0.35	0.75	--	--	--	vt37\	
vt38	--	--	--	0.95	--	0.95	0.60	--	--	--	--	0.33	0.00	0.00	--	0.05	0.30	0.70	0.60	0.60	--	vt38	i
vt39	--	--	--	1.00	--	1.00	1.00	--	--	--	--	0.00	0.25	0.25	--	0.05	0.05	0.00	0.20	0.20	--	vt39/	
vt24	0.15	--	0.25	1.00	0.25	1.00	1.00	0.35	0.00	--	0.10	--	0.20	0.15	0.13	0.00	0.00	0.15	0.10	0.25	0.50	vt24\	
vt52	0.05	0.00	0.00	0.78	0.00	0.78	--	--	0.00	--	0.10	--	0.20	0.20	0.00	0.00	0.35	0.55	--	--	0.00	vt52	
vt53	--	0.00	0.10	1.00	0.10	1.00	0.57	--	0.10	--	0.05	--	0.35	0.00	0.06	0.10	0.00	0.00	0.15	0.15	--	vt53	
vt54	0.25	--	0.20	0.88	0.20	0.88	0.77	1.00	0.16	--	0.10	--	0.30	0.25	0.13	0.05	0.40	0.35	0.20	0.20	--	vt54	
vt55	0.20	--	0.15	0.84	0.20	0.84	--	--	0.42	--	0.05	--	0.55	0.40	0.06	0.10	0.20	0.35	--	--	0.00	vt55	
vt56	0.10	--	0.05	0.93	0.05	0.93	--	--	0.00	--	0.35	--	0.45	0.45	0.00	0.25	0.45	0.75	--	--	0.50	vt56	a+i
vt57	0.15	--	0.25	1.00	0.25	1.00	--	--	0.00	--	0.10	--	0.05	0.00	0.26	0.15	0.05	0.05	--	--	0.50	vt57	
vt58	0.10	--	0.20	1.00	0.20	1.00	--	0.35	0.50	--	0.10	--	0.20	0.15	0.13	0.10	0.00	0.00	--	--	--	vt58	
vt59	0.05	--	0.05	0.92	0.05	0.92	1.00	--	0.05	--	0.30	--	0.30	0.25	0.00	0.10	0.15	0.80	0.15	0.20	0.25	vt59	
vt61	0.40	0.30	0.25	0.92	0.25	0.92	--	--	--	--	0.25	--	0.20	0.20	0.00	0.00	0.15	1.00	--	--	0.88	vt61	
vt66	--	--	0.20	1.00	0.20	1.00	0.90	--	0.62	--	0.15	--	0.62	0.15	0.06	0.10	0.70	0.55	0.55	0.55	--	vt66	
vt72	--	--	0.00	1.00	0.05	0.92	0.90	--	0.00	--	0.45	--	0.45	0.45	0.00	0.00	0.00	0.10	0.10	0.10	--	vt72 /	
vt02	0.20	--	0.00	--	0.00	--	--	--	0.00	0.25	0.25	--	--	--	0.00	0.00	0.00	0.90	--	--	0.77	vt02\	
vt09	--	--	0.00	--	0.00	--	--	--	0.00	0.00	0.00	--	--	--	0.00	0.00	0.00	0.00	--	--	--	vt09	
vt10	--	--	0.05	--	0.00	--	--	--	0.00	--	0.05	--	--	--	0.00	0.00	--	0.00	--	--	--	vt10	
vt11	--	--	0.00	--	0.00	--	--	--	0.11	0.10	0.10	--	--	--	0.00	0.00	0.00	0.10	--	--	--	vt11	
vt40	0.25	--	0.45	--	0.45	--	--	--	--	0.15	0.15	--	--	--	0.00	1.00	1.00	0.40	--	--	--	vt40	
vt41	0.10	--	0.45	--	0.45	--	--	--	0.00	0.30	0.30	--	--	--	0.00	1.00	1.00	0.15	--	--	0.75	vt41	
vt42	0.00	--	0.35	--	0.35	--	--	0.25	0.12	0.45	0.45	--	--	--	0.33	0.50	0.35	0.50	--	--	--	vt42	
vt43	0.15	--	0.45	--	0.45	--	--	--	0.00	0.40	0.40	--	--	--	0.13	0.20	0.00	0.05	--	--	0.40	vt43	
vt44	0.30	--	0.25	--	0.50	--	--	0.85	--	0.15	0.10	--	--	--	0.20	0.90	0.85	0.25	--	--	--	vt44	
vt45	0.15	--	0.25	--	0.25	--	--	--	0.00	0.15	0.15	--	--	--	0.33	0.10	0.15	0.25	--	--	--	vt45	
vt46	0.00	--	0.00	--	0.00	--	--	--	0.33	0.15	0.15	--	--	--	0.00	0.00	0.00	0.00	--	--	--	vt46	a
vt47	--	--	0.30	--	0.30	--	--	--	0.90	0.05	0.05	--	--	--	0.00	0.20	0.11	0.20	--	--	--	vt47	
vt48	0.55	--	0.55	--	0.60	--	--	--	0.00	0.05	0.10	--	--	--	--	0.10	0.30	0.60	--	--	0.75	vt48	
vt49	1.00	--	0.40	--	0.40	--	--	--	0.44	1.00	1.00	--	--	--	0.00	0.00	0.00	1.00	--	--	0.60	vt49	
vt50	0.10	--	0.35	--	0.35	--	--	--	0.50	0.30	0.20	--	--	--	0.00	0.25	0.50	0.80	--	--	0.91	vt50	
vt51	0.35	--	0.25	--	0.25	--	--	--	0.25	0.45	0.40	--	--	--	0.00	0.30	0.20	0.65	--	--	0.85	vt51	
vt63	0.20	--	0.50	--	0.50	--	--	--	0.40	0.30	0.25	--	--	--	0.40	0.90	--	0.00	--	--	--	vt63	
vt64	0.15	--	0.35	--	0.35	--	--	--	--	0.20	0.20	--	--	--	0.00	0.60	--	0.00	--	--	--	vt64	
vt65	0.10	--	0.35	--	0.30	--	--	0.65	--	0.10	0.10	--	--	--	0.06	0.25	0.05	0.15	--	--	--	vt65	
vt73	0.25	--	0.25	--	0.30	--	--	--	0.36	0.30	0.30	--	--	--	0.00	0.25	0.25	0.20	--	--	--	vt73	
vt74	0.25	--	0.25	--	0.20	--	--	1.00	--	0.15	0.20	--	--	--	0.13	1.00	1.00	0.15	--	--	--	vt74 /	
	26	3	33	15	33	15	8	7	27	20	33	3	15	15	32	36	33	36	8	8	14		
	0.21	0.10	0.23	0.95	0.24	0.94	0.84	0.64	0.19	0.25	0.21	0.19	0.30	0.22	0.08	0.24	0.27	0.35	0.26	0.28	0.55		
	Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15	Sys16	Sys17	Sys18	Sys19	Sys20	Sys21		

TREC-2001 Video Track Known-Item Search Results - Average Prevision by Topic and System

Topic groupings: i = interactive only a+i = automatic and interactive a = automatic only

Example: Intersection of known item (KI) and result item (RI) must be at least 0.666 of KI and 0.333 of RI

	Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15	Sys16	Sys17	Sys18	Sys19	Sys20	Sys21		
vt01	--	--	--	--	--	0.263	0.000	--	--	--	--	0.000	0.058	0.058	--	0.000	0.000	0.058	0.000	0.058	--	vt01	i
vt08	--	0.000	0.000	0.000	0.217	0.217	0.167	--	0.000	0.007	--	--	0.007	0.007	0.000	0.000	0.000	0.000	0.083	0.083	--	vt08	\
vt14	0.005	--	0.333	0.333	1.000	1.000	--	--	0.000	0.000	--	--	0.591	0.591	0.000	0.008	0.011	0.002	--	--	0.000	vt14	
vt15	--	0.003	0.000	0.000	0.500	0.750	0.036	--	0.000	0.003	--	--	0.292	0.009	0.000	0.000	0.250	0.250	0.000	0.000	0.000	vt15	
vt16	0.000	0.000	0.000	0.000	1.000	1.000	1.000	--	0.000	0.000	--	--	0.500	0.500	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt16	
vt17	--	0.000	0.000	0.000	0.667	0.667	--	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.009	0.000	--	--	0.000	vt17	
vt18	--	0.000	0.000	0.000	0.500	0.500	0.500	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt18	
vt19	--	0.000	0.000	0.000	1.000	1.000	0.100	--	0.000	0.000	--	--	0.591	0.591	0.000	0.000	0.000	0.008	0.720	0.720	--	vt19	
vt20	--	--	1.000	1.000	1.000	1.000	--	1.000	0.000	0.050	--	--	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	--	vt20	
vt21	--	0.000	0.000	0.000	0.000	0.000	--	0.000	0.000	0.000	--	--	0.000	0.000	0.000	0.000	--	0.000	--	--	--	vt21	
vt22	--	0.000	0.200	0.200	1.000	1.000	0.800	0.330	0.000	0.000	--	--	1.000	1.000	0.000	0.021	0.068	0.003	0.000	1.000	--	vt22	
vt23	--	0.000	0.500	0.500	1.000	1.000	0.500	0.500	0.000	0.000	--	--	0.000	0.000	0.000	0.500	0.000	0.000	0.000	0.611	--	vt23	
vt25	--	0.000	0.000	0.000	1.000	1.000	0.000	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt25	
vt26	0.000	--	0.000	0.000	0.500	0.500	--	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	--	--	--	vt26	
vt27	0.000	0.000	0.000	0.000	1.000	1.000	0.000	--	0.000	0.057	--	--	0.057	0.057	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt27	
vt28	--	0.000	0.500	0.500	0.833	0.833	0.000	--	0.000	0.000	--	--	0.049	0.049	0.000	0.042	0.000	0.000	0.000	0.000	0.000	vt28	
vt29	--	0.000	0.000	0.000	0.550	0.550	0.000	--	0.025	0.011	--	--	0.025	0.025	0.000	0.000	0.000	0.012	0.000	0.000	0.000	vt29	a+i
vt30	0.000	--	0.000	0.000	0.000	0.250	0.250	--	0.000	0.000	--	--	0.417	0.417	0.000	0.000	0.000	0.012	0.000	0.000	--	vt30	
vt31	--	0.000	0.143	0.143	0.274	0.274	0.143	--	0.000	0.002	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt31	
vt32	--	--	--	0.000	0.000	0.000	0.250	--	0.000	0.000	--	--	0.000	0.000	0.000	--	--	0.000	0.000	0.000	0.000	vt32	
vt33	0.000	--	0.000	0.000	0.000	0.000	0.000	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	--	vt33	
vt34	--	0.000	0.000	0.000	1.000	1.000	--	1.000	0.000	0.000	--	--	1.000	1.000	0.000	0.000	0.000	0.015	--	--	--	vt34	
vt35	0.071	--	0.433	0.433	0.818	0.818	--	0.055	0.000	0.577	--	--	0.526	0.526	0.107	0.000	0.000	0.004	--	--	0.000	vt35	
vt36	0.002	--	0.684	0.684	0.714	0.714	--	0.143	0.000	0.561	--	--	0.549	0.549	0.507	0.000	0.031	0.004	--	--	--	vt36	
vt60	--	0.003	0.001	0.001	0.137	0.137	0.013	--	0.033	0.005	--	--	0.005	0.043	0.000	0.000	0.000	0.000	0.001	0.001	0.002	vt60	
vt62	--	--	0.000	0.006	0.083	0.083	0.000	0.667	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	--	vt62	
vt67	0.000	0.000	0.100	0.100	1.000	1.000	0.300	--	0.005	0.100	--	--	0.100	0.090	0.000	0.000	0.000	0.000	0.000	0.000	0.073	vt67	
vt68	0.000	0.000	0.000	0.000	1.000	1.000	--	--	0.000	0.000	--	--	0.000	0.500	0.000	0.000	0.000	0.000	--	--	0.000	vt68	
vt69	--	0.000	0.000	0.000	1.000	1.000	1.000	0.000	0.000	0.000	--	--	0.500	0.000	0.000	0.000	0.000	0.000	0.000	0.000	--	vt69	
vt70	--	--	0.000	0.000	0.037	0.037	--	0.333	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	--	--	--	vt70	
vt71	--	0.000	0.000	0.000	0.273	0.273	0.050	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	--	vt71	/
vt03	0.000	--	1.000	1.000	--	--	--	--	0.000	0.500	0.500	--	--	--	0.042	0.528	0.500	0.500	--	--	0.000	vt03	\
vt04	0.030	--	0.000	0.000	--	--	--	--	0.000	0.000	0.000	--	--	--	0.000	0.000	0.062	0.096	--	--	0.000	vt04	
vt05	0.000	--	0.200	0.200	--	--	--	--	0.000	0.011	0.018	--	--	--	0.000	0.000	0.000	0.000	--	--	0.000	vt05	
vt06	0.036	--	0.265	0.265	--	--	--	--	0.000	0.000	0.000	--	--	--	0.250	0.292	0.258	0.250	--	--	0.000	vt06	a
vt07	--	0.000	0.000	0.000	--	--	--	--	0.000	0.000	0.000	--	--	--	0.000	0.000	--	0.000	--	--	--	vt07	
vt12	0.000	--	0.000	0.000	--	--	--	--	0.000	0.000	0.000	--	--	--	0.000	0.000	0.000	0.010	--	--	0.000	vt12	
vt13	0.000	--	0.250	0.250	--	--	--	--	0.000	0.167	0.167	--	--	--	0.000	0.083	0.083	0.050	--	--	0.000	vt13	/
	16	21	36	37	30	31	21	10	37	37	7	1	31	31	37	37	35	38	22	22	21		
	0.009	0.000	0.156	0.152	0.603	0.609	0.243	0.403	0.002	0.055	0.098	0.000	0.234	0.226	0.024	0.040	0.036	0.034	0.037	0.112	0.004		
	Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15	Sys16	Sys17	Sys18	Sys19	Sys20	Sys21		

TREC-2001 Video Track Known-Item Search Results - Recall by Topic and System

Topic groupings: i = interactive only a+i = automatic and interactive a = automatic only

Example: Intersection of known item (KI) and result item (RI) must be at least 0.666 of KI and 0.333 of RI

	Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15	Sys16	Sys17	Sys18	Sys19	Sys20	Sys21			
vt01	--	--	--	--	--	0.500	0.000	--	--	--	--	0.000	0.200	0.200	--	0.000	0.000	0.200	0.000	0.200	--	vt01		i
vt08	--	0.00)	0.000	0.000	0.666	0.666	0.166	--	0.000	0.166	--	--	0.166	0.166	0.000	0.000	0.000	0.000	0.166	0.166	--	vt08	\	
vt14	0.200	--	0.400	0.400	1.000	1.000	--	--	0.000	0.000	--	--	1.000	1.000	0.000	0.400	0.200	0.200	--	--	0.000	vt14	|	
vt15	--	0.25)	0.000	0.000	0.500	0.750	0.250	--	0.000	0.250	--	--	0.500	0.250	0.000	0.000	0.250	0.250	0.000	0.000	0.000	vt15	|	
vt16	0.000	0.00)	0.000	0.000	1.000	1.000	1.000	--	0.000	0.000	--	--	0.500	0.500	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt16	|	
vt17	--	0.00)	0.000	0.000	0.666	0.666	--	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.333	0.000	--	--	0.000	vt17	|	
vt18	--	0.00)	0.000	0.000	0.500	0.500	0.500	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt18	|	
vt19	--	0.00)	0.000	0.000	1.000	1.000	0.200	--	0.000	1.000	--	--	1.000	1.000	0.000	0.000	0.000	0.200	1.000	1.000	--	vt19	|	
vt20	--	--	1.000	1.000	1.000	1.000	--	1.000	0.000	1.000	--	--	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	--	vt20	|	
vt21	--	0.00)	0.000	0.000	0.000	0.000	--	0.000	0.000	0.000	--	--	0.000	0.000	0.000	0.000	--	0.000	--	--	--	vt21	|	
vt22	--	0.00)	0.200	0.200	1.000	1.000	0.800	0.600	0.000	1.000	--	--	1.000	1.000	0.000	0.600	0.400	0.200	0.000	1.000	--	vt22	|	
vt23	--	0.00)	0.500	0.500	1.000	1.000	0.500	0.500	0.000	0.000	--	--	0.000	0.000	0.000	0.500	0.000	0.000	0.000	1.000	--	vt23	|	
vt25	--	0.00)	0.000	0.000	1.000	1.000	0.000	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt25	|	
vt26	0.000	--	0.000	0.000	1.000	1.000	--	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	--	--	--	vt26	|	
vt27	0.000	0.00)	0.000	0.000	1.000	1.000	0.000	--	0.000	0.500	--	--	0.500	0.500	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt27	|	
vt28	--	0.00)	0.500	0.500	1.000	1.000	0.000	--	0.000	1.000	--	--	1.000	1.000	0.000	0.500	0.000	0.000	0.000	0.000	0.000	vt28	|	
vt29	--	0.00)	0.000	0.000	0.600	0.600	0.000	--	0.200	0.400	--	--	0.200	0.200	0.000	0.000	0.000	0.200	0.000	0.000	0.000	vt29	|	a+i
vt30	0.000	--	0.000	0.000	0.000	0.250	0.250	--	0.000	0.500	--	--	0.500	0.500	0.000	0.000	0.000	0.250	0.000	0.000	--	vt30	|	
vt31	--	0.00)	0.142	0.142	0.428	0.428	0.142	--	0.000	0.142	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt31	|	
vt32	--	--	--	0.000	0.000	0.000	1.000	--	0.000	0.000	--	--	0.000	0.000	0.000	--	--	0.000	0.000	0.000	0.000	vt32	|	
vt33	0.000	--	0.000	0.000	0.000	0.000	0.000	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	--	vt33	|	
vt34	--	0.00)	0.000	0.000	1.000	1.000	--	1.000	0.000	0.000	--	--	1.000	1.000	0.000	0.000	0.000	1.000	--	--	--	vt34	|	
vt35	0.272	--	0.727	0.727	0.818	0.818	--	0.636	0.000	0.909	--	--	0.909	0.727	0.363	0.000	0.000	0.181	--	--	0.000	vt35	|	
vt36	0.142	--	1.000	1.000	0.714	0.714	--	0.142	0.000	0.857	--	--	0.857	0.714	0.714	0.000	0.571	0.142	--	--	--	vt36	|	
vt60	--	0.05)	0.033	0.033	0.183	0.183	0.066	--	0.033	0.083	--	--	0.083	0.116	0.000	0.000	0.000	0.000	0.016	0.016	0.016	vt60	|	
vt62	--	--	0.000	0.333	0.333	0.333	0.000	0.666	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	--	vt62	|	
vt67	0.000	0.00)	0.100	0.100	1.000	1.000	0.400	--	0.100	0.100	--	--	0.100	0.200	0.000	0.000	0.000	0.000	0.000	0.000	0.200	vt67	|	
vt68	0.000	0.00)	0.000	0.000	1.000	1.000	--	--	0.000	0.000	--	--	0.000	1.000	0.000	0.000	0.000	0.000	--	--	0.000	vt68	|	
vt69	--	0.00)	0.000	0.000	1.000	1.000	1.000	0.000	0.000	0.000	--	--	0.500	0.000	0.000	0.000	0.000	0.000	0.000	0.000	--	vt69	|	
vt70	--	--	0.000	0.000	0.333	0.333	--	0.333	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	--	--	--	vt70	|	
vt71	--	0.00)	0.000	0.000	0.444	0.444	0.166	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	--	vt71	/	
vt03	0.000	--	1.000	1.000	--	--	--	--	0.000	0.500	0.500	--	--	--	0.500	1.000	0.500	0.500	--	--	0.000	vt03	\	
vt04	0.250	--	0.000	0.000	--	--	--	--	0.000	0.000	0.000	--	--	--	0.000	0.000	0.125	0.375	--	--	0.000	vt04	|	
vt05	0.000	--	0.200	0.200	--	--	--	--	0.000	0.400	0.400	--	--	--	0.000	0.000	0.000	0.000	--	--	0.000	vt05	|	
vt06	0.250	--	0.500	0.500	--	--	--	--	0.000	0.000	0.000	--	--	--	0.250	0.500	0.500	0.250	--	--	0.000	vt06	|	a
vt07	--	0.00)	0.000	0.000	--	--	--	--	0.000	0.000	0.000	--	--	--	0.000	0.000	--	0.000	--	--	--	vt07	|	
vt12	0.000	--	0.000	0.000	--	--	--	--	0.000	0.000	0.000	--	--	--	0.000	0.000	0.000	0.250	--	--	0.000	vt12	|	
vt13	0.000	--	0.500	0.500	--	--	--	--	0.000	0.500	0.500	--	--	--	0.000	0.500	0.500	0.500	--	--	0.000	vt13	/	
	16	21	36	37	30	31	21	10	37	37	7	1	31	31	37	37	35	38	22	22	21			
	0.070	0.014	0.189	0.193	0.673	0.683	0.307	0.488	0.009	0.252	0.200	0.000	0.355	0.357	0.049	0.108	0.097	0.124	0.054	0.154	0.010			
	Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15	Sys16	Sys17	Sys18	Sys19	Sys20	Sys21			

Known-Item Search Results - Precision@100 by Topic and System

Topic groupings: i = interactive only a+i = automatic and interactive a = automatic only

Example: Intersection of known item (KI) and result item (RI) must be at least 0.666 of KI and 0.333 of RI

	Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15	Sys16	Sys17	Sys18	Sys19	Sys20	Sys21		
vt01	--	--	--	--	--	0.050	0.000	--	--	--	--	0.000	0.020	0.020	--	0.000	0.000	0.020	0.000	0.020	--	vt01	i
vt08	--	0.000	0.000	0.000	0.040	0.040	0.010	--	0.000	0.010	--	--	0.010	0.010	0.000	0.000	0.000	0.000	0.010	0.010	--	vt08	\
vt14	0.010	--	0.020	0.020	0.050	0.050	--	--	0.000	0.000	--	--	0.050	0.050	0.000	0.020	0.010	0.010	--	--	0.000	vt14	|
vt15	--	0.010	0.000	0.000	0.020	0.030	0.010	--	0.000	0.010	--	--	0.020	0.010	0.000	0.000	0.010	0.010	0.000	0.000	0.000	vt15	|
vt16	0.000	0.000	0.000	0.000	0.020	0.020	0.020	--	0.000	0.000	--	--	0.010	0.010	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt16	|
vt17	--	0.000	0.000	0.000	0.020	0.020	--	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.010	0.000	--	--	0.000	vt17	|
vt18	--	0.000	0.000	0.000	0.010	0.010	0.010	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt18	|
vt19	--	0.000	0.000	0.000	0.050	0.050	0.010	--	0.000	0.000	--	--	0.050	0.050	0.000	0.000	0.000	0.010	0.050	0.050	--	vt19	|
vt20	--	--	0.010	0.010	0.010	0.010	--	0.010	0.000	0.010	--	--	0.010	0.010	0.000	0.000	0.000	0.000	0.000	0.000	--	vt20	|
vt21	--	0.000	0.000	0.000	0.000	0.000	--	0.000	0.000	0.000	--	--	0.000	0.000	0.000	0.000	--	0.000	--	--	--	vt21	|
vt22	--	0.000	0.010	0.010	0.050	0.050	0.040	0.030	0.000	0.000	--	--	0.050	0.050	0.000	0.030	0.020	0.010	0.000	0.050	--	vt22	|
vt23	--	0.000	0.010	0.010	0.020	0.020	0.010	0.010	0.000	0.000	--	--	0.000	0.000	0.000	0.010	0.000	0.000	0.000	0.020	--	vt23	|
vt25	--	0.000	0.000	0.000	0.010	0.010	0.000	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt25	|
vt26	0.000	--	0.000	0.000	0.010	0.010	--	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	--	--	--	vt26	|
vt27	0.000	0.000	0.000	0.000	0.040	0.040	0.000	--	0.000	0.020	--	--	0.020	0.020	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt27	|
vt28	--	0.000	0.010	0.010	0.020	0.020	0.000	--	0.000	0.000	--	--	0.020	0.020	0.000	0.010	0.000	0.000	0.000	0.000	0.000	vt28	| a+i
vt29	--	0.000	0.000	0.000	0.030	0.030	0.000	--	0.010	0.010	--	--	0.010	0.010	0.000	0.000	0.000	0.010	0.000	0.000	0.000	vt29	|
vt30	0.000	--	0.000	0.000	0.000	0.010	0.010	--	0.000	0.000	--	--	0.020	0.020	0.000	0.000	0.000	0.010	0.000	0.000	--	vt30	|
vt31	--	0.000	0.010	0.010	0.030	0.030	0.010	--	0.000	0.010	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt31	|
vt32	--	--	--	0.000	0.000	0.000	0.010	--	0.000	0.000	--	--	0.000	0.000	0.000	--	--	0.000	0.000	0.000	0.000	vt32	|
vt33	0.000	--	0.000	0.000	0.000	0.000	0.000	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	--	vt33	|
vt34	--	0.000	0.000	0.000	0.010	0.010	--	0.010	0.000	0.000	--	--	0.010	0.010	0.000	0.000	0.000	0.010	--	--	--	vt34	|
vt35	0.030	--	0.080	0.080	0.090	0.090	--	0.070	0.000	0.100	--	--	0.080	0.080	0.040	0.000	0.000	0.020	--	--	0.000	vt35	|
vt36	0.010	--	0.070	0.070	0.050	0.050	--	0.010	0.000	0.060	--	--	0.050	0.050	0.050	0.000	0.040	0.010	--	--	--	vt36	|
vt60	--	0.030	0.020	0.020	0.110	0.110	0.040	--	0.020	0.050	--	--	0.050	0.070	0.000	0.000	0.000	0.000	0.010	0.010	0.010	vt60	|
vt62	--	--	0.000	0.010	0.010	0.010	0.000	0.020	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	--	vt62	|
vt67	0.000	0.000	0.010	0.010	0.100	0.100	0.030	--	0.010	0.010	--	--	0.010	0.020	0.000	0.000	0.000	0.000	0.000	0.000	0.020	vt67	|
vt68	0.000	0.000	0.000	0.000	0.010	0.010	--	--	0.000	0.000	--	--	0.000	0.010	0.000	0.000	0.000	0.000	--	--	0.000	vt68	|
vt69	--	0.000	0.000	0.000	0.020	0.020	0.020	0.000	0.000	0.000	--	--	0.010	0.000	0.000	0.000	0.000	0.000	0.000	0.000	--	vt69	|
vt70	--	--	0.000	0.000	0.010	0.010	--	0.010	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	--	--	--	vt70	|
vt71	--	0.000	0.000	0.000	0.080	0.080	0.020	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	--	vt71	/
vt03	0.000	--	0.020	0.020	--	--	--	--	0.000	0.010	0.010	--	--	--	0.010	0.020	0.010	0.010	--	--	0.000	vt03	\
vt04	0.020	--	0.000	0.000	--	--	--	--	0.000	0.000	0.000	--	--	--	0.000	0.000	0.010	0.030	--	--	0.000	vt04	|
vt05	0.000	--	0.010	0.010	--	--	--	--	0.000	0.020	0.020	--	--	--	0.000	0.000	0.000	0.000	--	--	0.000	vt05	|
vt06	0.010	--	0.020	0.020	--	--	--	--	0.000	0.000	0.000	--	--	--	0.010	0.020	0.020	0.010	--	--	0.000	vt06	| a
vt07	--	0.000	0.000	0.000	--	--	--	--	0.000	0.000	0.000	--	--	--	0.000	0.000	--	0.000	--	--	--	vt07	|
vt12	0.000	--	0.000	0.000	--	--	--	--	0.000	0.000	0.000	--	--	--	0.000	0.000	0.000	0.010	--	--	0.000	vt12	|
vt13	0.000	--	0.010	0.010	--	--	--	--	0.000	0.010	0.010	--	--	--	0.000	0.010	0.010	0.010	--	--	0.000	vt13	/
	16	21	36	37	30	31	21	10	37	37	7	1	31	31	37	37	35	38	22	22	21		
	0.005	0.002	0.009	0.009	0.031	0.032	0.012	0.017	0.001	0.009	0.006	0.000	0.016	0.017	0.003	0.003	0.004	0.005	0.003	0.007	0.001		
	Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15	Sys16	Sys17	Sys18	Sys19	Sys20	Sys21		

TREC-2001 Video Track

Known-Item Search Results - Precision@ResultSetSize by Topic and System

Topic groupings: i = interactive only   a+i = automatic and interactive   a = automatic only

Example: Intersection of known item (KI) and result item (RI) must be at least 0.666 of KI and 0.333 of RI

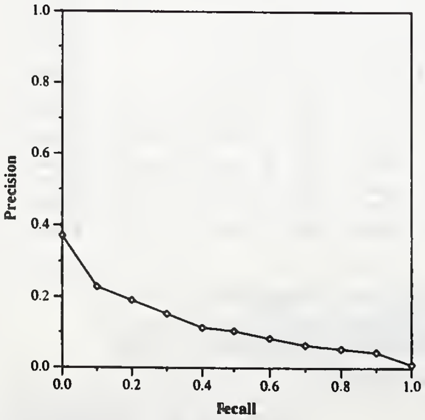
	Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15	Sys16	Sys17	Sys18	Sys19	Sys20	Sys21		
vt01	--	--	--	--	--	0.625	0.000	--	--	--	--	0.000	0.020	0.020	--	0.000	0.000	0.020	0.000	0.020	--	vt01	i
vt08	--	0.000	0.000	0.000	0.444	0.444	0.166	--	0.000	0.010	--	--	0.010	0.010	0.000	0.000	0.000	0.000	0.010	0.010	--	vt08	a+i
vt14	0.010	--	0.020	0.020	1.000	1.000	--	--	0.000	0.000	--	--	0.050	0.050	0.000	0.020	0.010	0.010	--	--	0.000	vt14	
vt15	--	0.010	0.000	0.000	1.000	1.000	0.142	--	0.000	0.010	--	--	0.020	0.010	0.000	0.000	0.010	0.010	0.000	0.000	0.000	vt15	
vt16	0.000	0.000	0.000	0.000	1.000	1.000	0.400	--	0.000	0.000	--	--	0.010	0.010	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt16	
vt17	--	0.000	0.000	0.000	1.000	1.000	--	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.010	0.000	--	--	0.000	vt17	
vt18	--	0.000	0.000	0.000	0.250	0.250	0.333	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt18	
vt19	--	0.000	0.000	0.000	1.000	1.000	0.250	--	0.000	0.000	--	--	0.050	0.050	0.000	0.000	0.000	0.032	0.050	0.050	--	vt19	
vt20	--	--	0.333	0.333	1.000	1.000	--	1.000	0.000	0.010	--	--	0.047	0.047	0.000	0.000	0.000	0.000	0.000	0.000	--	vt20	
vt21	--	0.000	0.000	0.000	0.000	0.000	--	0.000	0.000	0.000	--	--	0.000	0.000	0.000	0.000	--	0.000	--	--	--	vt21	
vt22	--	0.000	0.010	0.010	1.000	1.000	0.666	0.214	0.000	0.000	--	--	0.050	0.050	0.000	0.030	0.080	0.010	0.000	0.250	--	vt22	
vt23	--	0.000	0.010	0.010	1.000	1.000	1.000	1.000	0.000	0.000	--	--	0.000	0.000	0.000	0.010	0.000	0.000	0.000	0.100	--	vt23	
vt25	--	0.000	0.000	0.000	1.000	1.000	0.000	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt25	
vt26	0.000	--	0.000	0.000	0.500	0.500	--	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	--	--	--	vt26	
vt27	0.000	0.000	0.000	0.000	1.000	1.000	0.000	--	0.000	0.020	--	--	0.020	0.020	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt27	
vt28	--	0.000	0.010	0.010	0.666	0.666	0.000	--	0.000	0.000	--	--	0.020	0.020	0.000	0.010	0.000	0.000	0.000	0.000	0.000	vt28	
vt29	--	0.000	0.000	0.000	0.750	0.750	0.000	--	0.032	0.010	--	--	0.010	0.010	0.000	0.000	0.000	0.011	0.000	0.000	0.000	vt29	
vt30	0.000	--	0.000	0.000	0.000	0.500	0.250	--	0.000	0.000	--	--	0.020	0.020	0.000	0.000	0.000	0.010	0.000	0.000	--	vt30	
vt31	--	0.000	0.010	0.010	0.750	0.750	0.250	--	0.000	0.010	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	vt31	
vt32	--	--	--	0.000	0.000	0.000	0.250	--	0.000	0.000	--	--	0.000	0.000	0.000	--	--	0.000	0.000	0.000	0.000	vt32	
vt33	0.000	--	0.000	0.000	0.000	0.000	0.000	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	--	vt33	
vt34	--	0.000	0.000	0.000	1.000	1.000	--	1.000	0.000	0.000	--	--	0.010	0.010	0.000	0.000	0.000	0.010	--	--	--	vt34	
vt35	0.030	--	0.080	0.080	0.900	0.900	--	0.070	0.000	0.100	--	--	0.380	0.380	0.266	0.000	0.000	0.020	--	--	0.000	vt35	
vt36	0.010	--	0.070	0.070	1.000	1.000	--	0.010	0.000	0.060	--	--	0.238	0.238	0.333	0.000	0.040	0.010	--	--	--	vt36	
vt60	--	0.030	0.020	0.020	0.550	0.550	0.200	--	1.000	0.050	--	--	0.050	0.070	0.000	0.000	0.000	0.000	0.010	0.010	0.090	vt60	
vt62	--	--	0.000	0.010	0.200	0.200	0.000	1.000	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	--	vt62	
vt67	0.000	0.000	0.010	0.010	1.000	1.000	0.750	--	0.045	0.010	--	--	0.010	0.020	0.000	0.000	0.000	0.000	0.000	0.000	0.166	vt67	
vt68	0.000	0.000	0.000	0.000	1.000	1.000	--	--	0.000	0.000	--	--	0.000	0.010	0.000	0.000	0.000	0.000	--	--	0.000	vt68	
vt69	--	0.000	0.000	0.000	1.000	1.000	0.285	0.000	0.000	0.000	--	--	0.010	0.000	0.000	0.000	0.000	0.000	0.000	0.000	--	vt69	
vt70	--	--	0.000	0.000	0.083	0.083	--	0.050	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	--	--	--	vt70	
vt71	--	0.000	0.000	0.000	0.615	0.615	0.142	--	0.000	0.000	--	--	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	--	vt71	
vt03	0.000	--	0.020	0.020	--	--	--	--	0.000	0.010	0.010	--	--	--	0.066	0.020	0.010	0.010	--	--	0.000	vt03	a
vt04	0.020	--	0.000	0.000	--	--	--	--	0.000	0.000	0.000	--	--	--	0.000	0.000	0.010	0.030	--	--	0.000	vt04	
vt05	0.000	--	0.010	0.010	--	--	--	--	0.000	0.020	0.020	--	--	--	0.000	0.000	0.000	0.000	--	--	0.000	vt05	
vt06	0.010	--	0.020	0.020	--	--	--	--	0.000	0.000	0.000	--	--	--	0.066	0.020	0.020	0.010	--	--	0.000	vt06	
vt07	--	0.000	0.000	0.000	--	--	--	--	0.000	0.000	0.000	--	--	--	0.000	0.000	--	0.000	--	--	--	vt07	
vt12	0.000	--	0.000	0.000	--	--	--	--	0.000	0.000	0.000	--	--	--	0.000	0.000	0.000	0.010	--	--	0.000	vt12	
vt13	0.000	--	0.010	0.010	--	--	--	--	0.000	0.010	0.010	--	--	--	0.000	0.010	0.026	0.010	--	--	0.000	vt13	
	16	21	36	37	30	31	21	10	37	37	7	1	31	31	37	37	35	38	22	22	21		
	0.005	0.002	0.018	0.017	0.690	0.704	0.242	0.434	0.029	0.009	0.006	0.000	0.033	0.034	0.020	0.003	0.006	0.006	0.003	0.020	0.012		
	Sys01	Sys02	Sys03	Sys04	Sys05	Sys06	Sys07	Sys08	Sys09	Sys10	Sys11	Sys12	Sys13	Sys14	Sys15	Sys16	Sys17	Sys18	Sys19	Sys20	Sys21		

Summary Statistics	
Run ID:	ajouai0101
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

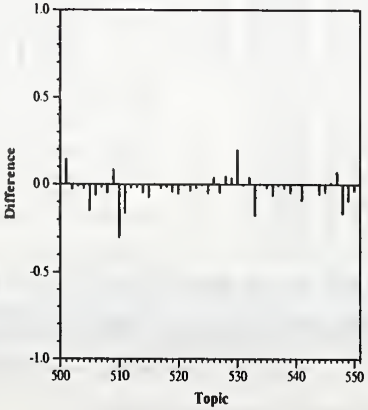
Total number of documents over all topics	
Retrieved:	48652
Relevant:	3363
Rel-ret:	1853

Recall Level Averages	
Recall	Precision
0.00	0.3710
0.10	0.2289
0.20	0.1906
0.30	0.1525
0.40	0.1132
0.50	0.1040
0.60	0.0835
0.70	0.0640
0.80	0.0538
0.90	0.0445
1.00	0.0093
Mean average precision	
non-interpolated	0.1114

Document Level Averages	
	Precision
At 5 docs	0.2040
At 10 docs	0.1980
At 15 docs	0.1693
At 20 docs	0.1560
At 30 docs	0.1493
At 100 docs	0.1074
At 200 docs	0.0850
At 500 docs	0.0576
At 1000 docs	0.0371
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1256



Recall-Precision Curve



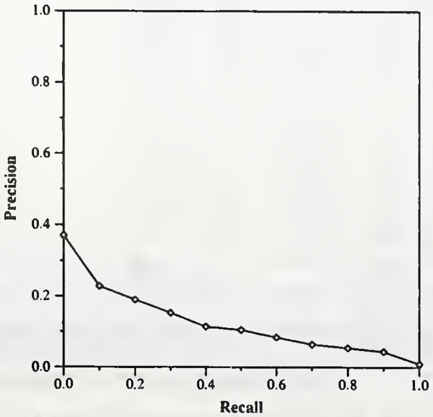
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	ajouai0103
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-used
Number of Topics:	50

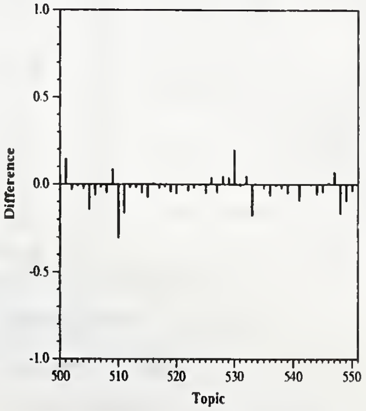
Total number of documents over all topics	
Retrieved:	48652
Relevant:	3363
Rel-ret:	1853

Recall Level Averages	
Recall	Precision
0.00	0.3707
0.10	0.2283
0.20	0.1896
0.30	0.1529
0.40	0.1135
0.50	0.1045
0.60	0.0838
0.70	0.0640
0.80	0.0543
0.90	0.0438
1.00	0.0093
Mean average precision	
non-interpolated	0.1116

Document Level Averages	
	Precision
At 5 docs	0.2040
At 10 docs	0.1960
At 15 docs	0.1693
At 20 docs	0.1570
At 30 docs	0.1520
At 100 docs	0.1072
At 200 docs	0.0851
At 500 docs	0.0576
At 1000 docs	0.0371
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1271



Recall-Precision Curve



Difference from Median in Average Precision per Topic

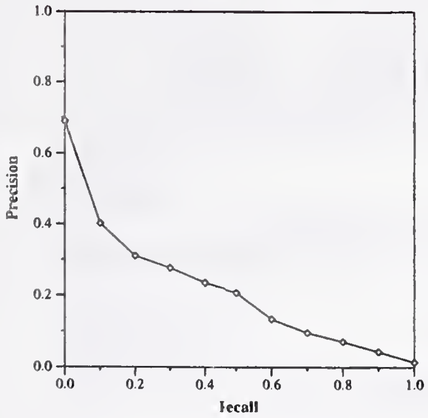
Web track, ad hcc task results — Carnegie Mellon University (CMUDIR)

Summary Statistics	
Run ID:	Lemur
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topi:s:	50

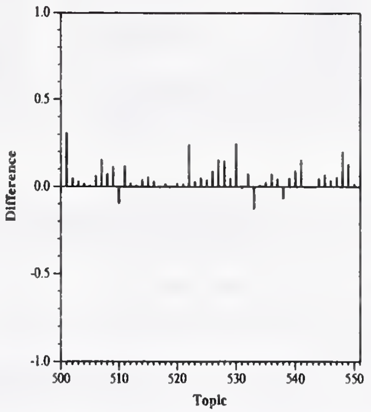
Total number of documents over all topics	
Retrieved:	48667
Relevant:	3363
Rel-ret:	2400

Recall Level Averages	
Recall	Precision
0.00	0.6894
0.10	0.4018
0.20	0.3106
0.30	0.2773
0.40	0.2355
0.50	0.2072
0.60	0.1342
0.70	0.0965
0.80	0.0718
0.90	0.0437
1.00	0.0138
Mean average precision	
non-interpolated	0.1985

Document Level Averages	
	Precision
At 5 docs	0.3720
At 10 docs	0.3200
At 15 docs	0.3093
At 20 docs	0.2920
At 30 docs	0.2580
At 100 docs	0.1758
At 200 docs	0.1305
At 500 docs	0.0770
At 1000 docs	0.0480
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2299



Recall-Pr:cision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	ictweb10n
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

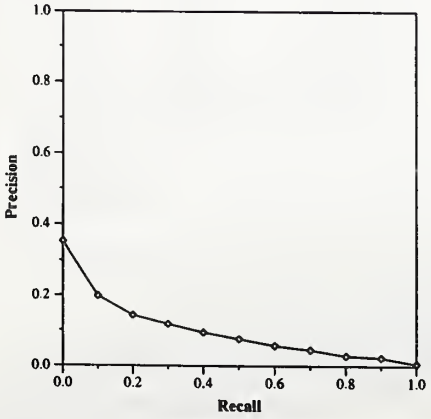
Total number of documents over all topics	
Retrieved:	33114
Relevant:	3363
Rel-ret:	1426

Recall Level Averages	
Recall	Precision
0.00	0.3519
0.10	0.1986
0.20	0.1426
0.30	0.1177
0.40	0.0939
0.50	0.0755
0.60	0.0566
0.70	0.0445
0.80	0.0282
0.90	0.0230
1.00	0.0056

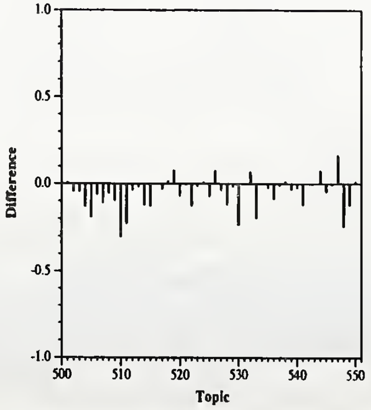
Mean average precision	
non-interpolated	0.0860

Document Level Averages	
	Precision
At 5 docs	0.1640
At 10 docs	0.1600
At 15 docs	0.1480
At 20 docs	0.1450
At 30 docs	0.1307
At 100 docs	0.0942
At 200 docs	0.0731
At 500 docs	0.0476
At 1000 docs	0.0285

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1138



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	ictweb10nf
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

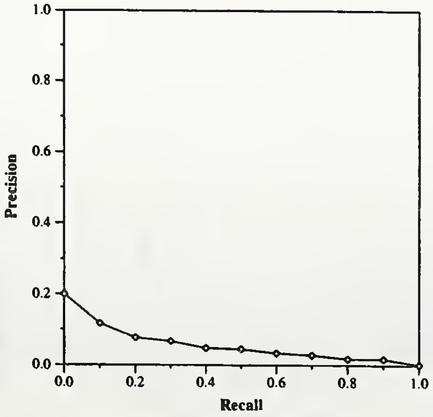
Total number of documents over all topics	
Retrieved:	33114
Relevant:	3363
Rel-ret:	1421

Recall Level Averages	
Recall	Precision
0.00	0.2001
0.10	0.1163
0.20	0.0766
0.30	0.0664
0.40	0.0475
0.50	0.0445
0.60	0.0334
0.70	0.0278
0.80	0.0179
0.90	0.0173
1.00	0.0011

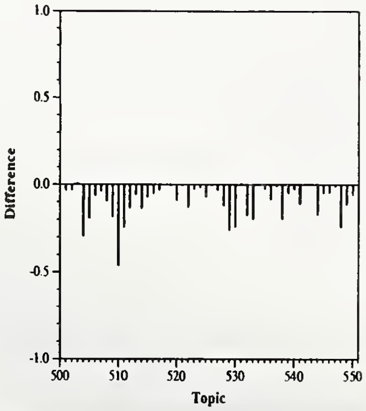
Mean average precision	
non-interpolated	0.0464

Document Level Averages	
	Precision
At 5 docs	0.0680
At 10 docs	0.0620
At 15 docs	0.0693
At 20 docs	0.0620
At 30 docs	0.0673
At 100 docs	0.0568
At 200 docs	0.0502
At 500 docs	0.0384
At 1000 docs	0.0284

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0657



Recall-Precision Curve



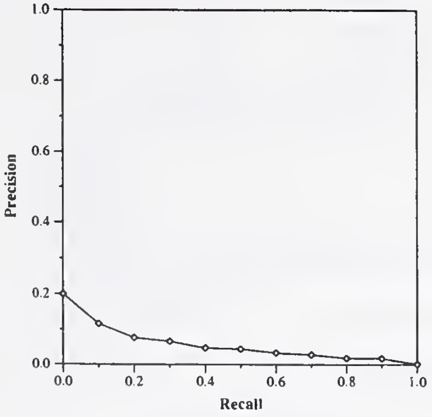
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	ictweb10nfl
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-used
Number of Topics:	50

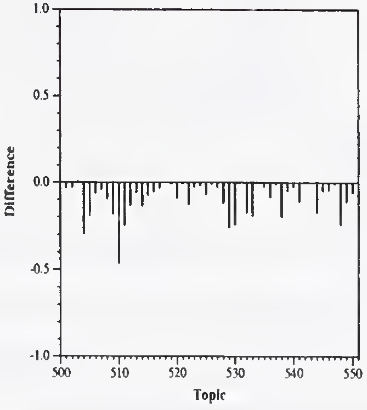
Total number of documents over all topics	
Retrieved:	33114
Relevant:	3363
Rel-ret:	1421

Recall Level Averages	
Recall	Precision
0.00	0.1998
0.10	0.1160
0.20	0.0765
0.30	0.0663
0.40	0.0474
0.50	0.0444
0.60	0.0334
0.70	0.0278
0.80	0.0179
0.90	0.0173
1.00	0.0011
Mean average precision	
non-interpolated	0.0463

Document Level Averages	
	Precision
At 5 docs	0.0680
At 10 docs	0.0620
At 15 docs	0.0693
At 20 docs	0.0610
At 30 docs	0.0673
At 100 docs	0.0568
At 200 docs	0.0501
At 500 docs	0.0384
At 1000 docs	0.0284
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0657



Recall-Precision Curve



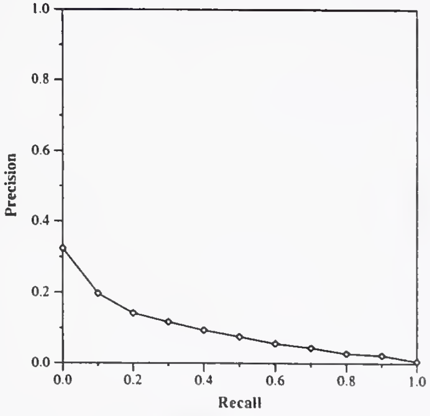
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	ictweb10nl
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-used
Number of Topics:	50

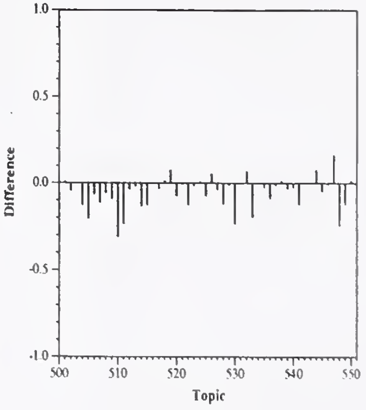
Total number of documents over all topics	
Retrieved:	33114
Relevant:	3363
Rel-ret:	1426

Recall Level Averages	
Recall	Precision
0.00	0.3236
0.10	0.1970
0.20	0.1420
0.30	0.1172
0.40	0.0938
0.50	0.0756
0.60	0.0564
0.70	0.0442
0.80	0.0281
0.90	0.0229
1.00	0.0056
Mean average precision	
non-interpolated	0.0860

Document Level Averages	
	Precision
At 5 docs	0.1600
At 10 docs	0.1580
At 15 docs	0.1480
At 20 docs	0.1410
At 30 docs	0.1273
At 100 docs	0.0954
At 200 docs	0.0731
At 500 docs	0.0476
At 1000 docs	0.0285
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1147



Recall-Precision Curve



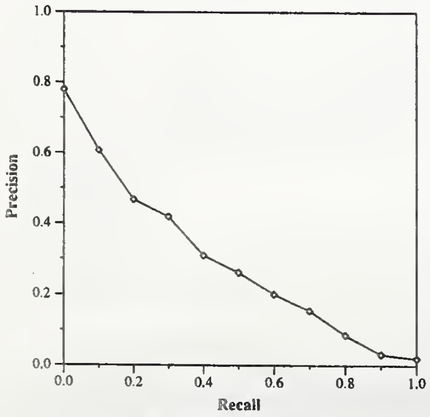
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	csiro0mwa1
Run Description	Manual; docstruct-used, urltext-used, links-used
Number of Topics:	50

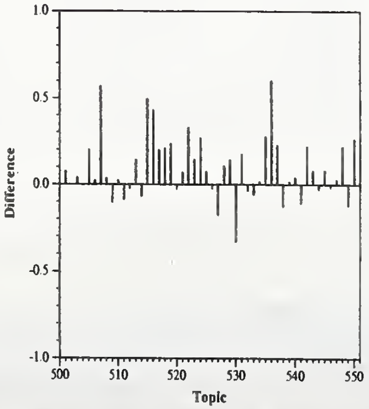
Total number of documents over all topics	
Retrieved:	43353
Relevant:	3363
Rel-ret:	2098

Recall Level Averages	
Recall	Precision
0.00	0.7803
0.10	0.6069
0.20	0.4679
0.30	0.4192
0.40	0.3090
0.50	0.2623
0.60	0.2015
0.70	0.1546
0.80	0.0858
0.90	0.0316
1.00	0.0197
Mean average precision	
non-interpolated	0.2817

Document Level Averages	
	Precision
At 5 docs	0.5440
At 10 docs	0.5020
At 15 docs	0.4520
At 20 docs	0.4170
At 30 docs	0.3640
At 100 docs	0.1968
At 200 docs	0.1286
At 500 docs	0.0700
At 1000 docs	0.0420
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3259



Recall-Precision Curve



Difference from Median in Average Precision per Topic

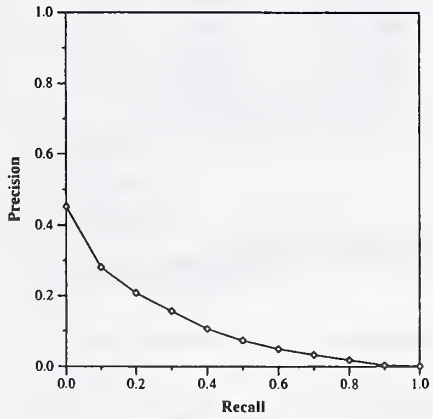
Web track, ad hoc task results — CSIRO

Summary Statistics	
Run ID:	csiro0awa1
Run Description	Automatic, title only; docstruct-used, urltext-used, links-used
Number of Topics:	50

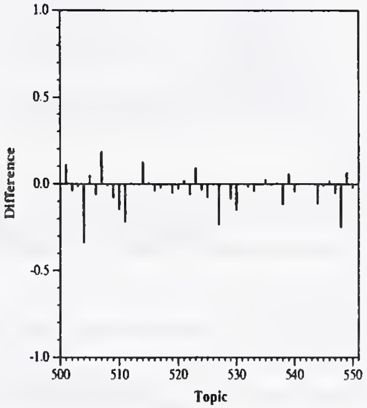
Total number of documents over all topics	
Retrieved:	49568
Relevant:	3363
Rel-ret:	1714

Recall Level Averages	
Recall	Precision
0.00	0.4522
0.10	0.2813
0.20	0.2083
0.30	0.1565
0.40	0.1061
0.50	0.0735
0.60	0.0496
0.70	0.0331
0.80	0.0181
0.90	0.0037
1.00	0.0012
Mean average precision	
non-interpolated	0.1085

Document Level Averages	
	Precision
At 5 docs	0.2760
At 10 docs	0.2440
At 15 docs	0.2227
At 20 docs	0.2190
At 30 docs	0.1880
At 100 docs	0.1058
At 200 docs	0.0794
At 500 docs	0.0532
At 1000 docs	0.0343
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1541



Recall-Precision Curve



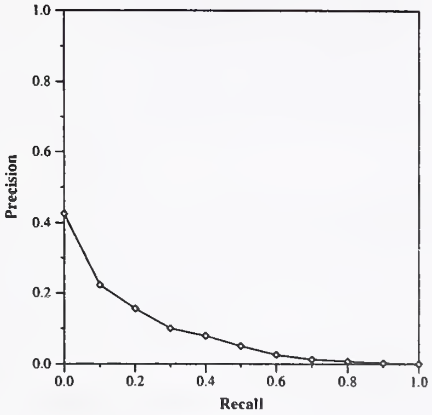
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	csiro0awa2
Run Description	Automatic, title only; docstruct-used, urltext-used, links-used
Number of Topics:	50

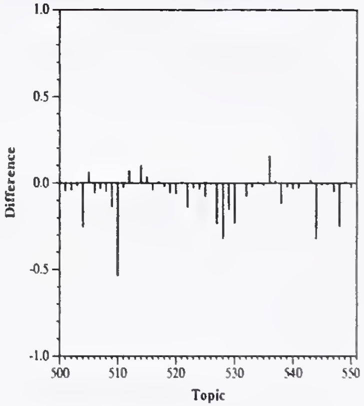
Total number of documents over all topics	
Retrieved:	49568
Relevant:	3363
Rel-ret:	1395

Recall Level Averages	
Recall	Precision
0.00	0.4257
0.10	0.2233
0.20	0.1562
0.30	0.1001
0.40	0.0793
0.50	0.0504
0.60	0.0261
0.70	0.0131
0.80	0.0081
0.90	0.0032
1.00	0.0009
Mean average precision	
non-interpolated	0.0789

Document Level Averages	
	Precision
At 5 docs	0.2200
At 10 docs	0.2000
At 15 docs	0.1893
At 20 docs	0.1770
At 30 docs	0.1587
At 100 docs	0.0948
At 200 docs	0.0707
At 500 docs	0.0435
At 1000 docs	0.0279
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1257



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	csiro0awa3
Run Description	Automatic, title only; docstruct-used, urltext-used, links-used
Number of Topics:	50

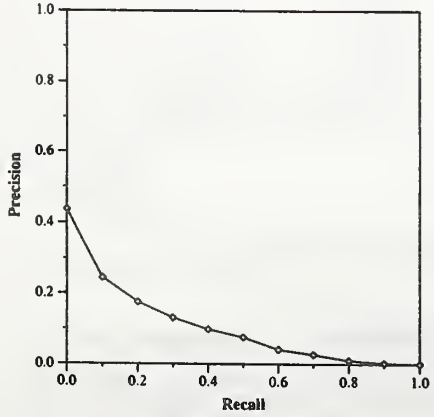
Total number of documents over all topics	
Retrieved:	49568
Relevant:	3363
Rel-ret:	1489

Recall Level Averages	
Recall	Precision
0.00	0.4366
0.10	0.2448
0.20	0.1758
0.30	0.1310
0.40	0.0980
0.50	0.0754
0.60	0.0409
0.70	0.0267
0.80	0.0105
0.90	0.0031
1.00	0.0009

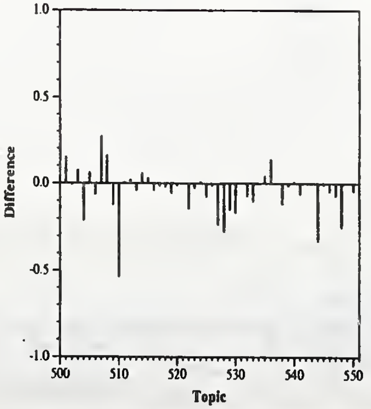
Mean average precision	
non-interpolated	0.0946

Document Level Averages	
	Precision
At 5 docs	0.2200
At 10 docs	0.2320
At 15 docs	0.2093
At 20 docs	0.1940
At 30 docs	0.1687
At 100 docs	0.1076
At 200 docs	0.0725
At 500 docs	0.0458
At 1000 docs	0.0298

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1454



Recall-Precision Curve



Difference from Median in Average Precision per Topic

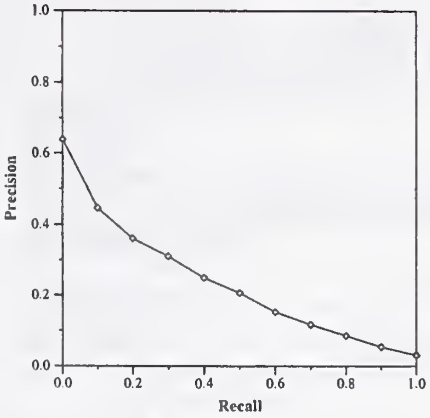
Web track, ad hoc task results — Fondazione Ugo Bordoni

Summary Statistics	
Run ID:	fub01be2
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

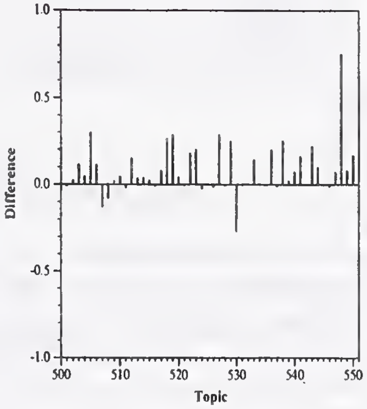
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2307

Recall Level Averages	
Recall	Precision
0.00	0.6389
0.10	0.4466
0.20	0.3611
0.30	0.3107
0.40	0.2497
0.50	0.2067
0.60	0.1525
0.70	0.1166
0.80	0.0854
0.90	0.0539
1.00	0.0307
Mean average precision	
non-interpolated	0.2226

Document Level Averages	
	Precision
At 5 docs	0.3760
At 10 docs	0.3440
At 15 docs	0.3227
At 20 docs	0.2860
At 30 docs	0.2513
At 100 docs	0.1738
At 200 docs	0.1285
At 500 docs	0.0774
At 1000 docs	0.0461
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2461



Recall-Precision Curve



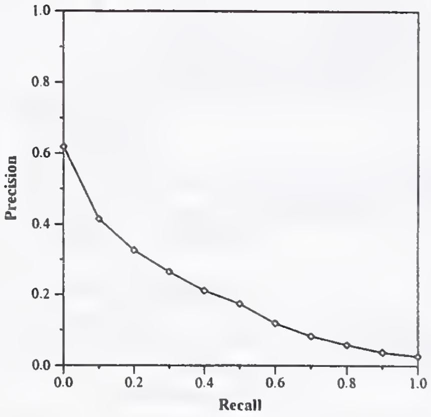
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	fub01idf
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

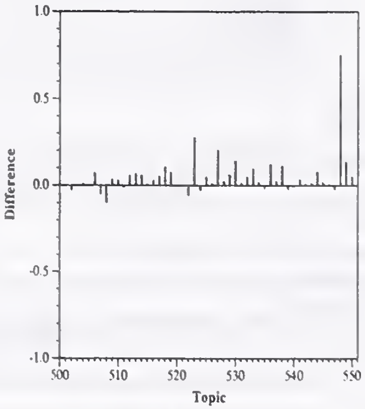
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2118

Recall Level Averages	
Recall	Precision
0.00	0.6183
0.10	0.4150
0.20	0.3261
0.30	0.2647
0.40	0.2110
0.50	0.1740
0.60	0.1185
0.70	0.0827
0.80	0.0581
0.90	0.0372
1.00	0.0262
Mean average precision	
non-interpolated	0.1900

Document Level Averages	
	Precision
At 5 docs	0.3720
At 10 docs	0.3360
At 15 docs	0.2987
At 20 docs	0.2880
At 30 docs	0.2580
At 100 docs	0.1644
At 200 docs	0.1230
At 500 docs	0.0710
At 1000 docs	0.0424
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2275



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	fub01ne
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

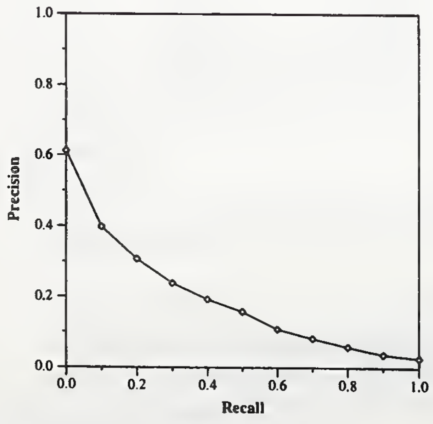
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2060

Recall Level Averages	
Recall	Precision
0.00	0.6120
0.10	0.3975
0.20	0.3072
0.30	0.2385
0.40	0.1924
0.50	0.1573
0.60	0.1079
0.70	0.0817
0.80	0.0585
0.90	0.0369
1.00	0.0262

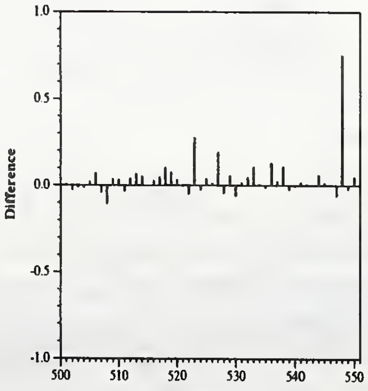
Mean average precision	
non-interpolated	0.1790

Document Level Averages	
	Precision
At 5 docs	0.3800
At 10 docs	0.3240
At 15 docs	0.2880
At 20 docs	0.2720
At 30 docs	0.2440
At 100 docs	0.1498
At 200 docs	0.1122
At 500 docs	0.0658
At 1000 docs	0.0412

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2140



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	fub01ne2
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

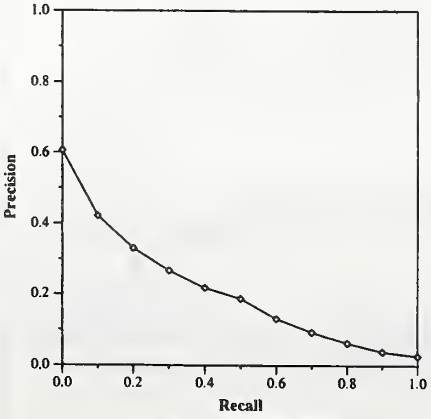
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2144

Recall Level Averages	
Recall	Precision
0.00	0.6056
0.10	0.4214
0.20	0.3307
0.30	0.2669
0.40	0.2178
0.50	0.1874
0.60	0.1307
0.70	0.0925
0.80	0.0623
0.90	0.0381
1.00	0.0258

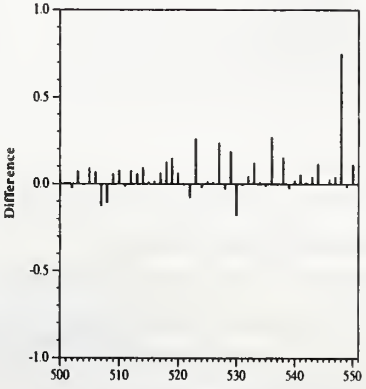
Mean average precision	
non-interpolated	0.1962

Document Level Averages	
	Precision
At 5 docs	0.3760
At 10 docs	0.3280
At 15 docs	0.2920
At 20 docs	0.2760
At 30 docs	0.2507
At 100 docs	0.1642
At 200 docs	0.1219
At 500 docs	0.0699
At 1000 docs	0.0429

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2343



Recall-Precision Curve



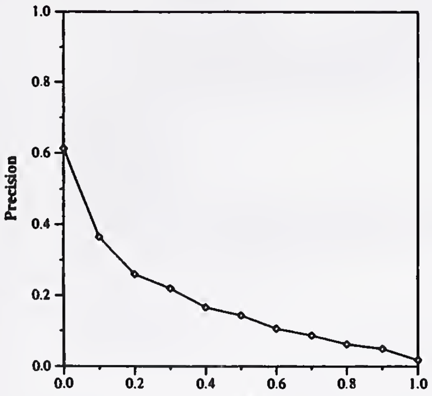
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	fdut10wac01
Run Description	Automatic, adhoc, title+desc+narr; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

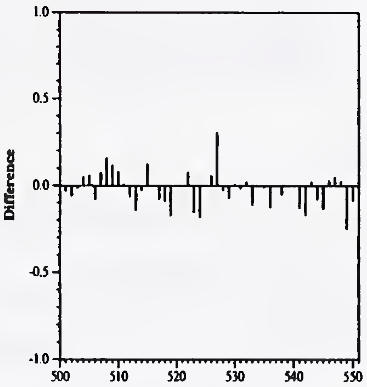
Total number of documents over all topics	
Retrieved:	42322
Relevant:	3363
Rel-ret:	1733

Recall Level Averages	
Recall	Precision
0.00	0.6126
0.10	0.3643
0.20	0.2594
0.30	0.2198
0.40	0.1661
0.50	0.1437
0.60	0.1065
0.70	0.0870
0.80	0.0626
0.90	0.0497
1.00	0.0177
Mean average precision	
non-interpolated	0.1661

Document Level Averages	
	Precision
At 5 docs	0.3840
At 10 docs	0.3020
At 15 docs	0.2653
At 20 docs	0.2500
At 30 docs	0.2260
At 100 docs	0.1510
At 200 docs	0.1055
At 500 docs	0.0569
At 1000 docs	0.0347
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2061



Recall-Precision Curve



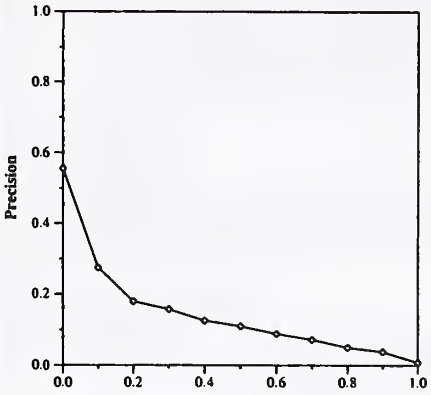
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	fdut10wal01
Run Description	Automatic, adhoc, title+desc+narr; docstruct-notused, urltext-notused, links-used
Number of Topics:	50

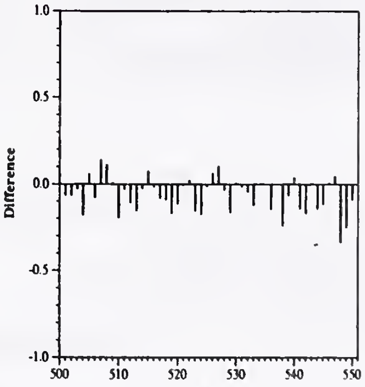
Total number of documents over all topics	
Retrieved:	42322
Relevant:	3363
Rel-ret:	1733

Recall Level Averages	
Recall	Precision
0.00	0.5551
0.10	0.2746
0.20	0.1793
0.30	0.1575
0.40	0.1257
0.50	0.1100
0.60	0.0888
0.70	0.0721
0.80	0.0501
0.90	0.0382
1.00	0.0077
Mean average precision	
non-interpolated	0.1248

Document Level Averages	
	Precision
At 5 docs	0.3320
At 10 docs	0.2480
At 15 docs	0.2093
At 20 docs	0.1850
At 30 docs	0.1627
At 100 docs	0.1272
At 200 docs	0.1022
At 500 docs	0.0569
At 1000 docs	0.0347
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1607



Recall-Precision Curve



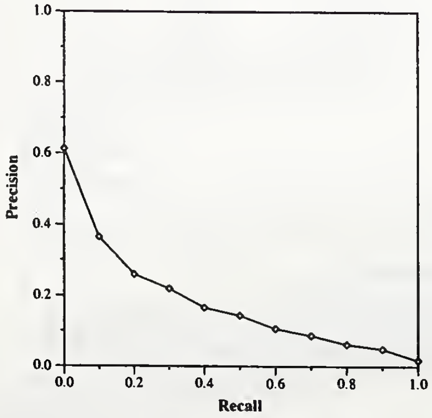
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	fdut10wtc01
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

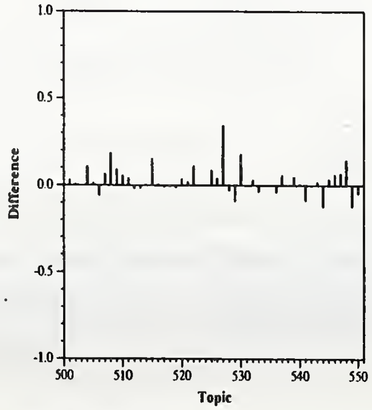
Total number of documents over all topics	
Retrieved:	42322
Relevant:	3363
Rel-ret:	1733

Recall Level Averages	
Recall	Precision
0.00	0.6126
0.10	0.3643
0.20	0.2594
0.30	0.2198
0.40	0.1661
0.50	0.1437
0.60	0.1065
0.70	0.0870
0.80	0.0626
0.90	0.0497
1.00	0.0177
Mean average precision	
non-interpolated	0.1661

Document Level Averages	
	Precision
At 5 docs	0.3840
At 10 docs	0.3020
At 15 docs	0.2653
At 20 docs	0.2500
At 30 docs	0.2260
At 100 docs	0.1510
At 200 docs	0.1055
At 500 docs	0.0569
At 1000 docs	0.0347
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2061



Recall-Precision Curve



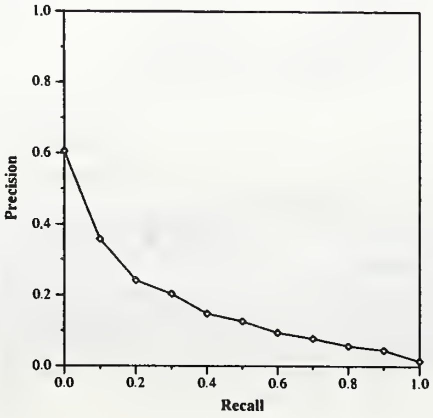
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	fdut10wtl01
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-used
Number of Topics:	50

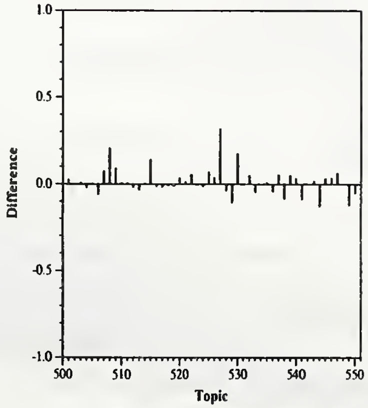
Total number of documents over all topics	
Retrieved:	42322
Relevant:	3363
Rel-ret:	1733

Recall Level Averages	
Recall	Precision
0.00	0.6053
0.10	0.3567
0.20	0.2411
0.30	0.2035
0.40	0.1482
0.50	0.1264
0.60	0.0947
0.70	0.0781
0.80	0.0577
0.90	0.0462
1.00	0.0151
Mean average precision	
non-interpolated	0.1544

Document Level Averages	
	Precision
At 5 docs	0.3920
At 10 docs	0.3100
At 15 docs	0.2667
At 20 docs	0.2410
At 30 docs	0.2133
At 100 docs	0.1456
At 200 docs	0.1019
At 500 docs	0.0560
At 1000 docs	0.0347
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1939



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Web track, ad hoc task results — Fujitsu

Summary Statistics	
Run ID:	flabxtd
Run Description	Automatic, adhoc, title+desc; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

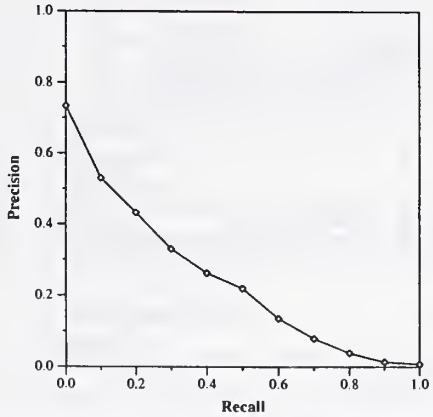
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2449

Recall Level Averages	
Recall	Precision
0.00	0.7336
0.10	0.5296
0.20	0.4329
0.30	0.3298
0.40	0.2618
0.50	0.2195
0.60	0.1361
0.70	0.0796
0.80	0.0391
0.90	0.0139
1.00	0.0078

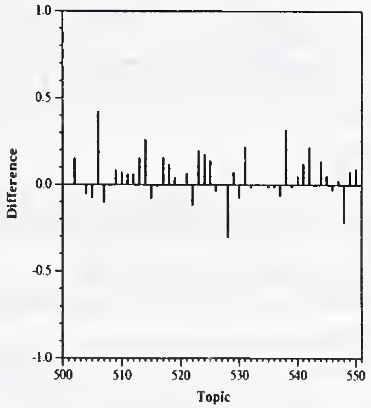
Mean average precision	
non-interpolated	0.2332

Document Level Averages	
	Precision
At 5 docs	0.5000
At 10 docs	0.4460
At 15 docs	0.3973
At 20 docs	0.3660
At 30 docs	0.3200
At 100 docs	0.1988
At 200 docs	0.1408
At 500 docs	0.0810
At 1000 docs	0.0490

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2617



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	flabxtdn
Run Description	Automatic, adhoc, title+desc+narr; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

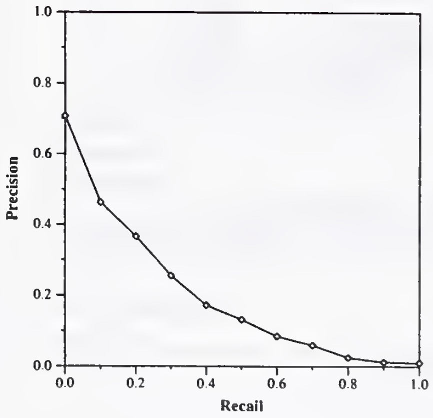
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2170

Recall Level Averages	
Recall	Precision
0.00	0.7075
0.10	0.4628
0.20	0.3661
0.30	0.2546
0.40	0.1725
0.50	0.1322
0.60	0.0854
0.70	0.0605
0.80	0.0254
0.90	0.0128
1.00	0.0111

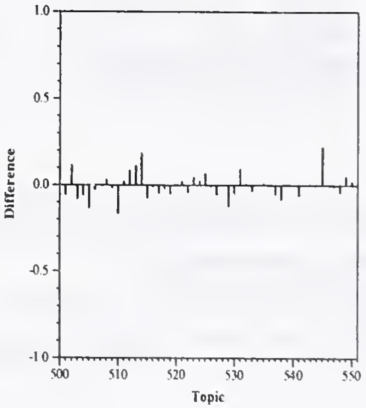
Mean average precision	
non-interpolated	0.1843

Document Level Averages	
	Precision
At 5 docs	0.4640
At 10 docs	0.3960
At 15 docs	0.3547
At 20 docs	0.3160
At 30 docs	0.2733
At 100 docs	0.1732
At 200 docs	0.1214
At 500 docs	0.0709
At 1000 docs	0.0434

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2244



Recall-Precision Curve



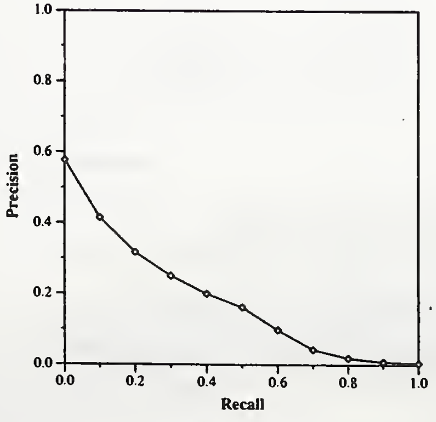
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	flabxt
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

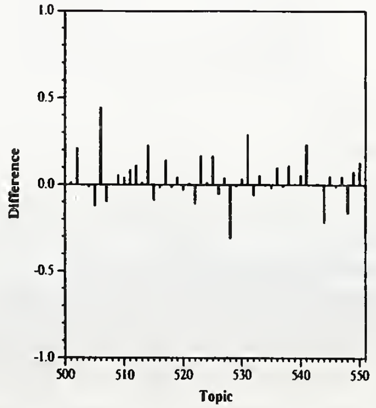
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2155

Recall Level Averages	
Recall	Precision
0.00	0.5775
0.10	0.4147
0.20	0.3170
0.30	0.2505
0.40	0.1999
0.50	0.1616
0.60	0.0981
0.70	0.0429
0.80	0.0192
0.90	0.0081
1.00	0.0045
Mean average precision	
non-interpolated	0.1719

Document Level Averages	
	Precision
At 5 docs	0.3600
At 10 docs	0.3360
At 15 docs	0.3000
At 20 docs	0.2790
At 30 docs	0.2547
At 100 docs	0.1640
At 200 docs	0.1191
At 500 docs	0.0720
At 1000 docs	0.0431
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2182



Recall-Precision Curve



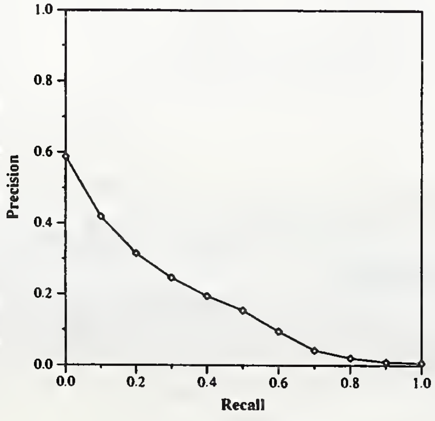
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	flabxtl
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-used
Number of Topics:	50

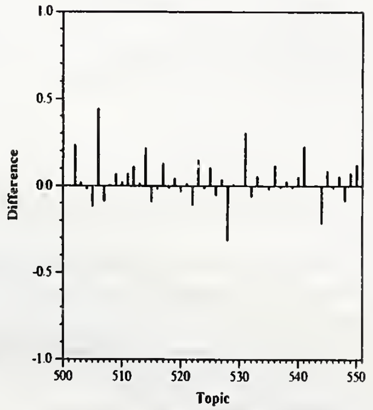
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2151

Recall Level Averages	
Recall	Precision
0.00	0.5874
0.10	0.4186
0.20	0.3148
0.30	0.2458
0.40	0.1947
0.50	0.1541
0.60	0.0957
0.70	0.0424
0.80	0.0208
0.90	0.0093
1.00	0.0068
Mean average precision	
non-interpolated	0.1705

Document Level Averages	
	Precision
At 5 docs	0.3640
At 10 docs	0.3360
At 15 docs	0.3000
At 20 docs	0.2770
At 30 docs	0.2420
At 100 docs	0.1606
At 200 docs	0.1174
At 500 docs	0.0709
At 1000 docs	0.0430
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2083



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Web track, ad hoc task results — Hummingbird

Summary Statistics	
Run ID:	hum01tdlx
Run Description	Automatic, adhoc, title+desc; docstruct-used, urltext-notused, links-notused
Number of Topics:	50

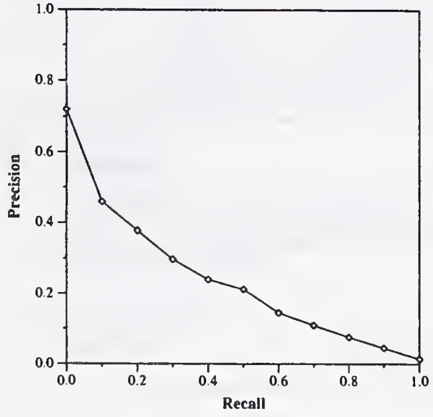
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2470

Recall Level Averages	
Recall	Precision
0.00	0.7207
0.10	0.4585
0.20	0.3781
0.30	0.2977
0.40	0.2409
0.50	0.2127
0.60	0.1453
0.70	0.1098
0.80	0.0759
0.90	0.0454
1.00	0.0143

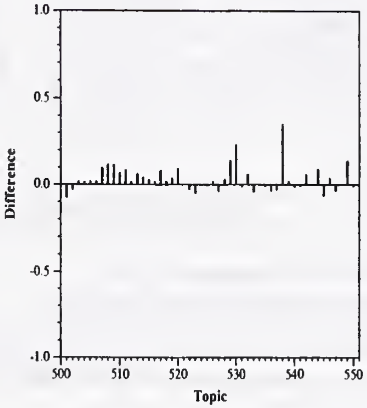
Mean average precision	
non-interpolated	0.2201

Document Level Averages	
	Precision
At 5 docs	0.4440
At 10 docs	0.3660
At 15 docs	0.3373
At 20 docs	0.3220
At 30 docs	0.2833
At 100 docs	0.1862
At 200 docs	0.1387
At 500 docs	0.0800
At 1000 docs	0.0494

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2466



Recall-Precision Curve



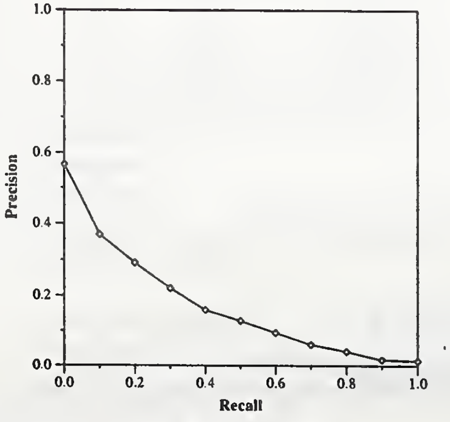
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	hum01t
Run Description	Automatic, title only; docstruct-used, urltext-notused, links-notused
Number of Topics:	50

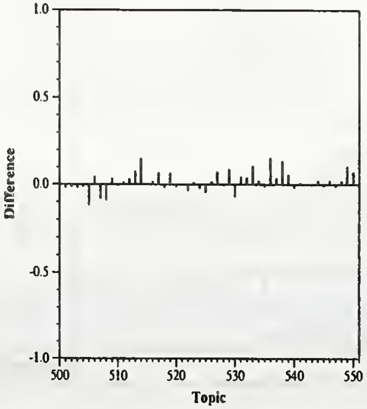
Total number of documents over all topics	
Retrieved:	49737
Relevant:	3363
Rel-ret:	2040

Recall Level Averages	
Recall	Precision
0.00	0.5665
0.10	0.3699
0.20	0.2918
0.30	0.2210
0.40	0.1596
0.50	0.1291
0.60	0.0948
0.70	0.0604
0.80	0.0407
0.90	0.0168
1.00	0.0139
Mean average precision	
non-interpolated	0.1582

Document Level Averages	
	Precision
At 5 docs	0.3960
At 10 docs	0.3080
At 15 docs	0.2813
At 20 docs	0.2600
At 30 docs	0.2267
At 100 docs	0.1442
At 200 docs	0.1100
At 500 docs	0.0665
At 1000 docs	0.0408
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1948



Recall-Precision Curve



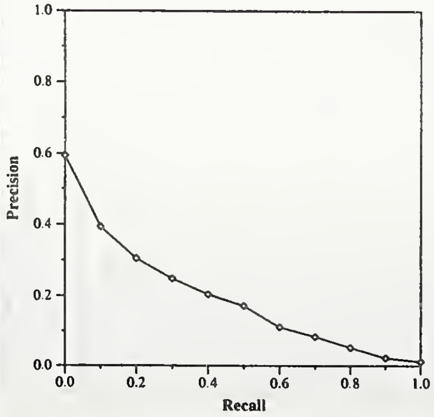
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	hum01tl
Run Description	Automatic, title only; docstruct-used, urltext-notused, links-notused
Number of Topics:	50

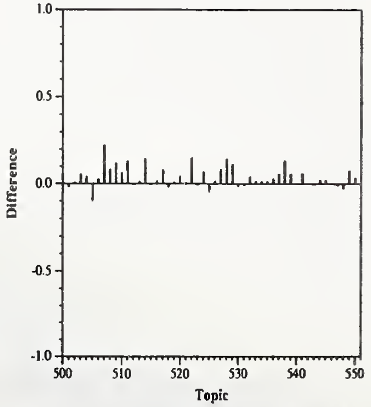
Total number of documents over all topics	
Retrieved:	48740
Relevant:	3363
Rel-ret:	2183

Recall Level Averages	
Recall	Precision
0.00	0.5940
0.10	0.3932
0.20	0.3053
0.30	0.2485
0.40	0.2045
0.50	0.1718
0.60	0.1117
0.70	0.0838
0.80	0.0523
0.90	0.0225
1.00	0.0118
Mean average precision	
non-interpolated	0.1784

Document Level Averages	
	Precision
At 5 docs	0.3680
At 10 docs	0.3220
At 15 docs	0.2893
At 20 docs	0.2660
At 30 docs	0.2373
At 100 docs	0.1510
At 200 docs	0.1157
At 500 docs	0.0696
At 1000 docs	0.0437
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2094



Recall-Precision Curve



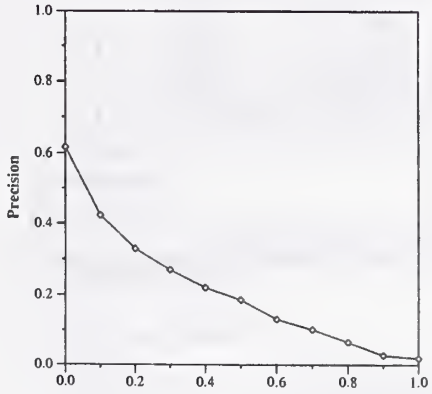
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	hum01tlx
Run Description	Automatic, title only; docstruct-used, urltext-notused, links-notused
Number of Topics:	50

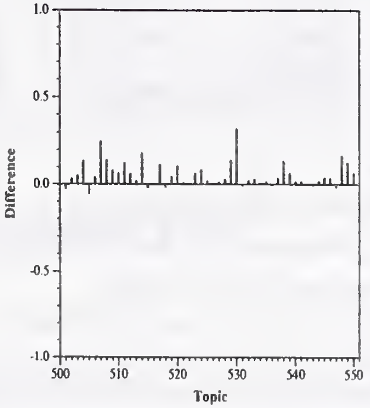
Total number of documents over all topics	
Retrieved:	48740
Relevant:	3363
Rel-ret:	2288

Recall Level Averages	
Recall	Precision
0.00	0.6168
0.10	0.4230
0.20	0.3295
0.30	0.2708
0.40	0.2210
0.50	0.1862
0.60	0.1319
0.70	0.1023
0.80	0.0656
0.90	0.0274
1.00	0.0186
Mean average precision	
non-interpolated	0.1949

Document Level Averages	
	Precision
At 5 docs	0.3840
At 10 docs	0.3320
At 15 docs	0.3053
At 20 docs	0.2860
At 30 docs	0.2547
At 100 docs	0.1648
At 200 docs	0.1195
At 500 docs	0.0719
At 1000 docs	0.0458
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2228



Recall-Precision Curve



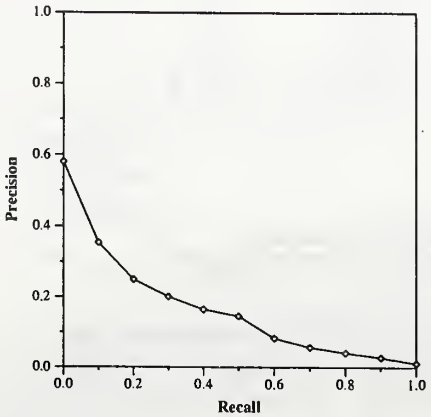
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	ARCJ0
Run Description	Automatic, title only; docstruct-used, urltext-notused, links-notused
Number of Topics:	50

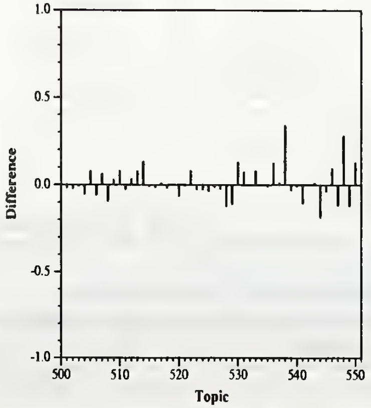
Total number of documents over all topics	
Retrieved:	44561
Relevant:	3363
Rel-ret:	1571

Recall Level Averages	
Recall	Precision
0.00	0.5802
0.10	0.3541
0.20	0.2502
0.30	0.2010
0.40	0.1647
0.50	0.1457
0.60	0.0832
0.70	0.0574
0.80	0.0418
0.90	0.0284
1.00	0.0114
Mean average precision	
non-interpolated	0.1497

Document Level Averages	
	Precision
At 5 docs	0.3080
At 10 docs	0.2640
At 15 docs	0.2360
At 20 docs	0.2210
At 30 docs	0.1920
At 100 docs	0.1194
At 200 docs	0.0872
At 500 docs	0.0496
At 1000 docs	0.0314
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1850



Recall-Precision Curve



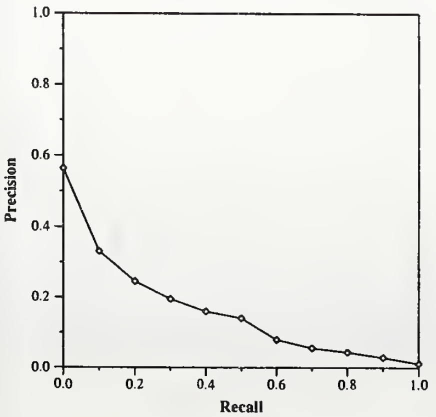
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	ARCJ5
Run Description	Automatic, title only; docstruct-used, urltext-notused, links-used
Number of Topics:	50

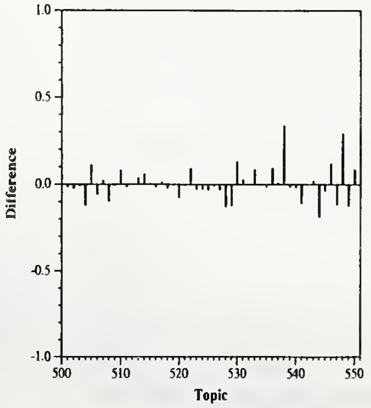
Total number of documents over all topics	
Retrieved:	44561
Relevant:	3363
Rel-ret:	1570

Recall Level Averages	
Recall	Precision
0.00	0.5635
0.10	0.3308
0.20	0.2454
0.30	0.1952
0.40	0.1599
0.50	0.1412
0.60	0.0803
0.70	0.0563
0.80	0.0445
0.90	0.0296
1.00	0.0119
Mean average precision	
non-interpolated	0.1439

Document Level Averages	
	Precision
At 5 docs	0.2840
At 10 docs	0.2640
At 15 docs	0.2360
At 20 docs	0.2130
At 30 docs	0.1833
At 100 docs	0.1188
At 200 docs	0.0866
At 500 docs	0.0492
At 1000 docs	0.0314
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1767



Recall-Precision Curve



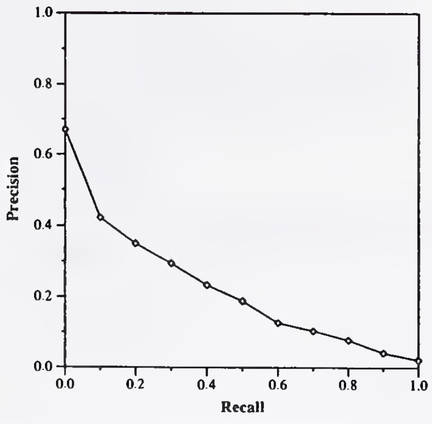
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	JuruFull
Run Description	Automatic, title only; docstruct-used, urltext-used, links-notused
Number of Topics:	50

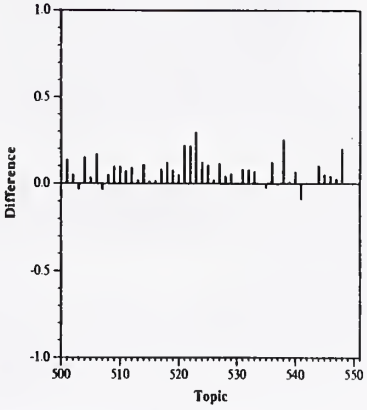
Total number of documents over all topics	
Retrieved:	49325
Relevant:	3363
Rel-ret:	2278

Recall Level Averages	
Recall	Precision
0.00	0.6698
0.10	0.4224
0.20	0.3505
0.30	0.2945
0.40	0.2336
0.50	0.1888
0.60	0.1269
0.70	0.1038
0.80	0.0778
0.90	0.0415
1.00	0.0215
Mean average precision	
non-interpolated	0.2105

Document Level Averages	
	Precision
At 5 docs	0.4320
At 10 docs	0.3620
At 15 docs	0.3373
At 20 docs	0.3130
At 30 docs	0.2753
At 100 docs	0.1722
At 200 docs	0.1240
At 500 docs	0.0737
At 1000 docs	0.0456
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2474



Recall-Precision Curve



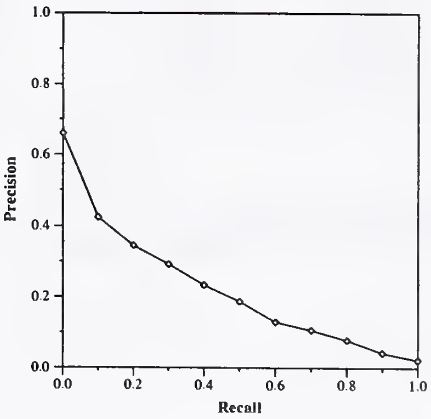
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	JuruFullQE
Run Description	Automatic, title only; docstruct-used, urltext-used, links-notused
Number of Topics:	50

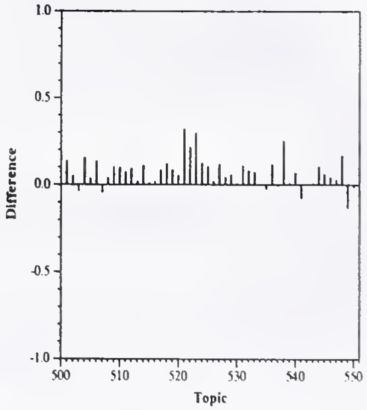
Total number of documents over all topics	
Retrieved:	49325
Relevant:	3363
Rel-ret:	2128

Recall Level Averages	
Recall	Precision
0.00	0.6598
0.10	0.4239
0.20	0.3447
0.30	0.2916
0.40	0.2334
0.50	0.1877
0.60	0.1297
0.70	0.1065
0.80	0.0780
0.90	0.0420
1.00	0.0220
Mean average precision	
non-interpolated	0.2091

Document Level Averages	
	Precision
At 5 docs	0.4400
At 10 docs	0.3540
At 15 docs	0.3360
At 20 docs	0.3160
At 30 docs	0.2740
At 100 docs	0.1692
At 200 docs	0.1198
At 500 docs	0.0696
At 1000 docs	0.0426
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2368



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	JuruPrune005
Run Description	Automatic, title only; docstruct-used, urltext-used, links-notused
Number of Topics:	50

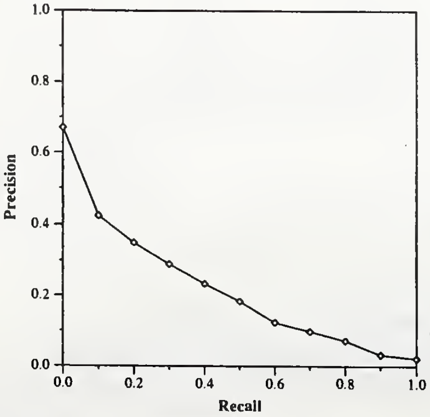
Total number of documents over all topics	
Retrieved:	48688
Relevant:	3363
Rel-ret:	2203

Recall Level Averages	
Recall	Precision
0.00	0.6693
0.10	0.4247
0.20	0.3498
0.30	0.2896
0.40	0.2334
0.50	0.1836
0.60	0.1235
0.70	0.0979
0.80	0.0716
0.90	0.0321
1.00	0.0216

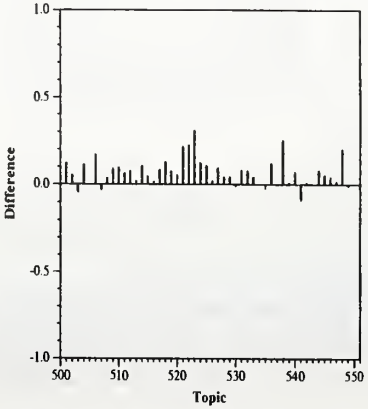
Mean average precision	
non-interpolated	0.2065

Document Level Averages	
	Precision
At 5 docs	0.4240
At 10 docs	0.3620
At 15 docs	0.3400
At 20 docs	0.3090
At 30 docs	0.2760
At 100 docs	0.1730
At 200 docs	0.1231
At 500 docs	0.0718
At 1000 docs	0.0441

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2465



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	JuruPruned01
Run Description	Automatic, title only; docstruct-used, urltext-used, links-notused
Number of Topics:	50

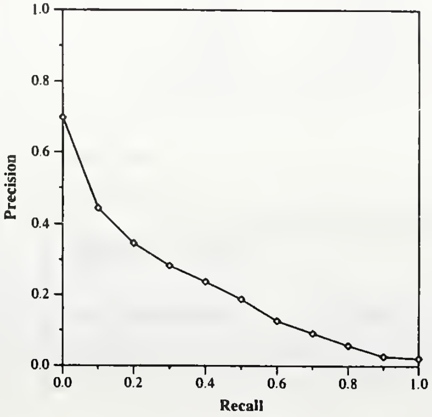
Total number of documents over all topics	
Retrieved:	48349
Relevant:	3363
Rel-ret:	2154

Recall Level Averages	
Recall	Precision
0.00	0.6974
0.10	0.4440
0.20	0.3465
0.30	0.2830
0.40	0.2371
0.50	0.1876
0.60	0.1265
0.70	0.0915
0.80	0.0570
0.90	0.0265
1.00	0.0211

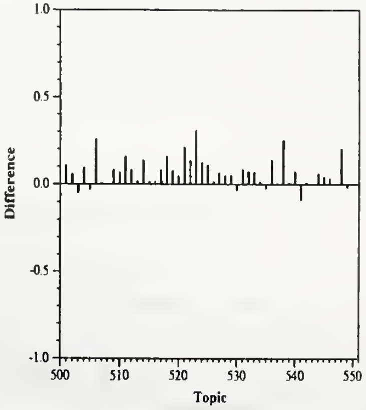
Mean average precision	
non-interpolated	0.2066

Document Level Averages	
	Precision
At 5 docs	0.4160
At 10 docs	0.3600
At 15 docs	0.3307
At 20 docs	0.3070
At 30 docs	0.2733
At 100 docs	0.1748
At 200 docs	0.1239
At 500 docs	0.0717
At 1000 docs	0.0431

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2519



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Web track, ad hoc task results — Illinois Institute of Technology, IIT Information Retrieval Lab

Summary Statistics	
Run ID:	iit01m
Run Description	Manual; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

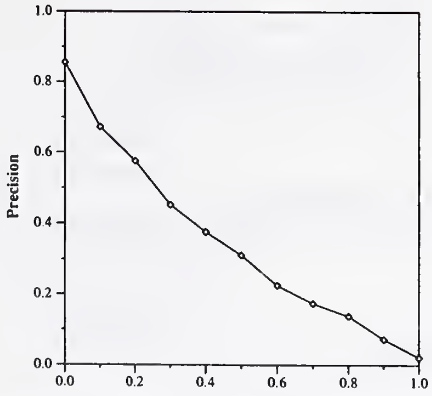
Total number of documents over all topics	
Retrieved:	35954
Relevant:	3363
Rel-ret:	2158

Recall Level Averages	
Recall	Precision
0.00	0.8550
0.10	0.6724
0.20	0.5754
0.30	0.4519
0.40	0.3762
0.50	0.3118
0.60	0.2257
0.70	0.1737
0.80	0.1381
0.90	0.0728
1.00	0.0212

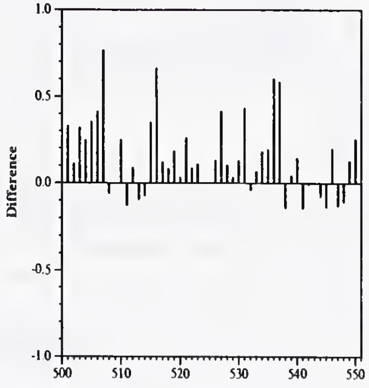
Mean average precision	
non-interpolated	0.3324

Document Level Averages	
	Precision
At 5 docs	0.6320
At 10 docs	0.5880
At 15 docs	0.5373
At 20 docs	0.4730
At 30 docs	0.4013
At 100 docs	0.2080
At 200 docs	0.1397
At 500 docs	0.0741
At 1000 docs	0.0432

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3517



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	iit01tde
Run Description	Automatic, adhoc, title+desc; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

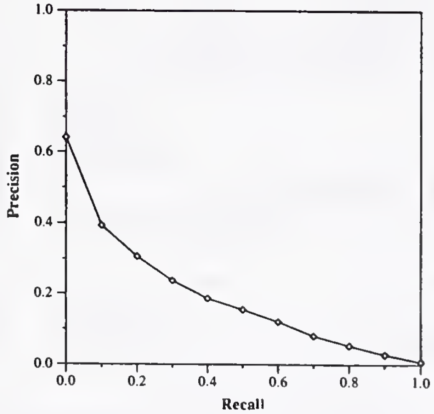
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2251

Recall Level Averages	
Recall	Precision
0.00	0.6413
0.10	0.3926
0.20	0.3064
0.30	0.2373
0.40	0.1862
0.50	0.1546
0.60	0.1207
0.70	0.0808
0.80	0.0529
0.90	0.0280
1.00	0.0079

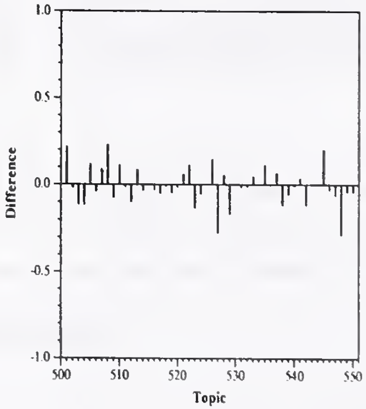
Mean average precision	
non-interpolated	0.1791

Document Level Averages	
	Precision
At 5 docs	0.3520
At 10 docs	0.3400
At 15 docs	0.3027
At 20 docs	0.2830
At 30 docs	0.2560
At 100 docs	0.1690
At 200 docs	0.1199
At 500 docs	0.0719
At 1000 docs	0.0450

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2125



Recall-Precision Curve



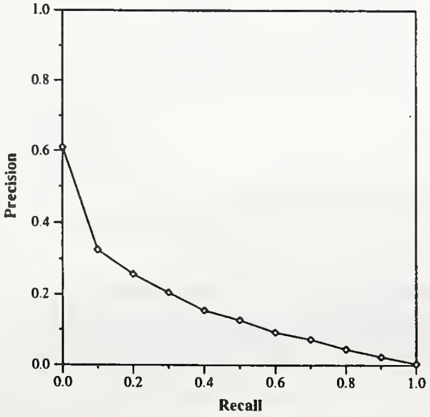
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	iit01t
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

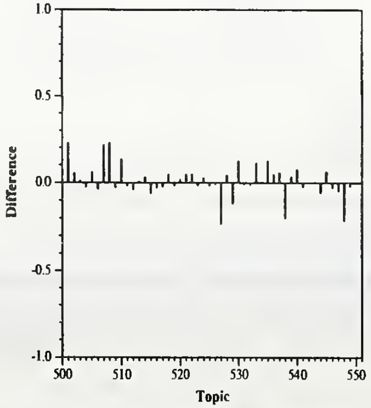
Total number of documents over all topics	
Retrieved:	47959
Relevant:	3363
Rel-ret:	2004

Recall Level Averages	
Recall	Precision
0.00	0.6095
0.10	0.3249
0.20	0.2578
0.30	0.2056
0.40	0.1544
0.50	0.1267
0.60	0.0923
0.70	0.0720
0.80	0.0448
0.90	0.0234
1.00	0.0039
Mean average precision	
non-interpolated	0.1509

Document Level Averages	
	Precision
At 5 docs	0.3320
At 10 docs	0.3000
At 15 docs	0.2760
At 20 docs	0.2470
At 30 docs	0.2167
At 100 docs	0.1392
At 200 docs	0.1029
At 500 docs	0.0620
At 1000 docs	0.0401
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1792



Recall-Precision Curve



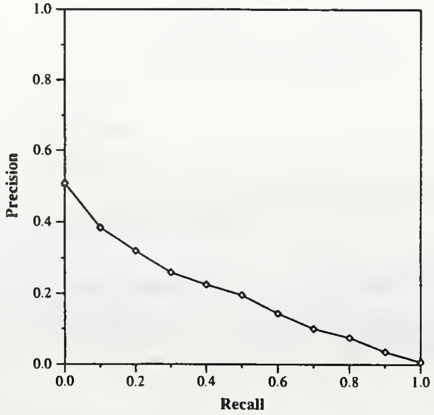
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	iit01tfc
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

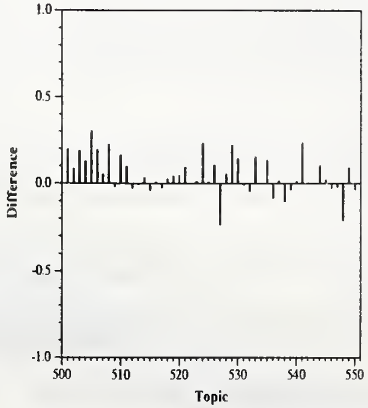
Total number of documents over all topics	
Retrieved:	49001
Relevant:	3363
Rel-ret:	2382

Recall Level Averages	
Recall	Precision
0.00	0.5069
0.10	0.3840
0.20	0.3195
0.30	0.2593
0.40	0.2252
0.50	0.1953
0.60	0.1432
0.70	0.1003
0.80	0.0752
0.90	0.0356
1.00	0.0066
Mean average precision	
non-interpolated	0.1890

Document Level Averages	
	Precision
At 5 docs	0.3080
At 10 docs	0.2840
At 15 docs	0.2707
At 20 docs	0.2590
At 30 docs	0.2427
At 100 docs	0.1692
At 200 docs	0.1273
At 500 docs	0.0782
At 1000 docs	0.0476
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2260



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	icadhoc1
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-used
Number of Topics:	50

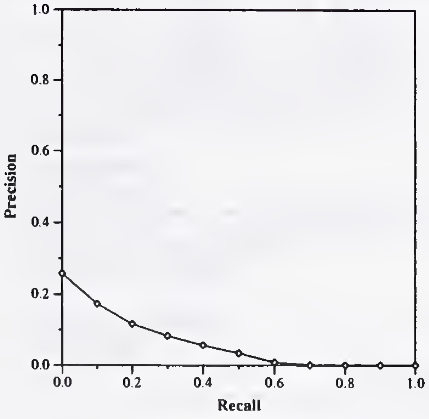
Total number of documents over all topics	
Retrieved:	49436
Relevant:	3363
Rel-ret:	1226

Recall Level Averages	
Recall	Precision
0.00	0.2582
0.10	0.1738
0.20	0.1168
0.30	0.0832
0.40	0.0561
0.50	0.0343
0.60	0.0084
0.70	0.0006
0.80	0.0006
0.90	0.0005
1.00	0.0005

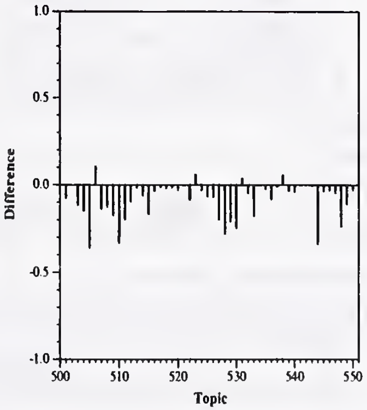
Mean average precision	
non-interpolated	0.0537

Document Level Averages	
	Precision
At 5 docs	0.1400
At 10 docs	0.1260
At 15 docs	0.1227
At 20 docs	0.1220
At 30 docs	0.1100
At 100 docs	0.0796
At 200 docs	0.0606
At 500 docs	0.0367
At 1000 docs	0.0245

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0993



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	icadhoc2
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-used
Number of Topics:	50

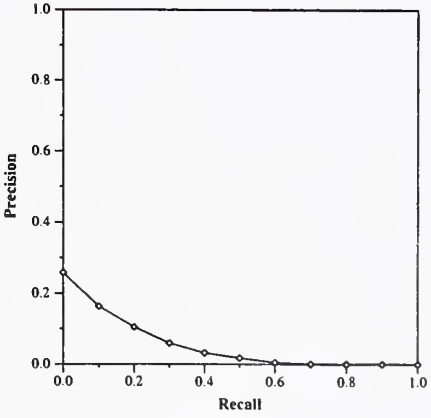
Total number of documents over all topics	
Retrieved:	49439
Relevant:	3363
Rel-ret:	1149

Recall Level Averages	
Recall	Precision
0.00	0.2583
0.10	0.1639
0.20	0.1053
0.30	0.0597
0.40	0.0322
0.50	0.0177
0.60	0.0054
0.70	0.0009
0.80	0.0003
0.90	0.0003
1.00	0.0003

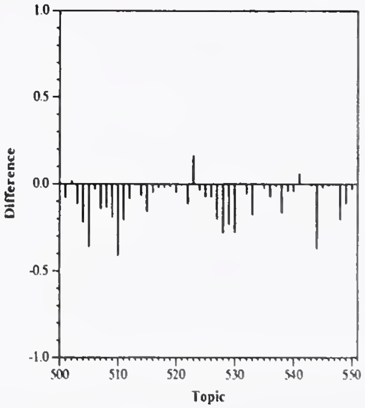
Mean average precision	
non-interpolated	0.0458

Document Level Averages	
	Precision
At 5 docs	0.1400
At 10 docs	0.1400
At 15 docs	0.1373
At 20 docs	0.1370
At 30 docs	0.1207
At 100 docs	0.0828
At 200 docs	0.0603
At 500 docs	0.0346
At 1000 docs	0.0230

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0855



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	icadhoc3
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

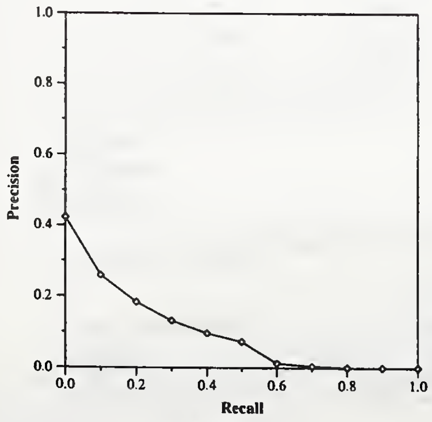
Total number of documents over all topics	
Retrieved:	49439
Relevant:	3363
Rel-ret:	1345

Recall Level Averages	
Recall	Precision
0.00	0.4224
0.10	0.2591
0.20	0.1836
0.30	0.1319
0.40	0.0971
0.50	0.0735
0.60	0.0130
0.70	0.0046
0.80	0.0010
0.90	0.0009
1.00	0.0009

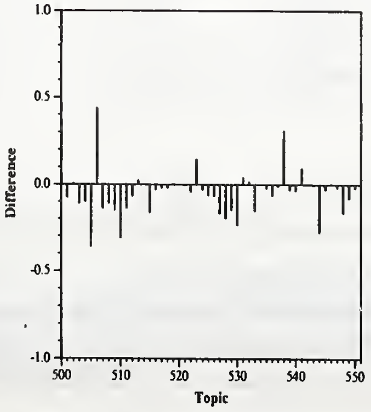
Mean average precision	
non-interpolated	0.0883

Document Level Averages	
	Precision
At 5 docs	0.2080
At 10 docs	0.1780
At 15 docs	0.1773
At 20 docs	0.1730
At 30 docs	0.1513
At 100 docs	0.0988
At 200 docs	0.0716
At 500 docs	0.0428
At 1000 docs	0.0269

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1360



Recall-Precision Curve



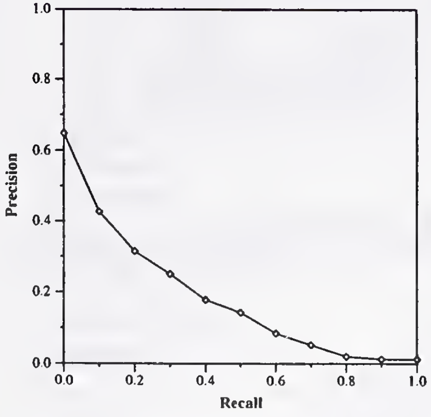
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	Merxtd
Run Description	Automatic, adhoc, title+desc; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

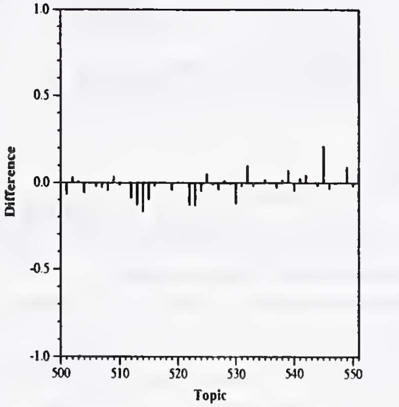
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2121

Recall Level Averages	
Recall	Precision
0.00	0.6481
0.10	0.4269
0.20	0.3142
0.30	0.2505
0.40	0.1782
0.50	0.1427
0.60	0.0854
0.70	0.0528
0.80	0.0206
0.90	0.0129
1.00	0.0123
Mean average precision	
non-interpolated	0.1729

Document Level Averages	
	Precision
At 5 docs	0.4360
At 10 docs	0.3660
At 15 docs	0.3320
At 20 docs	0.2960
At 30 docs	0.2560
At 100 docs	0.1558
At 200 docs	0.1108
At 500 docs	0.0696
At 1000 docs	0.0424
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2002



Recall-Precision Curve



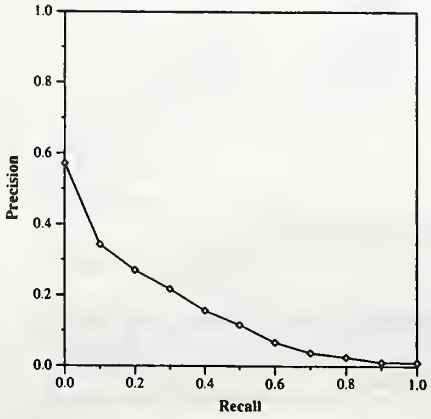
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	Merxt
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

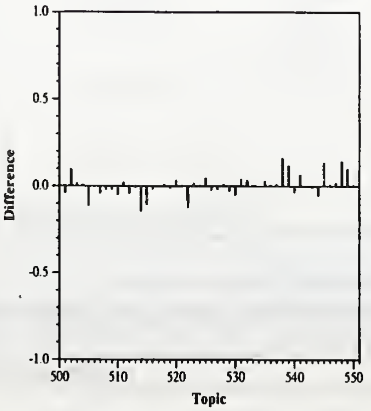
Total number of documents over all topics	
Retrieved:	49413
Relevant:	3363
Rel-ret:	1996

Recall Level Averages	
Recall	Precision
0.00	0.5717
0.10	0.3428
0.20	0.2699
0.30	0.2173
0.40	0.1569
0.50	0.1170
0.60	0.0669
0.70	0.0372
0.80	0.0254
0.90	0.0110
1.00	0.0104
Mean average precision	
non-interpolated	0.1438

Document Level Averages	
	Precision
At 5 docs	0.3160
At 10 docs	0.2880
At 15 docs	0.2653
At 20 docs	0.2480
At 30 docs	0.2247
At 100 docs	0.1376
At 200 docs	0.0999
At 500 docs	0.0619
At 1000 docs	0.0399
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1969



Recall-Precision Curve



Difference from Median in Average Precision per Topic

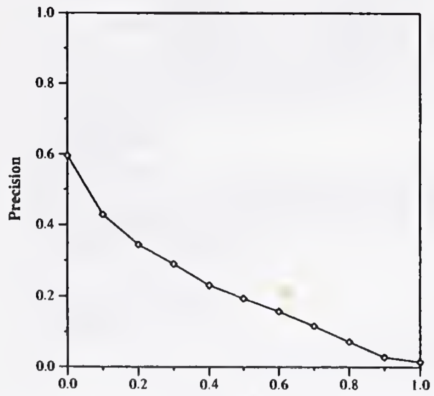
Web track, ad hoc task results — Johns Hopkins University, APL

Summary Statistics	
Run ID:	apl10wd
Run Description	Automatic, adhoc, title+desc+narr; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

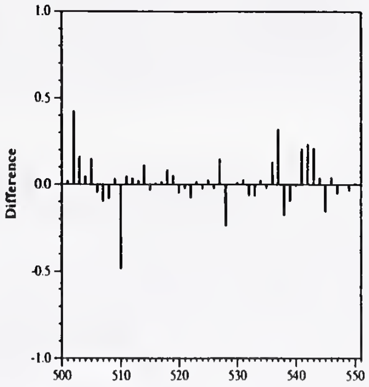
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2525

Recall Level Averages	
Recall	Precision
0.00	0.5953
0.10	0.4283
0.20	0.3447
0.30	0.2904
0.40	0.2309
0.50	0.1941
0.60	0.1579
0.70	0.1169
0.80	0.0718
0.90	0.0280
1.00	0.0142
Mean average precision	
non-interpolated	0.2035

Document Level Averages	
	Precision
At 5 docs	0.3720
At 10 docs	0.3380
At 15 docs	0.3213
At 20 docs	0.3050
At 30 docs	0.2827
At 100 docs	0.1956
At 200 docs	0.1406
At 500 docs	0.0826
At 1000 docs	0.0505
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2286



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	apl10wa
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

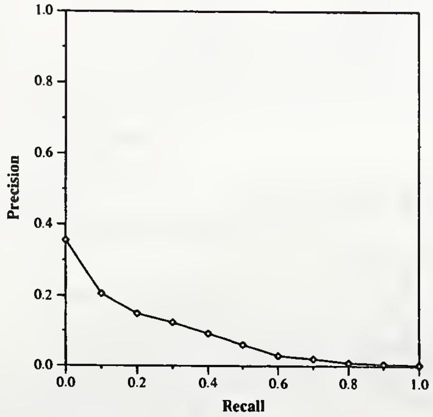
Total number of documents over all topics	
Retrieved:	49620
Relevant:	3363
Rel-ret:	1702

Recall Level Averages	
Recall	Precision
0.00	0.3549
0.10	0.2066
0.20	0.1494
0.30	0.1238
0.40	0.0921
0.50	0.0604
0.60	0.0292
0.70	0.0203
0.80	0.0093
0.90	0.0055
1.00	0.0029

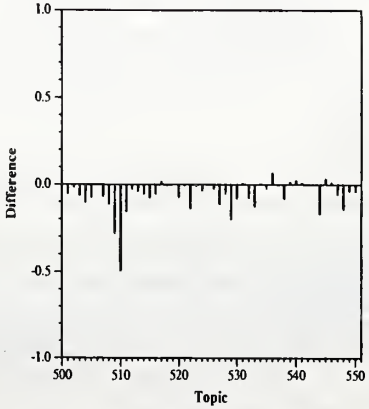
Mean average precision	
non-interpolated	0.0805

Document Level Averages	
	Precision
At 5 docs	0.1600
At 10 docs	0.1460
At 15 docs	0.1427
At 20 docs	0.1380
At 30 docs	0.1333
At 100 docs	0.0972
At 200 docs	0.0757
At 500 docs	0.0497
At 1000 docs	0.0340

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1189



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	apl10wb
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

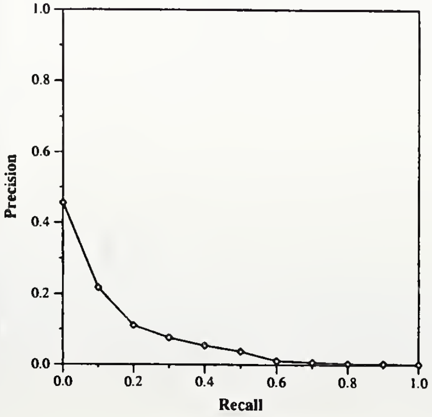
Total number of documents over all topics	
Retrieved:	17680
Relevant:	3363
Rel-ret:	599

Recall Level Averages	
Recall	Precision
0.00	0.4563
0.10	0.2182
0.20	0.1107
0.30	0.0761
0.40	0.0541
0.50	0.0375
0.60	0.0115
0.70	0.0069
0.80	0.0035
0.90	0.0035
1.00	0.0014

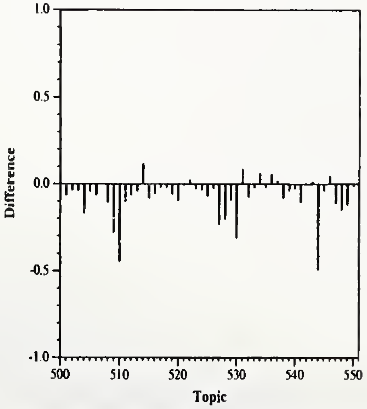
Mean average precision	
non-interpolated	0.0671

Document Level Averages	
	Precision
At 5 docs	0.2400
At 10 docs	0.1900
At 15 docs	0.1720
At 20 docs	0.1580
At 30 docs	0.1333
At 100 docs	0.0696
At 200 docs	0.0423
At 500 docs	0.0210
At 1000 docs	0.0120

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1073



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Web track, ad hoc task results — Johns Hopkins University, APL

Summary Statistics	
Run ID:	apl10wc
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

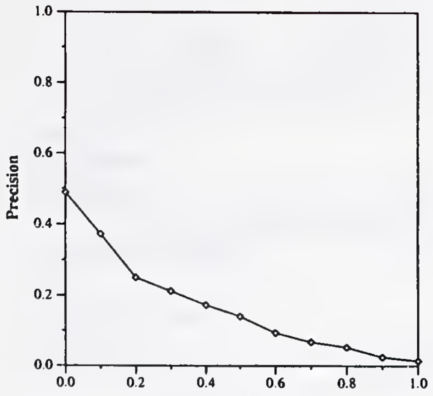
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2105

Recall Level Averages	
Recall	Precision
0.00	0.4890
0.10	0.3723
0.20	0.2513
0.30	0.2131
0.40	0.1729
0.50	0.1409
0.60	0.0940
0.70	0.0675
0.80	0.0526
0.90	0.0249
1.00	0.0143

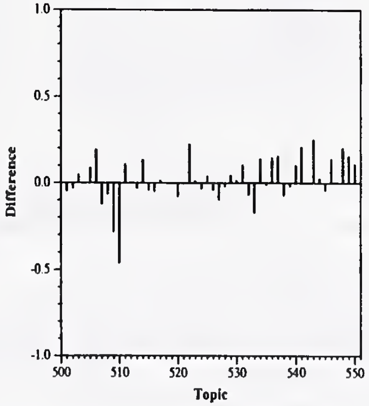
Mean average precision	
non-interpolated	0.1567

Document Level Averages	
	Precision
At 5 docs	0.2520
At 10 docs	0.2380
At 15 docs	0.2227
At 20 docs	0.2200
At 30 docs	0.1953
At 100 docs	0.1412
At 200 docs	0.1051
At 500 docs	0.0656
At 1000 docs	0.0421

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1972



Recall-Precision Curve



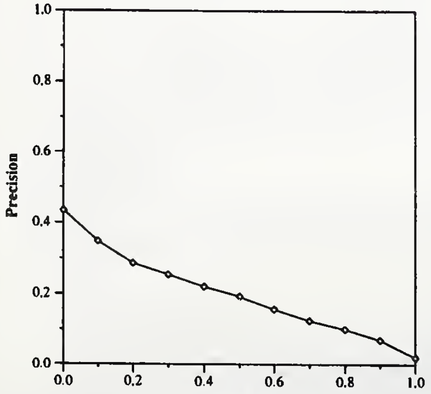
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	jscbtawtl1
Run Description	Automatic, title only; docstruct-used, urltext-notused, links-used
Number of Topics:	50

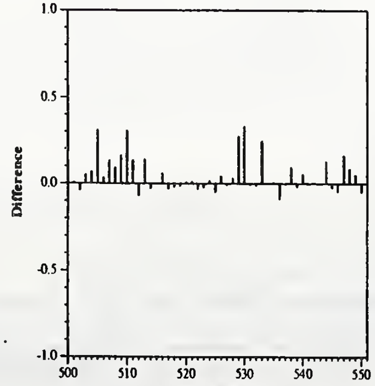
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2229

Recall Level Averages	
Recall	Precision
0.00	0.4340
0.10	0.3479
0.20	0.2868
0.30	0.2545
0.40	0.2207
0.50	0.1934
0.60	0.1571
0.70	0.1248
0.80	0.1003
0.90	0.0700
1.00	0.0196
Mean average precision	
non-interpolated	0.1890

Document Level Averages	
	Precision
At 5 docs	0.2960
At 10 docs	0.2980
At 15 docs	0.2733
At 20 docs	0.2530
At 30 docs	0.2213
At 100 docs	0.1584
At 200 docs	0.1176
At 500 docs	0.0714
At 1000 docs	0.0446
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2020



Recall-Precision Curve



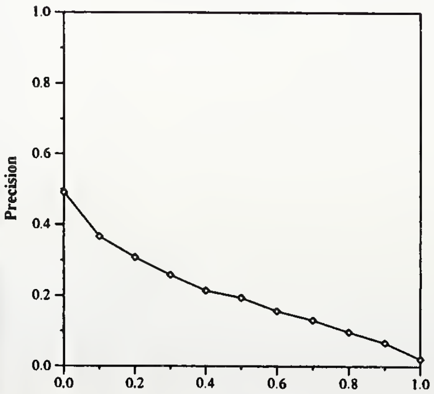
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	jscbtawtl2
Run Description	Automatic, title only; docstruct-used, urltext-notused, links-used
Number of Topics:	50

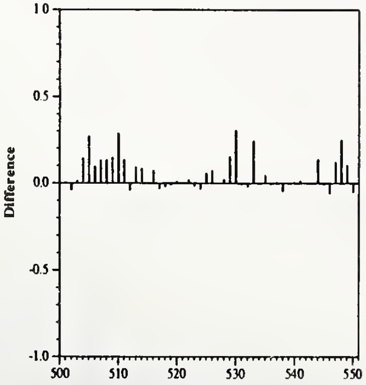
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2261

Recall Level Averages	
Recall	Precision
0.00	0.4906
0.10	0.3664
0.20	0.3082
0.30	0.2582
0.40	0.2144
0.50	0.1942
0.60	0.1569
0.70	0.1312
0.80	0.0976
0.90	0.0672
1.00	0.0208
Mean average precision	
non-interpolated	0.1954

Document Level Averages	
	Precision
At 5 docs	0.3360
At 10 docs	0.3000
At 15 docs	0.2867
At 20 docs	0.2760
At 30 docs	0.2340
At 100 docs	0.1630
At 200 docs	0.1200
At 500 docs	0.0726
At 1000 docs	0.0452
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2150



Recall-Precision Curve



Difference from Median in Average Precision per Topic

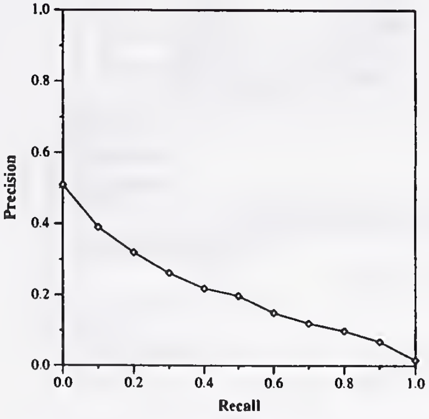
Web track, ad hoc task results — Justsystem Corporation

Summary Statistics	
Run ID:	jscbtawtl3
Run Description	Automatic, title only; docstruct-used, urltext-notused, links-used
Number of Topics:	50

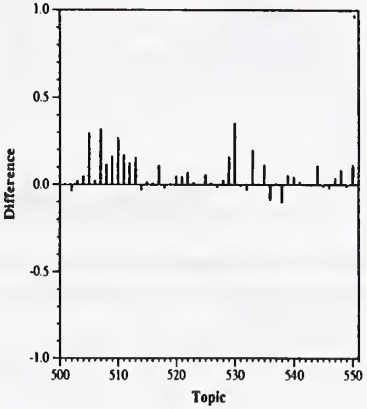
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2293

Recall Level Averages	
Recall	Precision
0.00	0.5087
0.10	0.3903
0.20	0.3202
0.30	0.2624
0.40	0.2192
0.50	0.1983
0.60	0.1508
0.70	0.1206
0.80	0.0990
0.90	0.0687
1.00	0.0155
Mean average precision	
non-interpolated	0.2003

Document Level Averages	
	Precision
At 5 docs	0.3360
At 10 docs	0.3120
At 15 docs	0.2960
At 20 docs	0.2750
At 30 docs	0.2507
At 100 docs	0.1690
At 200 docs	0.1228
At 500 docs	0.0741
At 1000 docs	0.0459
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2226



Recall-Precision Curve



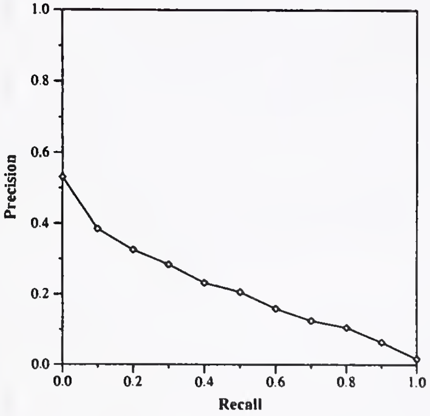
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	jscbtawtl4
Run Description	Automatic, title only; docstruct-used, urltext-notused, links-used
Number of Topics:	50

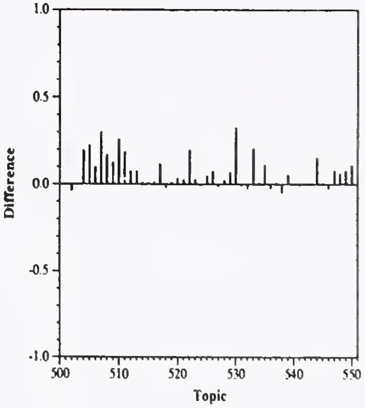
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2347

Recall Level Averages	
Recall	Precision
0.00	0.5301
0.10	0.3848
0.20	0.3260
0.30	0.2847
0.40	0.2327
0.50	0.2069
0.60	0.1602
0.70	0.1252
0.80	0.1061
0.90	0.0644
1.00	0.0168
Mean average precision	
non-interpolated	0.2060

Document Level Averages	
	Precision
At 5 docs	0.3440
At 10 docs	0.3140
At 15 docs	0.2920
At 20 docs	0.2830
At 30 docs	0.2540
At 100 docs	0.1688
At 200 docs	0.1261
At 500 docs	0.0773
At 1000 docs	0.0469
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2308



Recall-Precision Curve



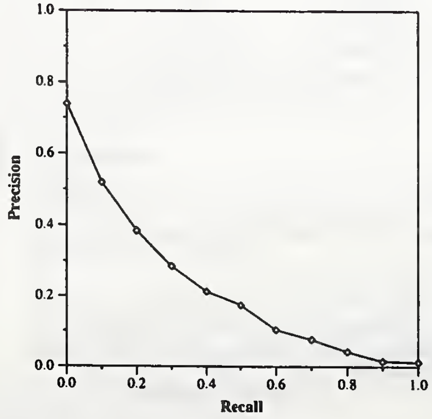
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	kuadhoc2001
Run Description	Automatic, adhoc, title+desc+narr; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

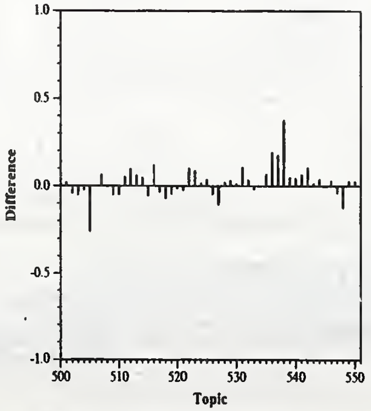
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2247

Recall Level Averages	
Recall	Precision
0.00	0.7386
0.10	0.5189
0.20	0.3843
0.30	0.2849
0.40	0.2127
0.50	0.1738
0.60	0.1033
0.70	0.0759
0.80	0.0419
0.90	0.0150
1.00	0.0124
Mean average precision	
non-interpolated	0.2088

Document Level Averages	
	Precision
At 5 docs	0.4760
At 10 docs	0.4100
At 15 docs	0.3760
At 20 docs	0.3380
At 30 docs	0.2840
At 100 docs	0.1770
At 200 docs	0.1233
At 500 docs	0.0722
At 1000 docs	0.0449
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2374



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	msrcn1
Run Description	Automatic, title only; docstruct-used, urltext-used, links-notused
Number of Topics:	50

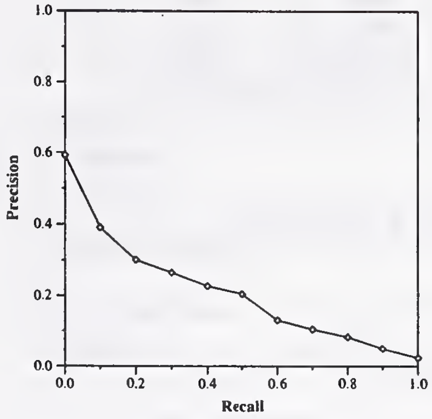
Total number of documents over all topics	
Retrieved:	49639
Relevant:	3363
Rel-ret:	2253

Recall Level Averages	
Recall	Precision
0.00	0.5923
0.10	0.3912
0.20	0.3012
0.30	0.2655
0.40	0.2271
0.50	0.2046
0.60	0.1301
0.70	0.1045
0.80	0.0827
0.90	0.0501
1.00	0.0239

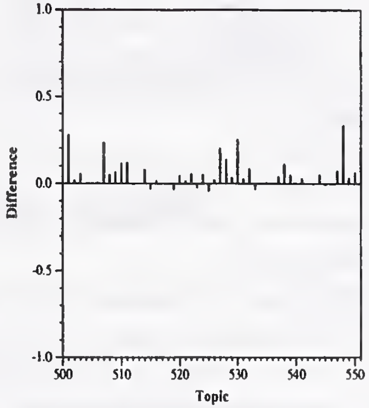
Mean average precision	
non-interpolated	0.1913

Document Level Averages	
	Precision
At 5 docs	0.3520
At 10 docs	0.3080
At 15 docs	0.2880
At 20 docs	0.2740
At 30 docs	0.2447
At 100 docs	0.1590
At 200 docs	0.1171
At 500 docs	0.0717
At 1000 docs	0.0451

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2263



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	msrcn2
Run Description	Automatic, title only; docstruct-used, urltext-used, links-used
Number of Topics:	50

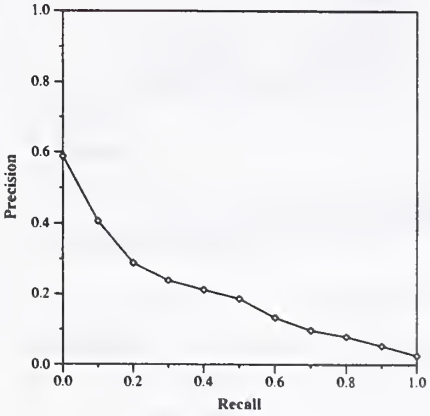
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2180

Recall Level Averages	
Recall	Precision
0.00	0.5877
0.10	0.4060
0.20	0.2884
0.30	0.2385
0.40	0.2121
0.50	0.1868
0.60	0.1333
0.70	0.0976
0.80	0.0793
0.90	0.0531
1.00	0.0252

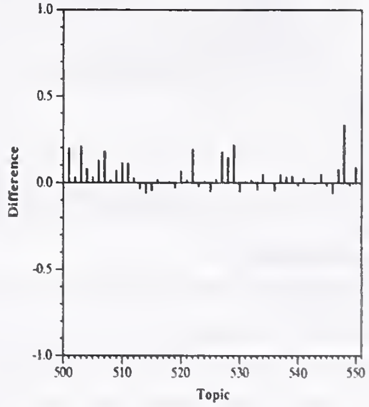
Mean average precision	
non-interpolated	0.1864

Document Level Averages	
	Precision
At 5 docs	0.3600
At 10 docs	0.3160
At 15 docs	0.2973
At 20 docs	0.2620
At 30 docs	0.2287
At 100 docs	0.1414
At 200 docs	0.1130
At 500 docs	0.0687
At 1000 docs	0.0436

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2095



Recall-Precision Curve



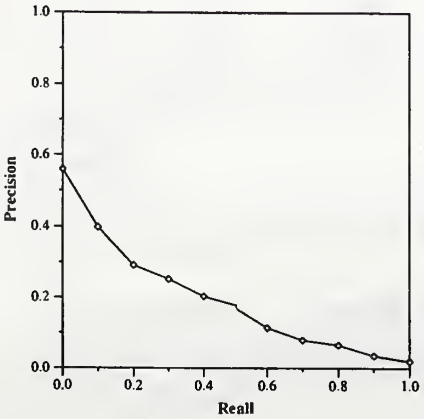
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	msrcn3
Run Description	Automatic, title only; docstruct-used, urltext-used, links-notused
Number of Topics:	50

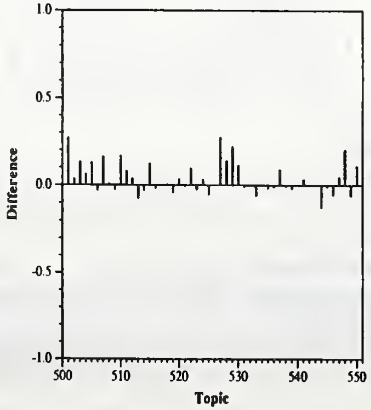
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	1997

Recall Level Averages	
Recall	Precision
0.00	0.5584
0.10	0.3985
0.20	0.2916
0.30	0.2522
0.40	0.2033
0.50	0.1757
0.60	0.1141
0.70	0.0794
0.80	0.0656
0.90	0.0355
1.00	0.0194
Mean average precision	
non-interpoated	0.1779

Document Level Averages	
	Precision
At 5 docs	0.3320
At 10 docs	0.3060
At 15 docs	0.2827
At 20 docs	0.2660
At 30 docs	0.2407
At 100 docs	0.1430
At 200 docs	0.1030
At 500 docs	0.0609
At 1000 docs	0.0399
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2152



Recall-Precsion Curve



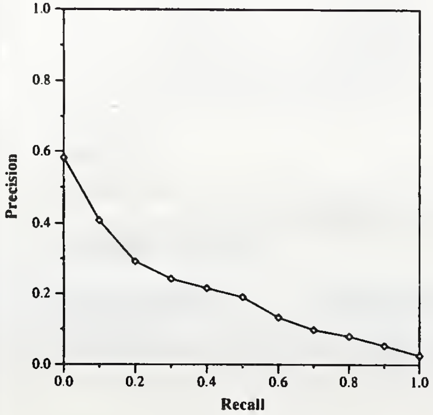
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	msrcn4
Run Description	Automatic, title only; docstruct-used, urltext-used, links-notused
Number of Topics:	50

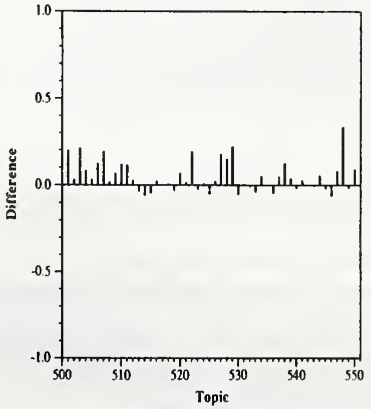
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2180

Recall Level Averages	
Recall	Precision
0.00	0.5816
0.10	0.4080
0.20	0.2921
0.30	0.2425
0.40	0.2160
0.50	0.1907
0.60	0.1335
0.70	0.0983
0.80	0.0800
0.90	0.0537
1.00	0.0258
Mean average precision	
non-interpolated	0.1880

Document Level Averages	
	Precision
At 5 docs	0.3560
At 10 docs	0.3160
At 15 docs	0.2987
At 20 docs	0.2620
At 30 docs	0.2300
At 100 docs	0.1412
At 200 docs	0.1130
At 500 docs	0.0686
At 1000 docs	0.0436
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2193



Recall-Precision Curve



Difference from Median in Average Precision per Topic

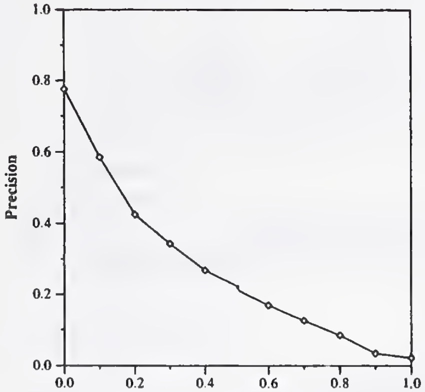
Web track, ad ho: task results — Microsoft Research Ltd

Summary Statistics	
Run ID:	ok10wtnd0
Run Description	Automatic, adhoc, title+desc+narr; docstruct-notused, urltext-used, links-notused
Number of Topic:	50

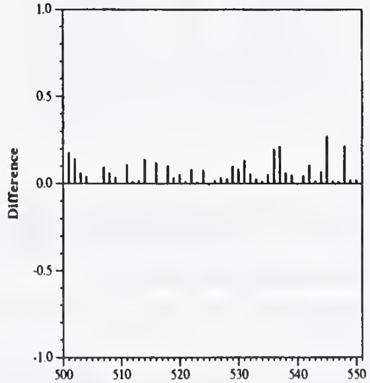
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2586

Recall Lvel Averages	
Recall	Precision
0.00	0.7755
0.10	0.5854
0.20	0.4245
0.30	0.3434
0.40	0.2687
0.50	0.2191
0.60	0.1693
0.70	0.1255
0.80	0.0846
0.90	0.0342
1.00	0.0213
Mean average precision	
non-interpoated	0.2512

Document Level Averages	
	Precision
At 5 docs	0.5360
At 10 docs	0.4540
At 15 docs	0.3960
At 20 docs	0.3590
At 30 docs	0.3193
At 100 docs	0.2042
At 200 docs	0.1456
At 500 docs	0.0843
At 1000 docs	0.0517
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2806



Recall-Precsion Curve



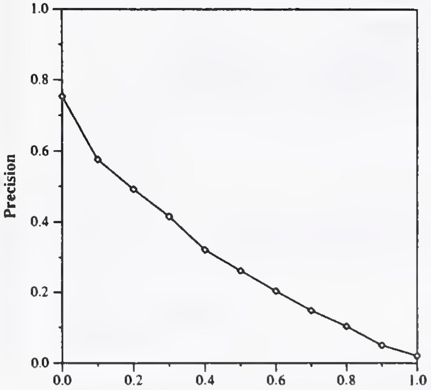
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	ok10wtnd1
Run Description	Automatic, adhoc, title+desc+narr; docstruct-notused, urltext-used, links-notused
Number of Topics:	50

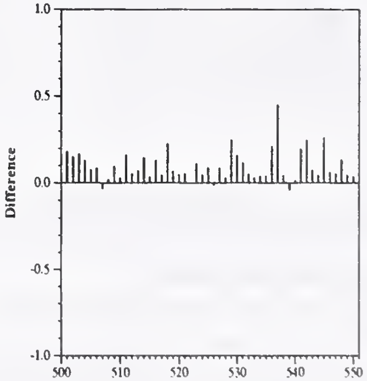
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2689

Recall Level Averages	
Recall	Precision
0.00	0.7532
0.10	0.5761
0.20	0.4915
0.30	0.4157
0.40	0.3220
0.50	0.2620
0.60	0.2039
0.70	0.1480
0.80	0.1034
0.90	0.0499
1.00	0.0202
Mean average precision	
non-interpolated	0.2831

Document Level Averages	
	Precision
At 5 docs	0.5560
At 10 docs	0.4680
At 15 docs	0.4427
At 20 docs	0.4010
At 30 docs	0.3527
At 100 docs	0.2236
At 200 docs	0.1588
At 500 docs	0.0911
At 1000 docs	0.0538
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3023



Recall-Precision Curve



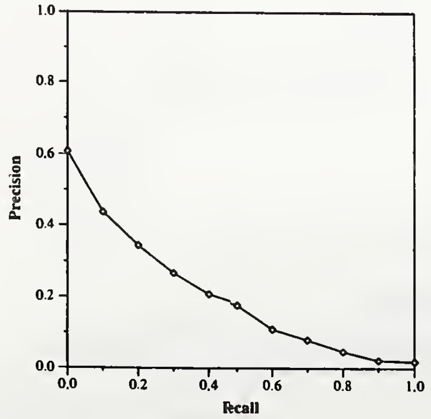
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	ok10wt1
Run Descriptior	Automatic, title only; docstruct-notused, urltext-used, links-notused
Number of Topi:s:	50

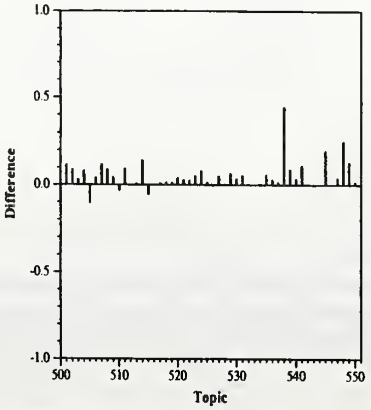
Total number of documents over all topics	
Retrieved:	49650
Relevant:	3363
Rel-ret:	2333

Recall level Averages	
Recal	Precision
0.00	0.6082
0.10	0.4370
0.20	0.3442
0.30	0.2671
0.40	0.2076
0.50	0.1767
0.60	0.1094
0.70	0.0795
0.80	0.0471
0.90	0.0221
1.00	0.0187
Mean averge precision	
non-interplated	0.1908

Document Level Averages	
	Precision
At 5 docs	0.3600
At 10 docs	0.3360
At 15 docs	0.3080
At 20 docs	0.2950
At 30 docs	0.2613
At 100 docs	0.1678
At 200 docs	0.1230
At 500 docs	0.0749
At 1000 docs	0.0467
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2235



Recall-Prcision Curve



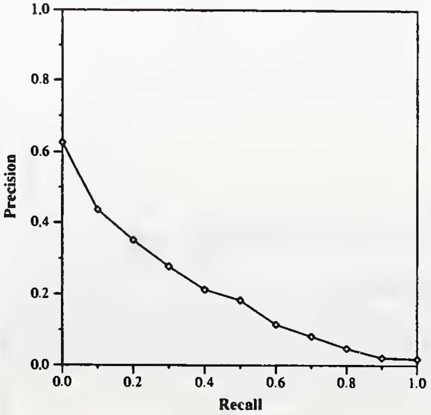
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	ok10wt3
Run Description	Automatic, title only; docstruct-notused, urltext-used, links-notused
Number of Topics:	50

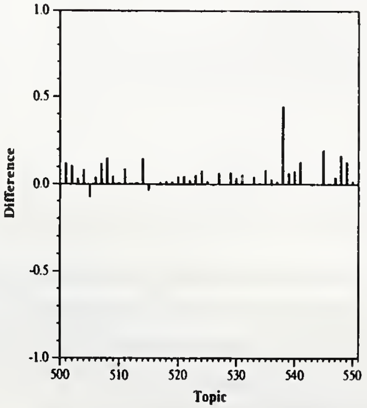
Total number of documents over all topics	
Retrieved:	49650
Relevant:	3363
Rel-ret:	2359

Recall Level Averages	
Recall	Precision
0.00	0.6264
0.10	0.4361
0.20	0.3512
0.30	0.2777
0.40	0.2127
0.50	0.1828
0.60	0.1152
0.70	0.0818
0.80	0.0479
0.90	0.0216
1.00	0.0178
Mean average precision	
non-interpolated	0.1952

Document Level Averages	
	Precision
At 5 docs	0.3840
At 10 docs	0.3360
At 15 docs	0.3213
At 20 docs	0.3040
At 30 docs	0.2667
At 100 docs	0.1686
At 200 docs	0.1251
At 500 docs	0.0762
At 1000 docs	0.0472
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2196



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Web track, ad hcc task results — NexTrieve BV

Summary Statistics	
Run ID:	Ntvenx1
Run Descriptior	Automatic, title only; docstruct-used, urltext-notused, links-notused
Number of Topics:	50

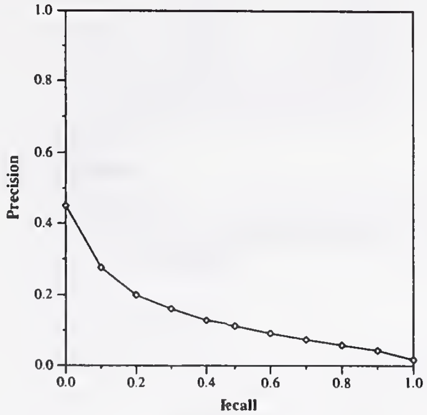
Total number of documents over all topics	
Retrieved:	49662
Relevant:	3363
Rel-ret:	1775

Recall Level Averages	
Recal	Precision
0.00	0.4503
0.10	0.2763
0.20	0.2002
0.30	0.1615
0.40	0.1288
0.50	0.1125
0.60	0.0920
0.70	0.0741
0.80	0.0584
0.90	0.0431
1.00	0.0169

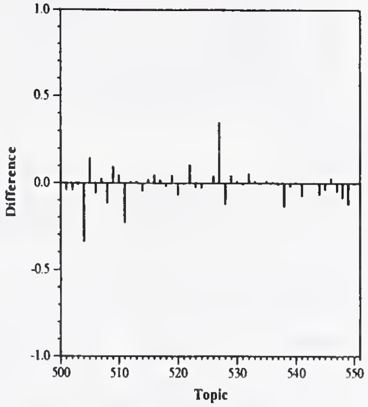
Mean avenge precision	
non-interplated	0.1273

Document Level Averages	
	Precision
At 5 docs	0.2720
At 10 docs	0.2540
At 15 docs	0.2240
At 20 docs	0.2080
At 30 docs	0.1940
At 100 docs	0.1176
At 200 docs	0.0820
At 500 docs	0.0508
At 1000 docs	0.0355

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1511



Recall-Prcision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	Ntvenx2
Run Description	Automatic, title only; docstruct-used, urltext-notused, links-notused
Number of Topics:	50

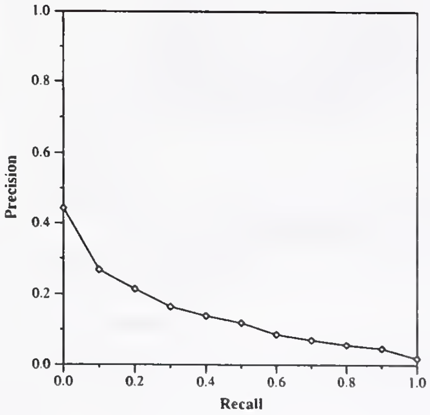
Total number of documents over all topics	
Retrieved:	49662
Relevant:	3363
Rel-ret:	1665

Recall Level Averages	
Recall	Precision
0.00	0.4430
0.10	0.2676
0.20	0.2147
0.30	0.1652
0.40	0.1394
0.50	0.1192
0.60	0.0868
0.70	0.0707
0.80	0.0570
0.90	0.0469
1.00	0.0187

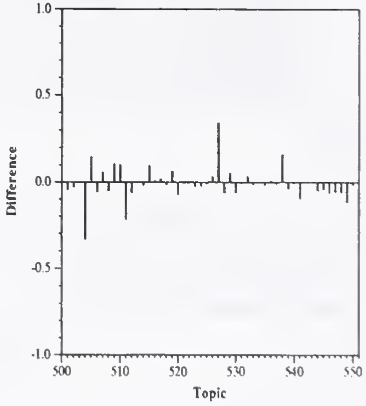
Mean average precision	
non-interpolated	0.1313

Document Level Averages	
	Precision
At 5 docs	0.2840
At 10 docs	0.2460
At 15 docs	0.2240
At 20 docs	0.2080
At 30 docs	0.1913
At 100 docs	0.1194
At 200 docs	0.0851
At 500 docs	0.0506
At 1000 docs	0.0333

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1555



Recall-Precision Curve



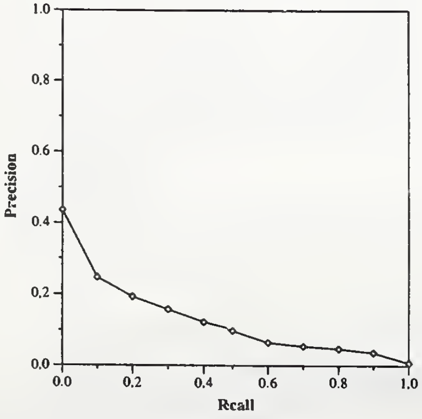
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	Ntvfnx3
Run Descriptior	Automatic, title only; docstruct-used, urltext-notused, links-notused
Number of Topis:	50

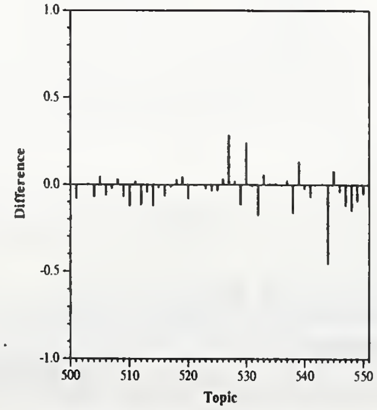
Total number of documents over all topics	
Retrieved:	43390
Relevant:	3363
Rel-ret:	1504

Recall level Averages	
Recal	Precision
0.00	0.4359
0.10	0.2469
0.20	0.1931
0.30	0.1588
0.40	0.1233
0.50	0.0983
0.60	0.0652
0.70	0.0548
0.80	0.0478
0.90	0.0366
1.00	0.0078
Mean averge precision	
non-interpolated	0.1128

Document Level Averages	
	Precision
At 5 docs	0.2200
At 10 docs	0.2120
At 15 docs	0.1933
At 20 docs	0.1920
At 30 docs	0.1740
At 100 docs	0.1194
At 200 docs	0.0863
At 500 docs	0.0516
At 1000 docs	0.0301
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1405



Recall-Preision Curve



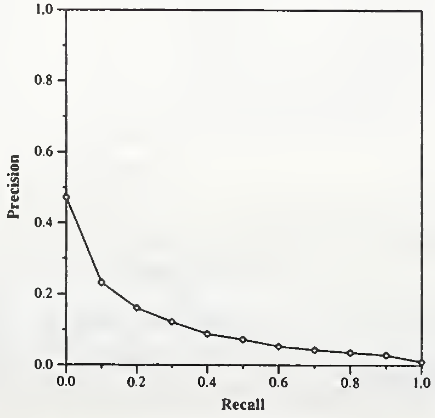
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	Ntvfnx4
Run Description	Automatic, title only; docstruct-used, urltext-notused, links-notused
Number of Topics:	50

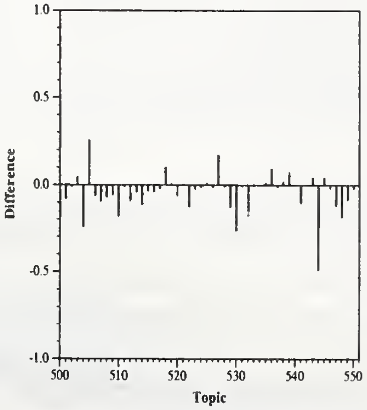
Total number of documents over all topics	
Retrieved:	44191
Relevant:	3363
Rel-ret:	1290

Recall Level Averages	
Recall	Precision
0.00	0.4725
0.10	0.2313
0.20	0.1615
0.30	0.1221
0.40	0.0884
0.50	0.0726
0.60	0.0536
0.70	0.0432
0.80	0.0357
0.90	0.0289
1.00	0.0097
Mean average precision	
non-interpolated	0.0978

Document Level Averages	
	Precision
At 5 docs	0.2480
At 10 docs	0.2180
At 15 docs	0.1893
At 20 docs	0.1770
At 30 docs	0.1613
At 100 docs	0.1008
At 200 docs	0.0766
At 500 docs	0.0430
At 1000 docs	0.0258
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1249



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Web track, ad hoc task results — Pohang University of Science and Technology (POSTECH)

Summary Statistics	
Run ID:	posnir01ptd
Run Description	Automatic, adhoc, title+desc; docstruct-used, urltext-notused, links-notused
Number of Topis:	50

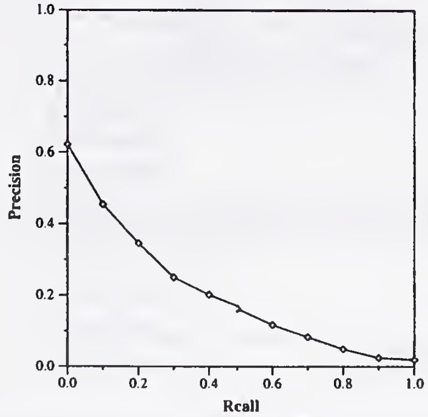
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2225

Recall Level Averages	
Recall	Precision
0.00	0.6216
0.10	0.4548
0.20	0.3464
0.30	0.2509
0.40	0.2026
0.50	0.1637
0.60	0.1182
0.70	0.0842
0.80	0.0498
0.90	0.0249
1.00	0.0193

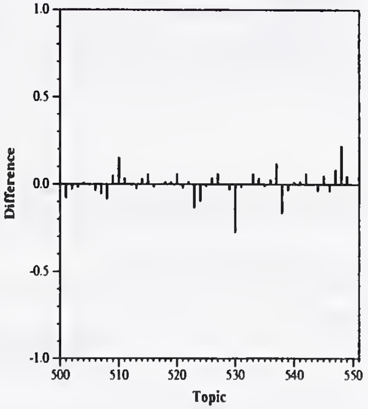
Mean averge precision	
non-interpdated	0.1877

Document Level Averages	
	Precision
At 5 docs	0.3880
At 10 docs	0.3280
At 15 docs	0.3160
At 20 docs	0.2950
At 30 docs	0.2660
At 100 docs	0.1762
At 200 docs	0.1279
At 500 docs	0.0726
At 1000 docs	0.0445

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2240



Recall-Preision Curve



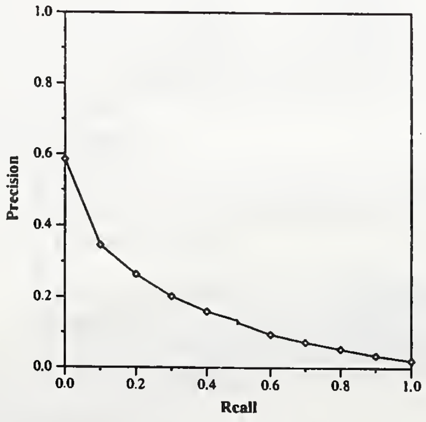
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	posnir01pt
Run Description	Automatic, title only; docstruct-used, urltext-notused, links-notused
Number of Topics:	50

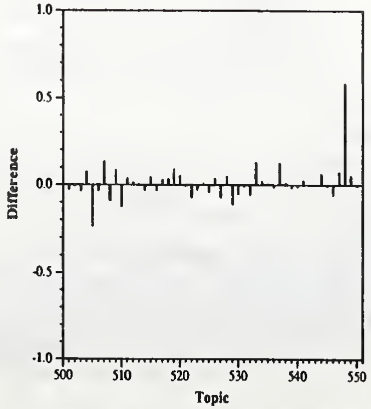
Total number of documents over all topics	
Retrieved:	48825
Relevant:	3363
Rel-ret:	2111

Recall Level Averages	
Recall	Precision
0.00	0.5855
0.10	0.3452
0.20	0.2643
0.30	0.2022
0.40	0.1592
0.50	0.1307
0.60	0.0951
0.70	0.0730
0.80	0.0537
0.90	0.0351
1.00	0.0210
Mean average precision	
non-interpolated	0.1521

Document Level Averages	
	Precision
At 5 docs	0.3040
At 10 docs	0.2460
At 15 docs	0.2293
At 20 docs	0.2110
At 30 docs	0.1893
At 100 docs	0.1396
At 200 docs	0.1040
At 500 docs	0.0675
At 1000 docs	0.0422
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1760



Recall-Precision Curve



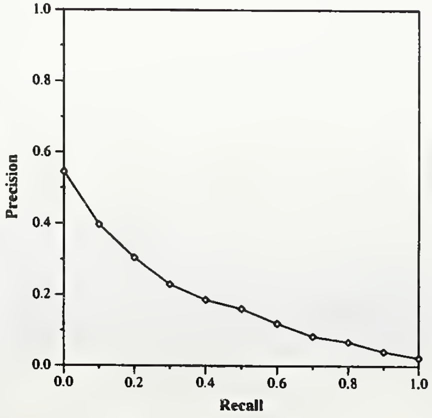
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	posnir01rpt
Run Description	Automatic, title only; docstruct-used, urltext-notused, links-notused
Number of Topics:	50

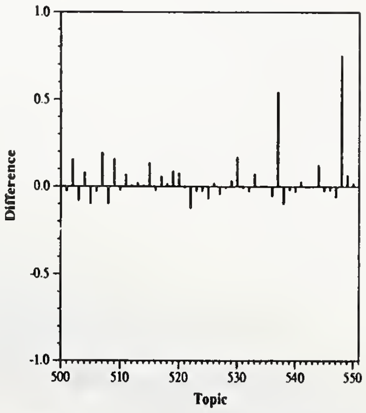
Total number of documents over all topics	
Retrieved:	49953
Relevant:	3363
Rel-ret:	2103

Recall Level Averages	
Recall	Precision
0.00	0.5451
0.10	0.3963
0.20	0.3050
0.30	0.2299
0.40	0.1861
0.50	0.1609
0.60	0.1202
0.70	0.0840
0.80	0.0680
0.90	0.0408
1.00	0.0230
Mean average precision	
non-interpolated	0.1771

Document Level Averages	
	Precision
At 5 docs	0.3640
At 10 docs	0.2880
At 15 docs	0.2627
At 20 docs	0.2480
At 30 docs	0.2220
At 100 docs	0.1470
At 200 docs	0.1071
At 500 docs	0.0672
At 1000 docs	0.0421
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2081



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Web track, ad ho‹ task results — Pohang University of Science and Technology (POSTECH)

Summary Statistics	
Run ID:	posnir01st
Run Description	Automatic, title only; docstruct-used, urltext-notused, links-notused
Number of Topic:	50

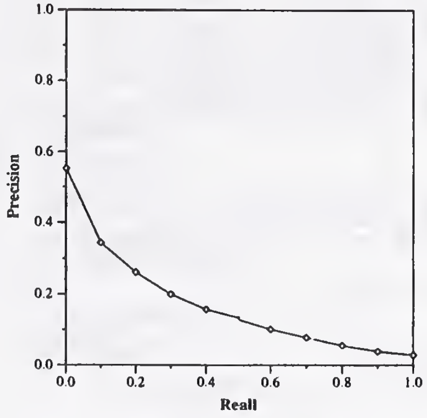
Total number of documents over all topics	
Retrieved:	48825
Relevant:	3363
Rel-ret:	2099

Recall L:vel Averages	
Recall	Precision
0.00	0.5517
0.10	0.3436
0.20	0.2621
0.30	0.2003
0.40	0.1578
0.50	0.1317
0.60	0.1015
0.70	0.0783
0.80	0.0553
0.90	0.0388
1.00	0.0286

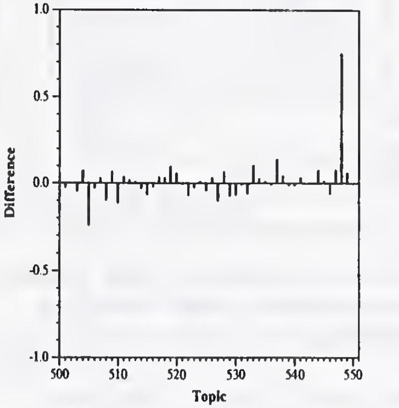
Mean avera;e precision	
non-interpoated	0.1535

Document Level Averages	
	Precision
At 5 docs	0.2960
At 10 docs	0.2480
At 15 docs	0.2387
At 20 docs	0.2220
At 30 docs	0.1987
At 100 docs	0.1370
At 200 docs	0.1045
At 500 docs	0.0674
At 1000 docs	0.0420

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1853



Recall-Precsion Curve



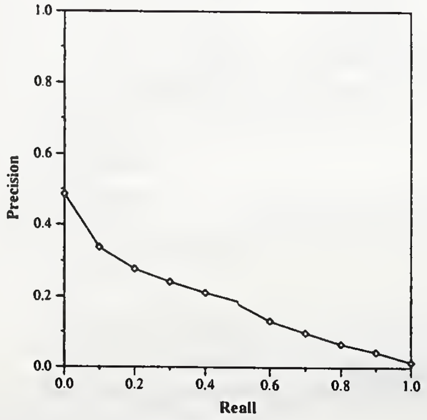
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	pir1Wa
Run Description	Automatic, adhoc, title+desc+narr; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

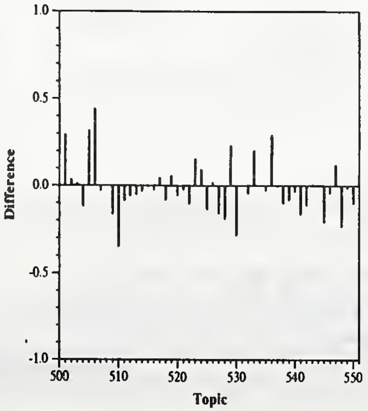
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2284

Recall Level Averages	
Recall	Precision
0.00	0.4852
0.10	0.3370
0.20	0.2778
0.30	0.2420
0.40	0.2115
0.50	0.1847
0.60	0.1319
0.70	0.0982
0.80	0.0662
0.90	0.0436
1.00	0.0145
Mean average precision	
non-interpoated	0.1715

Document Level Averages	
	Precision
At 5 docs	0.2680
At 10 docs	0.2780
At 15 docs	0.2493
At 20 docs	0.2370
At 30 docs	0.2220
At 100 docs	0.1488
At 200 docs	0.1114
At 500 docs	0.0698
At 1000 docs	0.0457
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1968



Recall-Precsion Curve



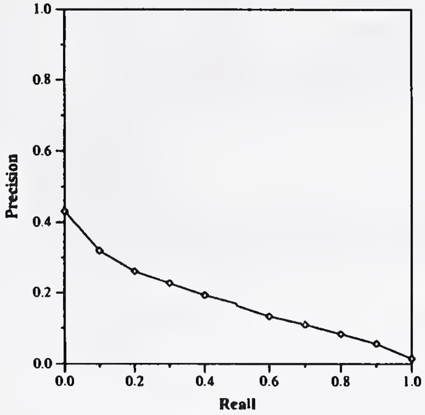
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	pir1Wt1
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topic:	50

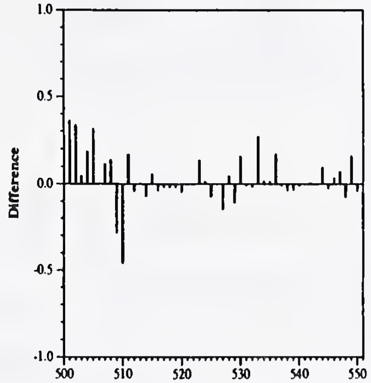
'otal number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2263

Recall Lvel Averages	
Recall	Precision
0.00	0.4313
0.10	0.3194
0.20	0.2612
0.30	0.2272
0.40	0.1927
0.50	0.1661
0.60	0.1328
0.70	0.1099
0.80	0.0834
0.90	0.0560
1.00	0.0140
Mean average precision	
non-interpoated	0.1660

Document Level Averages	
	Precision
At 5 docs	0.2360
At 10 docs	0.2220
At 15 docs	0.2227
At 20 docs	0.2070
At 30 docs	0.2013
At 100 docs	0.1468
At 200 docs	0.1152
At 500 docs	0.0717
At 1000 docs	0.0453
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1700



Recall-Precision Curve



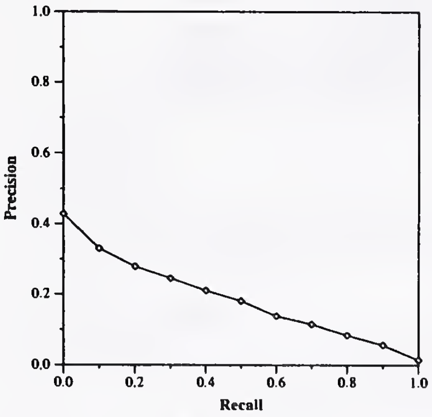
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	pir1Wt2
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

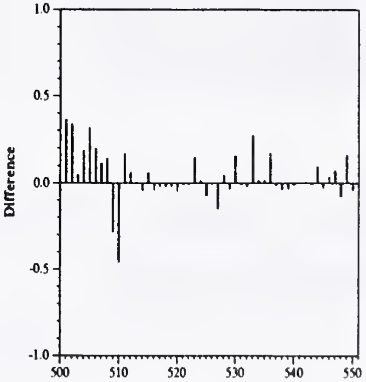
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2275

Recall Level Averages	
Recall	Precision
0.00	0.4279
0.10	0.3297
0.20	0.2792
0.30	0.2453
0.40	0.2103
0.50	0.1802
0.60	0.1380
0.70	0.1146
0.80	0.0840
0.90	0.0566
1.00	0.0141
Mean average precision	
non-interpolated	0.1742

Document Level Averages	
	Precision
At 5 docs	0.2320
At 10 docs	0.2160
At 15 docs	0.2227
At 20 docs	0.2110
At 30 docs	0.2040
At 100 docs	0.1520
At 200 docs	0.1181
At 500 docs	0.0724
At 1000 docs	0.0455
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1894



Recall-Precision Curve



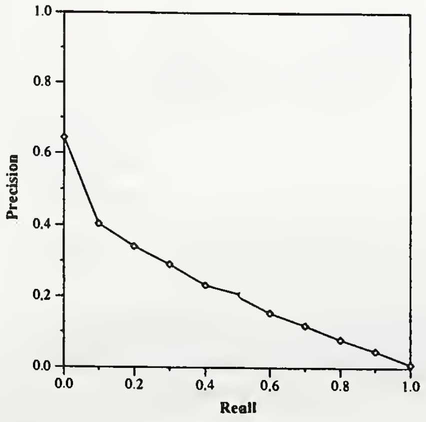
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	ricAP
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topic:	50

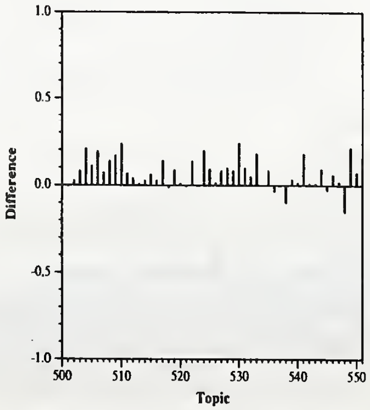
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2452

Recall Level Averages	
Recall	Precision
0.00	0.6434
0.10	0.4038
0.20	0.3413
0.30	0.2908
0.40	0.2323
0.50	0.2060
0.60	0.1544
0.70	0.1189
0.80	0.0792
0.90	0.0470
1.00	0.0088
Mean average precision	
non-interpoated	0.2077

Document Level Averages	
	Precision
At 5 docs	0.3800
At 10 docs	0.3380
At 15 docs	0.3053
At 20 docs	0.3000
At 30 docs	0.2773
At 100 docs	0.1762
At 200 docs	0.1326
At 500 docs	0.0797
At 1000 docs	0.0490
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2420



Recall-Precsion Curve



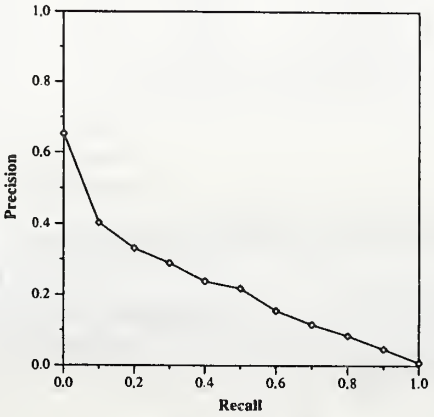
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	ricMM
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

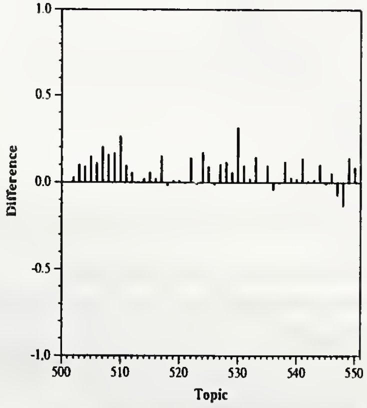
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2371

Recall Level Averages	
Recall	Precision
0.00	0.6520
0.10	0.4027
0.20	0.3311
0.30	0.2894
0.40	0.2380
0.50	0.2180
0.60	0.1555
0.70	0.1159
0.80	0.0843
0.90	0.0469
1.00	0.0084
Mean average precision	
non-interpolated	0.2084

Document Level Averages	
	Precision
At 5 docs	0.3880
At 10 docs	0.3420
At 15 docs	0.3040
At 20 docs	0.2950
At 30 docs	0.2647
At 100 docs	0.1684
At 200 docs	0.1261
At 500 docs	0.0754
At 1000 docs	0.0474
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2334



Recall-Precision Curve



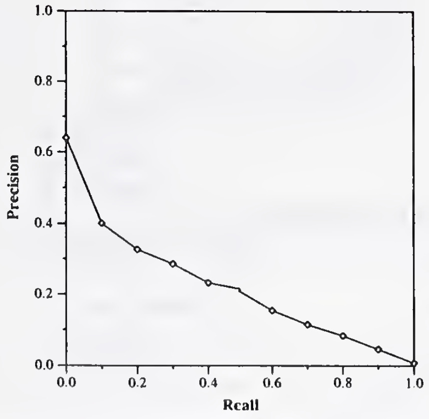
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	ricMS
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

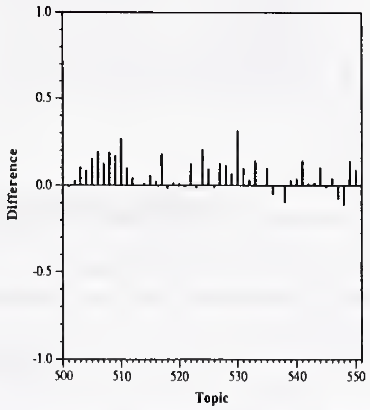
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2371

Recall Level Averages	
Recall	Precision
0.00	0.6396
0.10	0.4006
0.20	0.3262
0.30	0.2857
0.40	0.2320
0.50	0.2128
0.60	0.1545
0.70	0.1149
0.80	0.0833
0.90	0.0459
1.00	0.0074
Mean average precision	
non-interpolated	0.2068

Document Level Averages	
	Precision
At 5 docs	0.3800
At 10 docs	0.3360
At 15 docs	0.3107
At 20 docs	0.2950
At 30 docs	0.2653
At 100 docs	0.1684
At 200 docs	0.1260
At 500 docs	0.0754
At 1000 docs	0.0474
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2343



Recall-Precision Curve



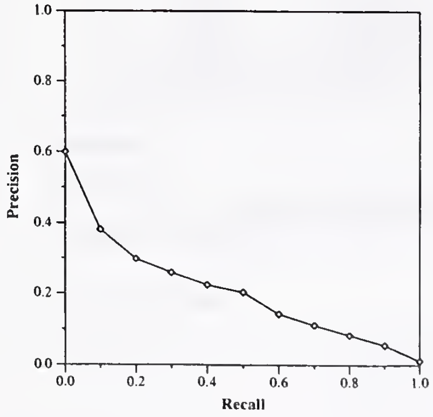
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	ricST
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

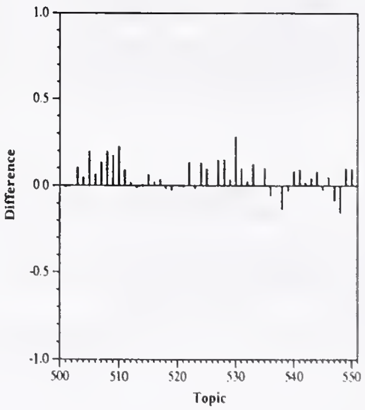
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2317

Recall Level Averages	
Recall	Precision
0.00	0.5993
0.10	0.3813
0.20	0.2977
0.30	0.2586
0.40	0.2240
0.50	0.2033
0.60	0.1429
0.70	0.1113
0.80	0.0834
0.90	0.0556
1.00	0.0118
Mean average precision	
non-interpolated	0.1933

Document Level Averages	
	Precision
At 5 docs	0.3640
At 10 docs	0.3260
At 15 docs	0.2947
At 20 docs	0.2720
At 30 docs	0.2433
At 100 docs	0.1620
At 200 docs	0.1215
At 500 docs	0.0730
At 1000 docs	0.0463
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2028



Recall-Precision Curve



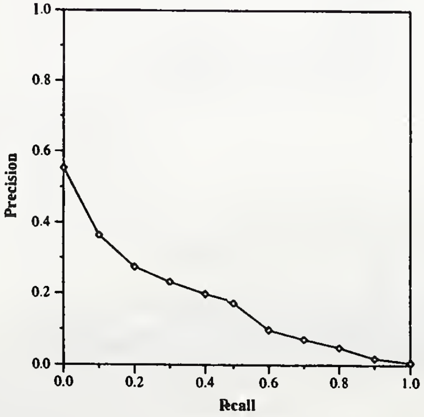
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	tnout10t1
Run Descriptior	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

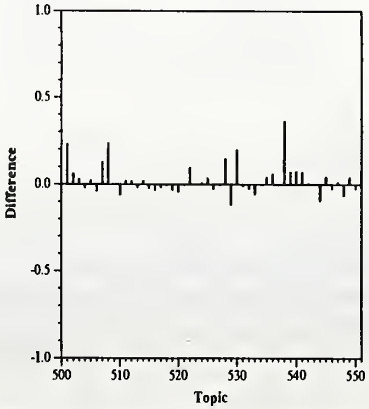
Total number of documents over all topics	
Retrieved:	48708
Relevant:	3363
Rel-ret:	2180

Recall level Averages	
Recal	Precision
0.00	0.5529
0.10	0.3637
0.20	0.2744
0.30	0.2324
0.40	0.1985
0.50	0.1725
0.60	0.0982
0.70	0.0717
0.80	0.0489
0.90	0.0184
1.00	0.0060
Mean averge precision	
non-interplated	0.1652

Document Level Averages	
	Precision
At 5 docs	0.3360
At 10 docs	0.2780
At 15 docs	0.2573
At 20 docs	0.2420
At 30 docs	0.2120
At 100 docs	0.1462
At 200 docs	0.1100
At 500 docs	0.0671
At 1000 docs	0.0436
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1988



Recall-Prcision Curve



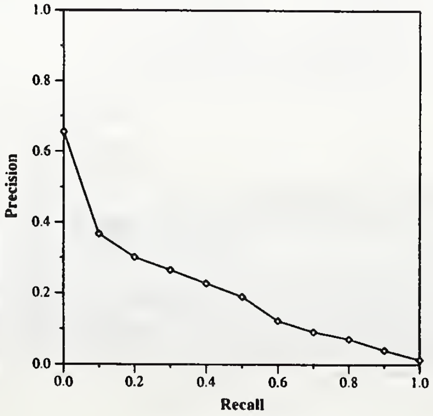
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	tnout10t2
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

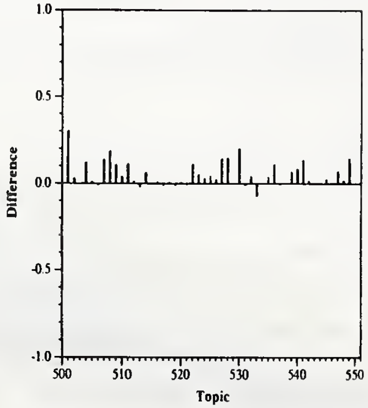
Total number of documents over all topics	
Retrieved:	48708
Relevant:	3363
Rel-ret:	2355

Recall Level Averages	
Recall	Precision
0.00	0.6541
0.10	0.3665
0.20	0.3010
0.30	0.2648
0.40	0.2273
0.50	0.1899
0.60	0.1225
0.70	0.0921
0.80	0.0716
0.90	0.0404
1.00	0.0133
Mean average precision	
non-interpolated	0.1891

Document Level Averages	
	Precision
At 5 docs	0.3640
At 10 docs	0.3340
At 15 docs	0.3027
At 20 docs	0.2720
At 30 docs	0.2393
At 100 docs	0.1666
At 200 docs	0.1241
At 500 docs	0.0768
At 1000 docs	0.0471
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2092



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UniNEn7d
Run Descriptior	Automatic, adhoc, title+desc+narr; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

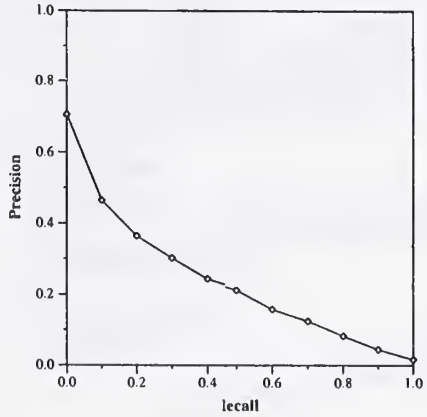
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	2439

Recall Level Averages	
Recal	Precision
0.00	0.7062
0.10	0.4653
0.20	0.3645
0.30	0.3022
0.40	0.2443
0.50	0.2123
0.60	0.1579
0.70	0.1245
0.80	0.0827
0.90	0.0447
1.00	0.0165

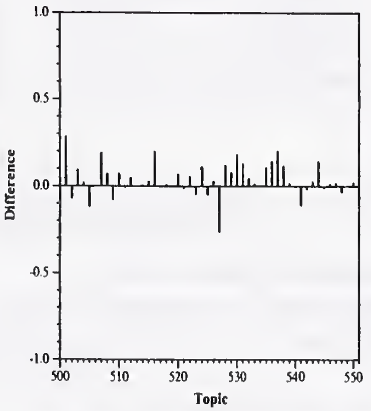
Mean avenge precision	
non-interplated	0.2242

Document Level Averages	
	Precision
At 5 docs	0.4360
At 10 docs	0.4060
At 15 docs	0.3493
At 20 docs	0.3280
At 30 docs	0.2847
At 100 docs	0.1752
At 200 docs	0.1307
At 500 docs	0.0758
At 1000 docs	0.0488

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2429



Recall-Prcision Curve



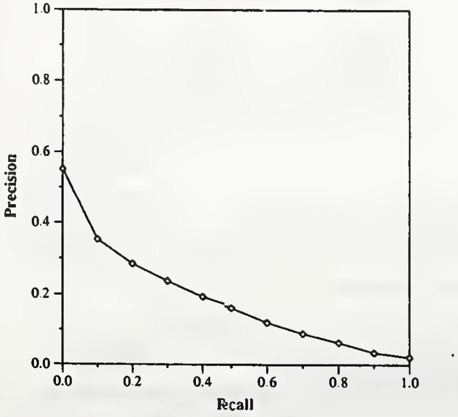
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UniNEt7dL
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

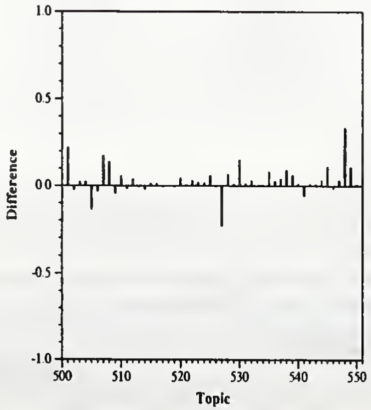
Total number of documents over all topics	
Retrieved:	49528
Relevant:	3363
Rel-ret:	2201

Recall Level Averages	
Recall	Precision
0.00	0.5508
0.10	0.3525
0.20	0.2851
0.30	0.2379
0.40	0.1940
0.50	0.1616
0.60	0.1201
0.70	0.0888
0.80	0.0631
0.90	0.0347
1.00	0.0215
Mean average precision	
non-interpolated	0.1699

Document Level Averages	
	Precision
At 5 docs	0.3200
At 10 docs	0.2640
At 15 docs	0.2547
At 20 docs	0.2470
At 30 docs	0.2280
At 100 docs	0.1466
At 200 docs	0.1086
At 500 docs	0.0660
At 1000 docs	0.0440
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2023



Recall-Precision Curve



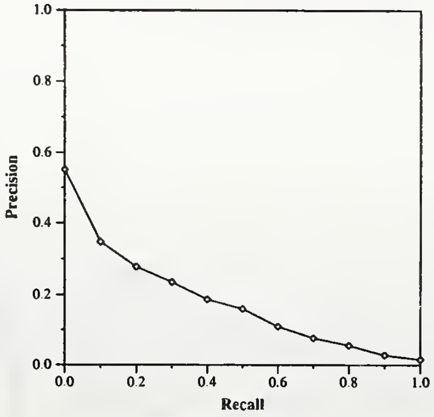
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UniNEtd
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

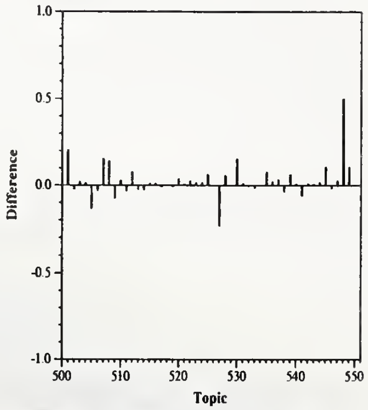
Total number of documents over all topics	
Retrieved:	49528
Relevant:	3363
Rel-ret:	2167

Recall Level Averages	
Recall	Precision
0.00	0.5508
0.10	0.3467
0.20	0.2781
0.30	0.2355
0.40	0.1874
0.50	0.1599
0.60	0.1101
0.70	0.0767
0.80	0.0557
0.90	0.0280
1.00	0.0155
Mean average precision	
non-interpolated	0.1659

Document Level Averages	
	Precision
At 5 docs	0.3120
At 10 docs	0.2720
At 15 docs	0.2547
At 20 docs	0.2400
At 30 docs	0.2200
At 100 docs	0.1434
At 200 docs	0.1072
At 500 docs	0.0647
At 1000 docs	0.0433
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1974



Recall-Precision Curve



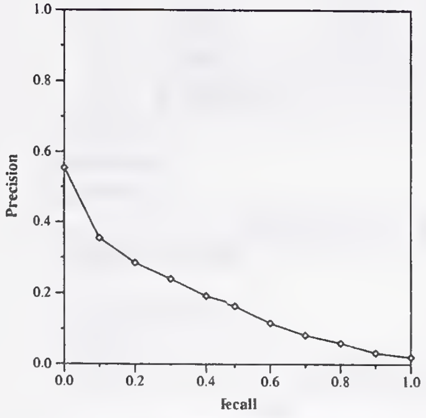
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UniNEtdL
Run Descriptioı	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

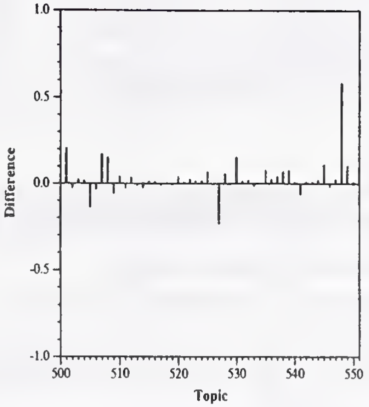
Total number of documents over all topics	
Retrieved:	49528
Relevant:	3363
Rel-ret:	2159

Recall Level Averages	
Recal	Precision
0.00	0.5545
0.10	0.3548
0.20	0.2852
0.30	0.2403
0.40	0.1923
0.50	0.1639
0.60	0.1169
0.70	0.0830
0.80	0.0611
0.90	0.0331
1.00	0.0205
Mean averge precision	
non-interplated	0.1715

Document Level Averages	
	Precision
At 5 docs	0.3080
At 10 docs	0.2660
At 15 docs	0.2547
At 20 docs	0.2430
At 30 docs	0.2260
At 100 docs	0.1458
At 200 docs	0.1076
At 500 docs	0.0649
At 1000 docs	0.0432
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1990



Recall-Precision Curve



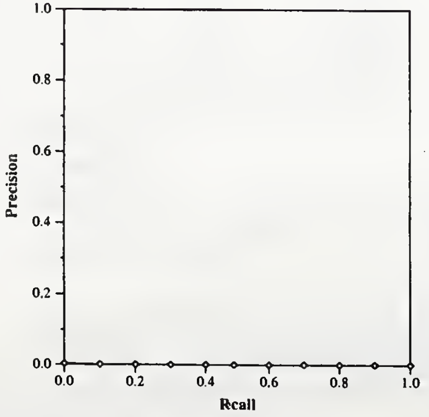
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	irtLnua
Run Description	Automatic, title only; docstruct-used, urltext-notused, links-notused
Number of Topi:s:	50

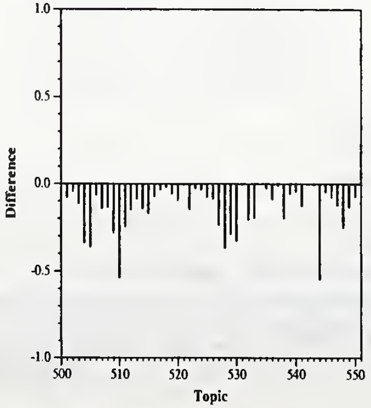
Total number of documents over all topics	
Retrieved:	31680
Relevant:	3363
Rel-ret:	28

Recall level Averages	
Recal	Precision
0.00	0.0024
0.10	0.0007
0.20	0.0006
0.30	0.0004
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 aver;ge precision	
non-interp.lated	0.0002

Document Level Averages	
	Precision
At 5 docs	0.0000
At 10 docs	0.0000
At 15 docs	0.0000
At 20 docs	0.0010
At 30 docs	0.0007
At 100 docs	0.0006
At 200 docs	0.0011
At 500 docs	0.0008
At 1000 docs	0.0006
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0004



Recall-Prcision Curve



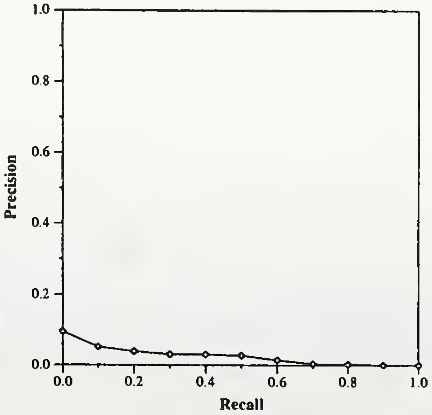
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	irtLnut
Run Description	Automatic, title only; docstruct-used, urltext-notused, links-notused
Number of Topics:	50

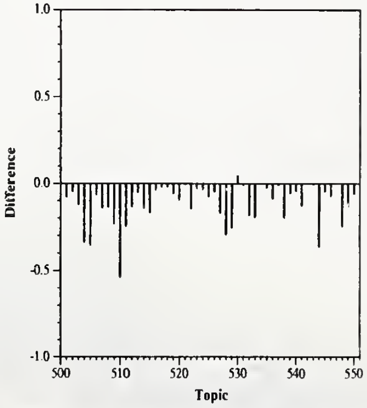
Total number of documents over all topics	
Retrieved:	46432
Relevant:	3363
Rel-ret:	838

Recall Level Averages	
Recall	Precision
0.00	0.0960
0.10	0.0522
0.20	0.0395
0.30	0.0310
0.40	0.0304
0.50	0.0276
0.60	0.0147
0.70	0.0039
0.80	0.0027
0.90	0.0001
1.00	0.0001
Mean average precision	
non-interpolated	0.0221

Document Level Averages	
	Precision
At 5 docs	0.0400
At 10 docs	0.0440
At 15 docs	0.0453
At 20 docs	0.0470
At 30 docs	0.0467
At 100 docs	0.0336
At 200 docs	0.0273
At 500 docs	0.0180
At 1000 docs	0.0168
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0321



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Web track, ad hœ task results — University of North Carolina at Chapel Hill-Yang

Summary Statistics	
Run ID:	uncfslm
Run Description	Automatic, adhoc, title+desc; docstruct-notused, urltext-notused, links-used
Number of Topis:	50

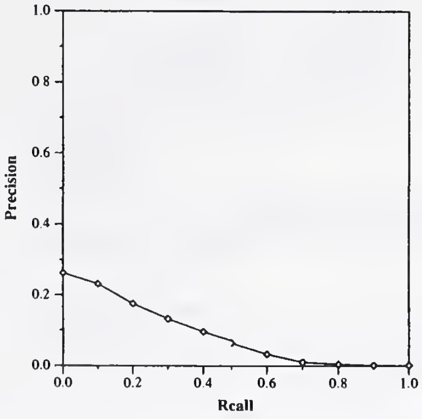
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	1788

Recall level Averages	
Recall	Precision
0.00	0.2622
0.10	0.2318
0.20	0.1757
0.30	0.1325
0.40	0.0961
0.50	0.0644
0.60	0.0334
0.70	0.0108
0.80	0.0048
0.90	0.0018
1.00	0.0018

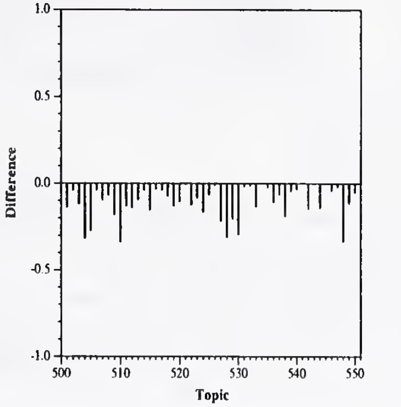
Mean averge precision	
non-interpdated	0.0781

Document Level Averages	
	Precision
At 5 docs	0.0520
At 10 docs	0.0700
At 15 docs	0.1373
At 20 docs	0.1660
At 30 docs	0.1667
At 100 docs	0.1346
At 200 docs	0.0993
At 500 docs	0.0564
At 1000 docs	0.0358

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1420



Recall-Preision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	uncvsmm
Run Description	Automatic, adhoc, title+desc; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

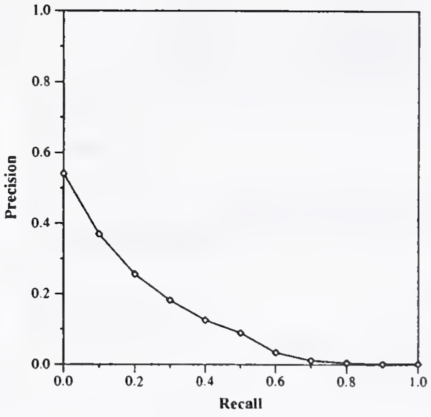
Total number of documents over all topics	
Retrieved:	50000
Relevant:	3363
Rel-ret:	1797

Recall Level Averages	
Recall	Precision
0.00	0.5403
0.10	0.3694
0.20	0.2561
0.30	0.1818
0.40	0.1255
0.50	0.0894
0.60	0.0346
0.70	0.0118
0.80	0.0051
0.90	0.0020
1.00	0.0020

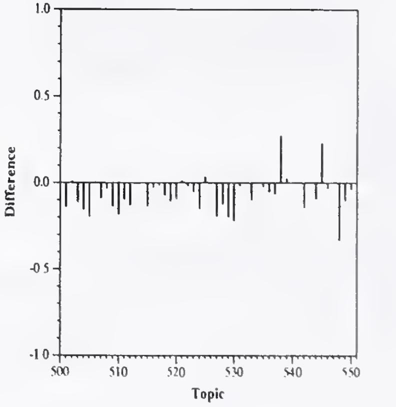
Mean average precision	
non-interpolated	0.1269

Document Level Averages	
	Precision
At 5 docs	0.3200
At 10 docs	0.3000
At 15 docs	0.2573
At 20 docs	0.2390
At 30 docs	0.2160
At 100 docs	0.1440
At 200 docs	0.1012
At 500 docs	0.0566
At 1000 docs	0.0359

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1761



Recall-Precision Curve



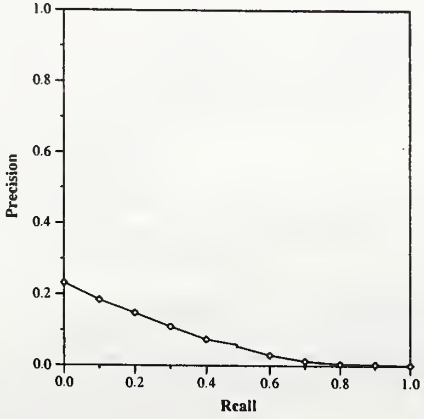
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	uncfsls
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-used
Number of Topis:	50

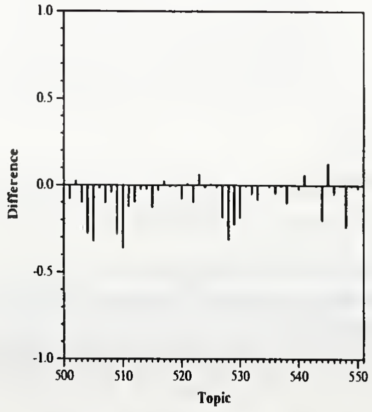
Total number of documents over all topics	
Retrieved:	48402
Relevant:	3363
Rel-ret:	1666

Recall level Averages	
Recall	Precision
0.00	0.2324
0.10	0.1852
0.20	0.1490
0.30	0.1106
0.40	0.0747
0.50	0.0568
0.60	0.0298
0.70	0.0128
0.80	0.0045
0.90	0.0032
1.00	0.0010
Mean average precision	
non-interpdated	0.0663

Document Level Averages	
	Precision
At 5 docs	0.0600
At 10 docs	0.0760
At 15 docs	0.1200
At 20 docs	0.1360
At 30 docs	0.1407
At 100 docs	0.1162
At 200 docs	0.0869
At 500 docs	0.0512
At 1000 docs	0.0333
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1206



Recall-Preision Curve



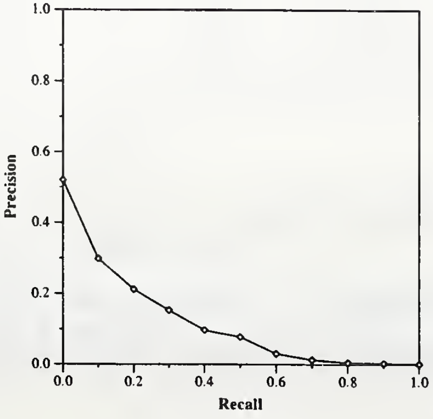
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	uncvsms
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

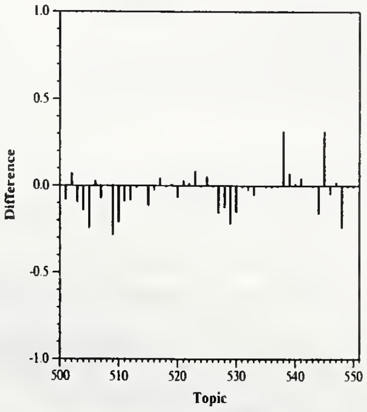
Total number of documents over all topics	
Retrieved:	48363
Relevant:	3363
Rel-ret:	1668

Recall Level Averages	
Recall	Precision
0.00	0.5201
0.10	0.2983
0.20	0.2122
0.30	0.1533
0.40	0.0977
0.50	0.0782
0.60	0.0305
0.70	0.0131
0.80	0.0047
0.90	0.0033
1.00	0.0011
Mean average precision	
non-interpolated	0.1069

Document Level Averages	
	Precision
At 5 docs	0.2480
At 10 docs	0.2400
At 15 docs	0.2000
At 20 docs	0.1920
At 30 docs	0.1720
At 100 docs	0.1226
At 200 docs	0.0884
At 500 docs	0.0514
At 1000 docs	0.0334
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1578



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	PDWTAHDR
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topic:	50

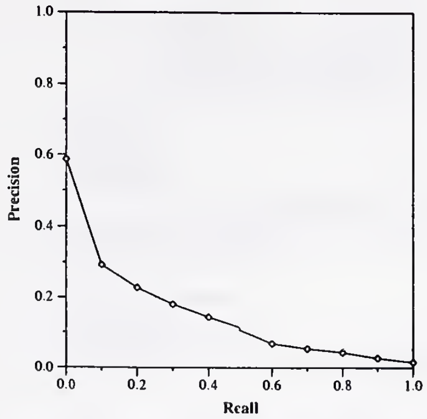
ʾotal number of documents over all topics	
Retrieved:	48714
Relevant:	3363
Rel-ret:	1888

Recall Lɜvel Averages	
Recall	Precision
0.00	0.5868
0.10	0.2927
0.20	0.2280
0.30	0.1806
0.40	0.1439
0.50	0.1103
0.60	0.0678
0.70	0.0532
0.80	0.0429
0.90	0.0273
1.00	0.0159

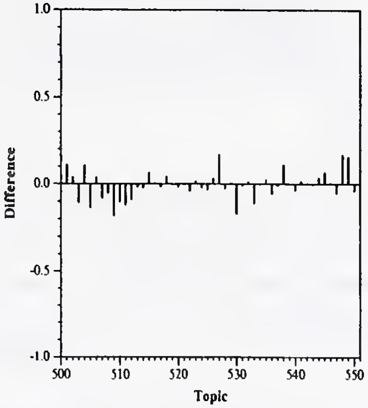
Mean averaɜe precision	
non-interpcated	0.1332

Document Level Averages	
	Precision
At 5 docs	0.3120
At 10 docs	0.2680
At 15 docs	0.2373
At 20 docs	0.2220
At 30 docs	0.1953
At 100 docs	0.1274
At 200 docs	0.0928
At 500 docs	0.0607
At 1000 docs	0.0378

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1501



Recall-Preision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	PDWTAHPR
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

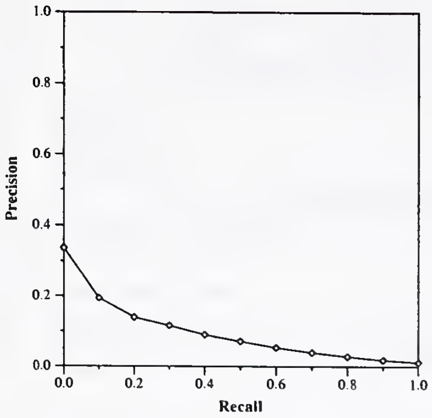
Total number of documents over all topics	
Retrieved:	48710
Relevant:	3363
Rel-ret:	1833

Recall Level Averages	
Recall	Precision
0.00	0.3363
0.10	0.1940
0.20	0.1393
0.30	0.1156
0.40	0.0896
0.50	0.0707
0.60	0.0531
0.70	0.0395
0.80	0.0285
0.90	0.0189
1.00	0.0121

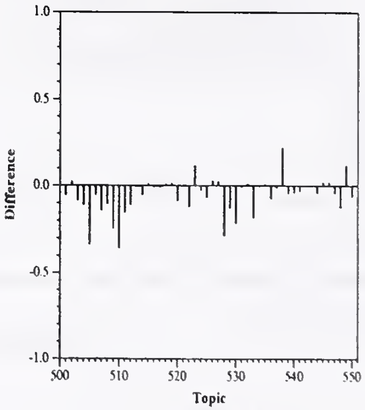
Mean average precision	
non-interpolated	0.0842

Document Level Averages	
	Precision
At 5 docs	0.1680
At 10 docs	0.1600
At 15 docs	0.1453
At 20 docs	0.1460
At 30 docs	0.1393
At 100 docs	0.1014
At 200 docs	0.0849
At 500 docs	0.0588
At 1000 docs	0.0367

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1048



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	PDWTAHTL
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-used
Number of Topis:	50

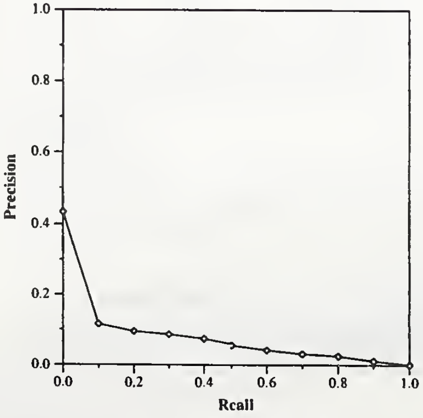
Total number of documents over all topics	
Retrieved:	48714
Relevant:	3363
Rel-ret:	1888

Recall Level Averages	
Recall	Precision
0.00	0.4333
0.10	0.1153
0.20	0.0946
0.30	0.0857
0.40	0.0733
0.50	0.0546
0.60	0.0416
0.70	0.0308
0.80	0.0251
0.90	0.0117
1.00	0.0012

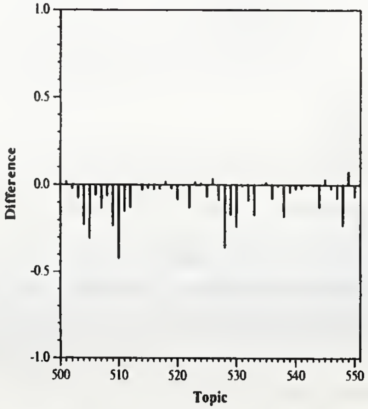
Mean averge precision	
non-interpdated	0.0601

Document Level Averages	
	Precision
At 5 docs	0.1400
At 10 docs	0.1140
At 15 docs	0.0973
At 20 docs	0.0900
At 30 docs	0.0793
At 100 docs	0.0682
At 200 docs	0.0797
At 500 docs	0.0590
At 1000 docs	0.0378

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0762



Recall-Preision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	PDWTAHWL
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-used
Number of Topics:	50

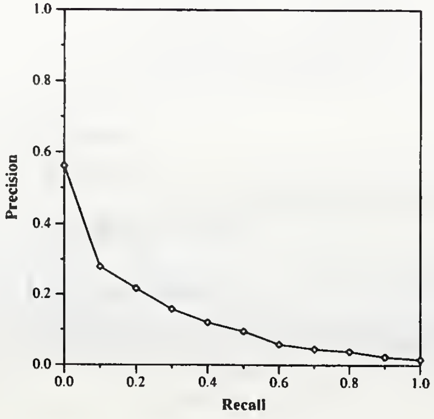
Total number of documents over all topics	
Retrieved:	48714
Relevant:	3363
Rel-ret:	1888

Recall Level Averages	
Recall	Precision
0.00	0.5614
0.10	0.2791
0.20	0.2174
0.30	0.1582
0.40	0.1205
0.50	0.0951
0.60	0.0581
0.70	0.0446
0.80	0.0377
0.90	0.0226
1.00	0.0152

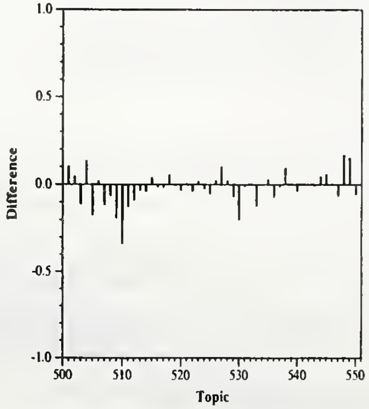
Mean average precision	
non-interpolated	0.1209

Document Level Averages	
	Precision
At 5 docs	0.3080
At 10 docs	0.2540
At 15 docs	0.2227
At 20 docs	0.2070
At 30 docs	0.1860
At 100 docs	0.1156
At 200 docs	0.0869
At 500 docs	0.0570
At 1000 docs	0.0378

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1396



Recall-Precision Curve



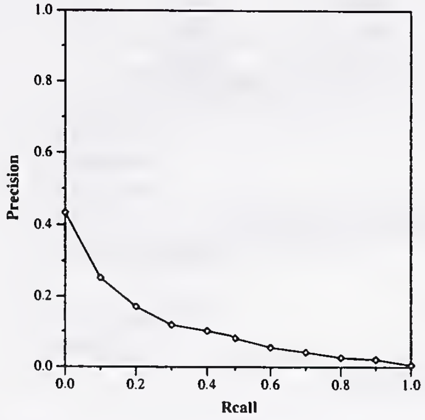
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	uwmtaw0
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topi:s:	50

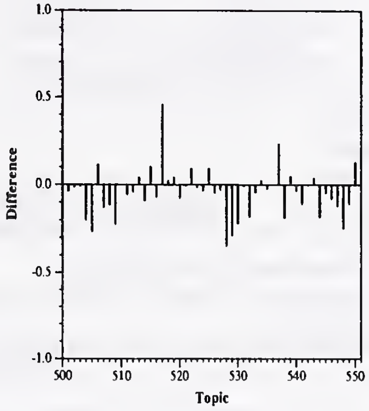
Total number of documents over all topics	
Retrieved:	48311
Relevant:	3363
Rel-ret:	1361

Recall level Averages	
Recal	Precision
0.00	0.4330
0.10	0.2526
0.20	0.1703
0.30	0.1180
0.40	0.1011
0.50	0.0809
0.60	0.0545
0.70	0.0412
0.80	0.0262
0.90	0.0208
1.00	0.0046
Mean aver:ge precision	
non-interplated	0.0951

Document Level Averages	
	Precision
At 5 docs	0.2000
At 10 docs	0.1860
At 15 docs	0.1787
At 20 docs	0.1700
At 30 docs	0.1620
At 100 docs	0.1130
At 200 docs	0.0724
At 500 docs	0.0422
At 1000 docs	0.0272
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1267



Recall-Preision Curve



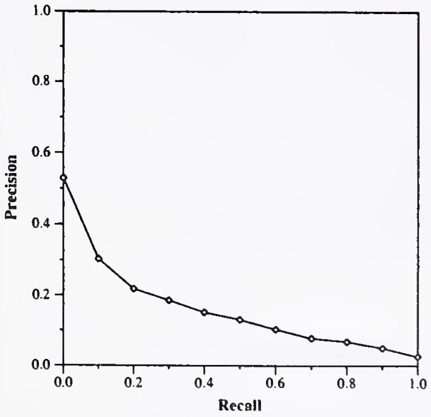
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	uwmtaw1
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topics:	50

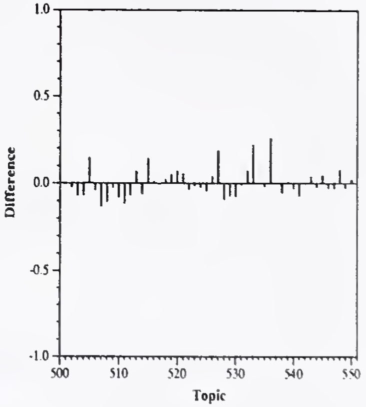
Total number of documents over all topics	
Retrieved:	48578
Relevant:	3363
Rel-ret:	1949

Recall Level Averages	
Recall	Precision
0.00	0.5286
0.10	0.3032
0.20	0.2182
0.30	0.1854
0.40	0.1508
0.50	0.1300
0.60	0.1025
0.70	0.0772
0.80	0.0677
0.90	0.0496
1.00	0.0255
Mean average precision	
non-interpolated	0.1416

Document Level Averages	
	Precision
At 5 docs	0.2760
At 10 docs	0.2580
At 15 docs	0.2293
At 20 docs	0.2110
At 30 docs	0.1833
At 100 docs	0.1284
At 200 docs	0.1038
At 500 docs	0.0635
At 1000 docs	0.0390
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1645



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	uwmtaw2
Run Description	Automatic, title only; docstruct-notused, urltext-notused, links-notused
Number of Topic:	50

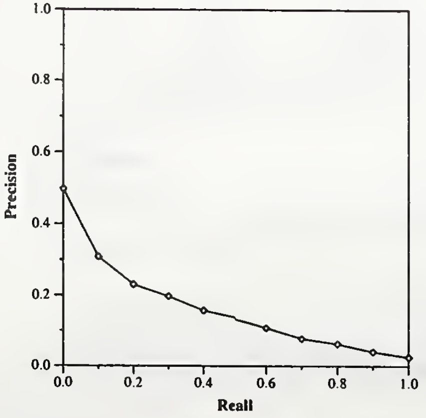
⁊otal number of documents over all topics	
Retrieved:	47911
Relevant:	3363
Rel-ret:	1995

Recall Lvel Averages	
Recall	Precision
0.00	0.4967
0.10	0.3087
0.20	0.2310
0.30	0.1976
0.40	0.1572
0.50	0.1354
0.60	0.1079
0.70	0.0776
0.80	0.0628
0.90	0.0407
1.00	0.0255

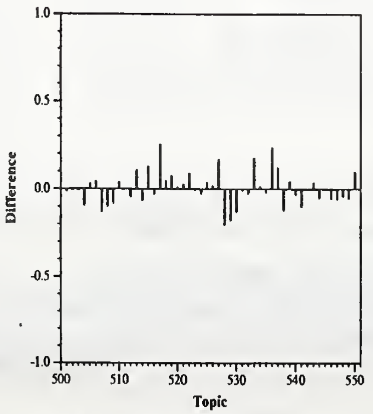
Mean averaʒe precision	
non-interpoated	0.1420

Document Level Averages	
	Precision
At 5 docs	0.2320
At 10 docs	0.2220
At 15 docs	0.2187
At 20 docs	0.2130
At 30 docs	0.1967
At 100 docs	0.1388
At 200 docs	0.1068
At 500 docs	0.0645
At 1000 docs	0.0399

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1726



Recall-Precsion Curve



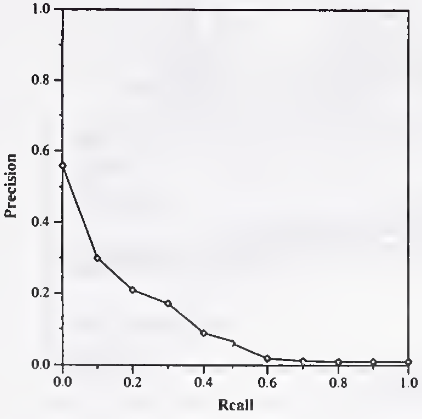
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	yeahdb01
Run Description	Automatic, adhoc, title+desc; docstruct-used, urltext-notused, links-notused
Number of Topis:	50

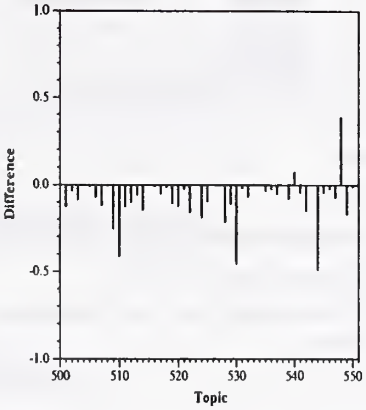
Total number of documents over all topics	
Retrieved:	47259
Relevant:	3363
Rel-ret:	1176

Recall Level Averages	
Recall	Precision
0.00	0.5585
0.10	0.2991
0.20	0.2097
0.30	0.1722
0.40	0.0898
0.50	0.0615
0.60	0.0195
0.70	0.0124
0.80	0.0100
0.90	0.0100
1.00	0.0100
Mean averge precision	
non-interpdated	0.1094

Document Level Averages	
	Precision
At 5 docs	0.2960
At 10 docs	0.2600
At 15 docs	0.2307
At 20 docs	0.2020
At 30 docs	0.1733
At 100 docs	0.1152
At 200 docs	0.0716
At 500 docs	0.0399
At 1000 docs	0.0235
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1564



Recall-Preision Curve



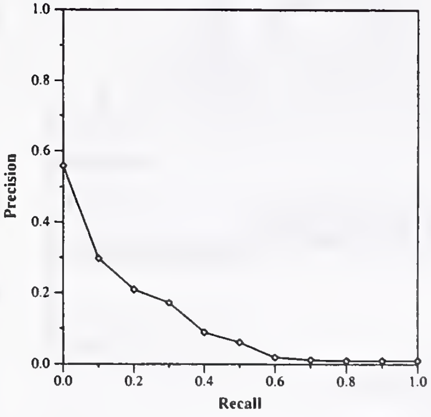
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	yeahtd01
Run Description	Automatic, adhoc, title+desc; docstruct-used, urltext-notused, links-used
Number of Topics:	50

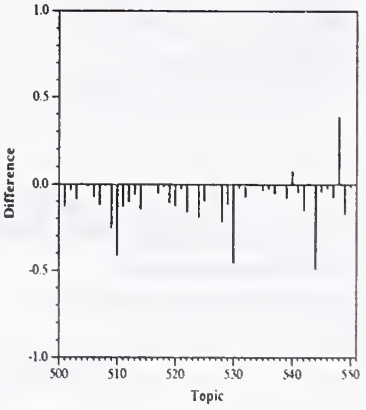
Total number of documents over all topics	
Retrieved:	47259
Relevant:	3363
Rel-ret:	1176

Recall Level Averages	
Recall	Precision
0.00	0.5592
0.10	0.2974
0.20	0.2092
0.30	0.1720
0.40	0.0897
0.50	0.0614
0.60	0.0195
0.70	0.0124
0.80	0.0100
0.90	0.0100
1.00	0.0100
Mean average precision	
non-interpolated	0.1092

Document Level Averages	
	Precision
At 5 docs	0.2960
At 10 docs	0.2620
At 15 docs	0.2320
At 20 docs	0.2020
At 30 docs	0.1747
At 100 docs	0.1148
At 200 docs	0.0716
At 500 docs	0.0399
At 1000 docs	0.0235
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1584



Recall-Precision Curve



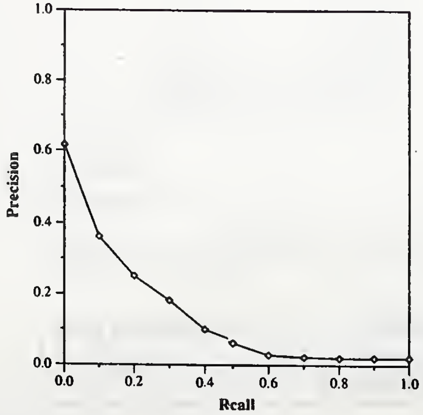
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	yeaht01
Run Description	Automatic, title only; docstruct-used, urltext-notused, links-used
Number of Topics:	50

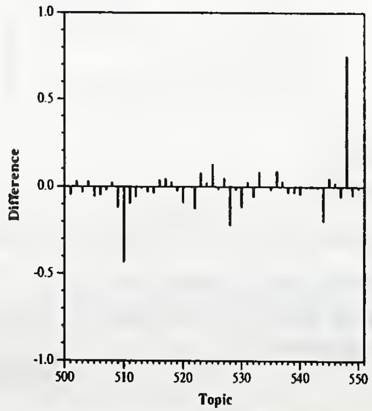
Total number of documents over all topics	
Retrieved:	44922
Relevant:	3363
Rel-ret:	1337

Recall Level Averages	
Recall	Precision
0.00	0.6152
0.10	0.3619
0.20	0.2511
0.30	0.1820
0.40	0.0998
0.50	0.0616
0.60	0.0286
0.70	0.0225
0.80	0.0200
0.90	0.0200
1.00	0.0200
Mean average precision	
non-interpolated	0.1286

Document Level Averages	
	Precision
At 5 docs	0.3880
At 10 docs	0.3240
At 15 docs	0.2800
At 20 docs	0.2520
At 30 docs	0.2180
At 100 docs	0.1282
At 200 docs	0.0830
At 500 docs	0.0473
At 1000 docs	0.0267
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1796



Recall-Precision Curve



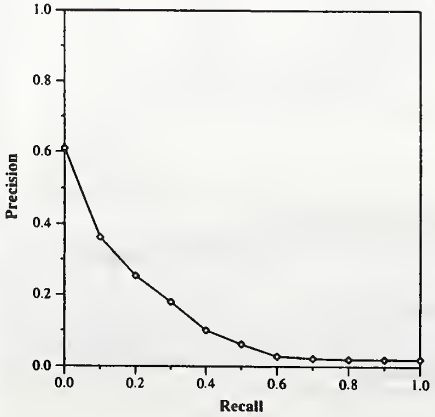
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	yeahtb01
Run Description	Automatic, title only; docstruct-used, urltext-notused, links-notused
Number of Topics:	50

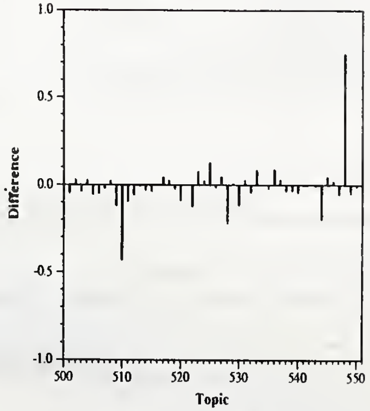
Total number of documents over all topics	
Retrieved:	44918
Relevant:	3363
Rel-ret:	1338

Recall Level Averages	
Recall	Precision
0.00	0.6086
0.10	0.3618
0.20	0.2534
0.30	0.1796
0.40	0.1002
0.50	0.0618
0.60	0.0286
0.70	0.0225
0.80	0.0200
0.90	0.0200
1.00	0.0200
Mean average precision	
non-interpolated	0.1287

Document Level Averages	
	Precision
At 5 docs	0.3840
At 10 docs	0.3240
At 15 docs	0.2800
At 20 docs	0.2520
At 30 docs	0.2180
At 100 docs	0.1284
At 200 docs	0.0830
At 500 docs	0.0474
At 1000 docs	0.0268
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1811



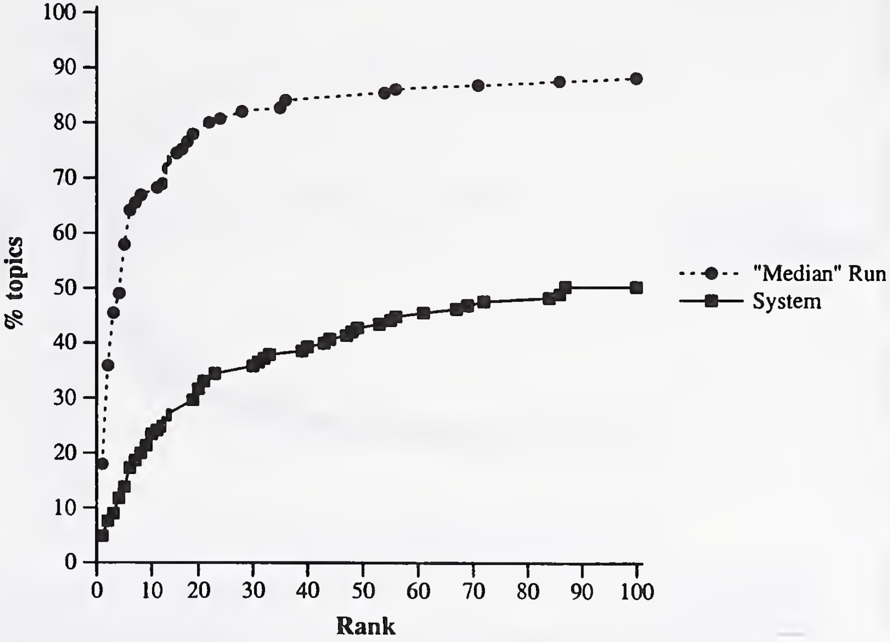
Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics

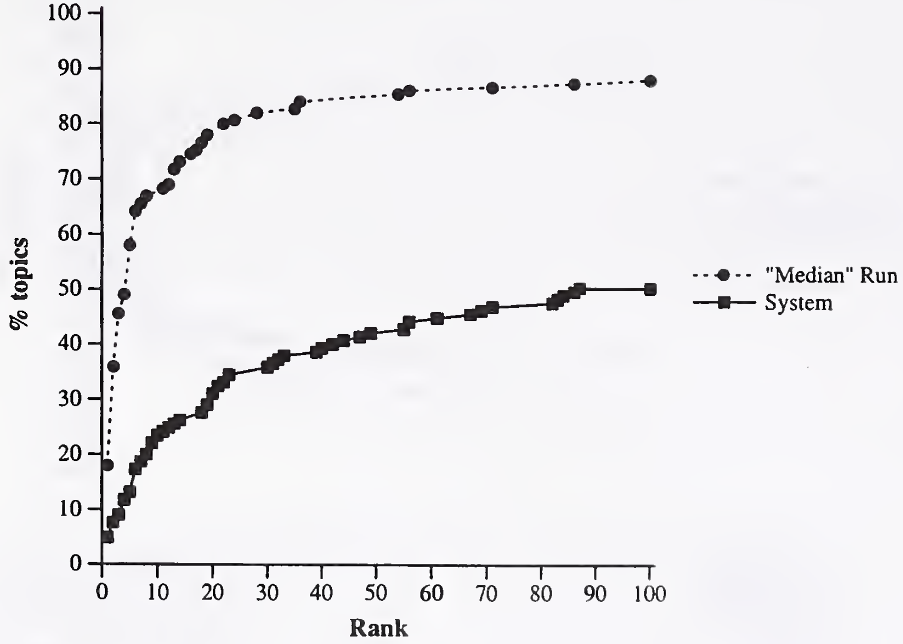
Run ID	ajouai0102
Run Description	docstruct-notused, urltext-notused, links-notused
Num topics	145
Mean reciprocl rank	0.101
Num found atrank 1	7 (4.8%)
Num found intop 10	34 (23.4%)
Num not founl in top 100	72 (49.7%)



Cumulative % of topics that retrieve homepage by given rank

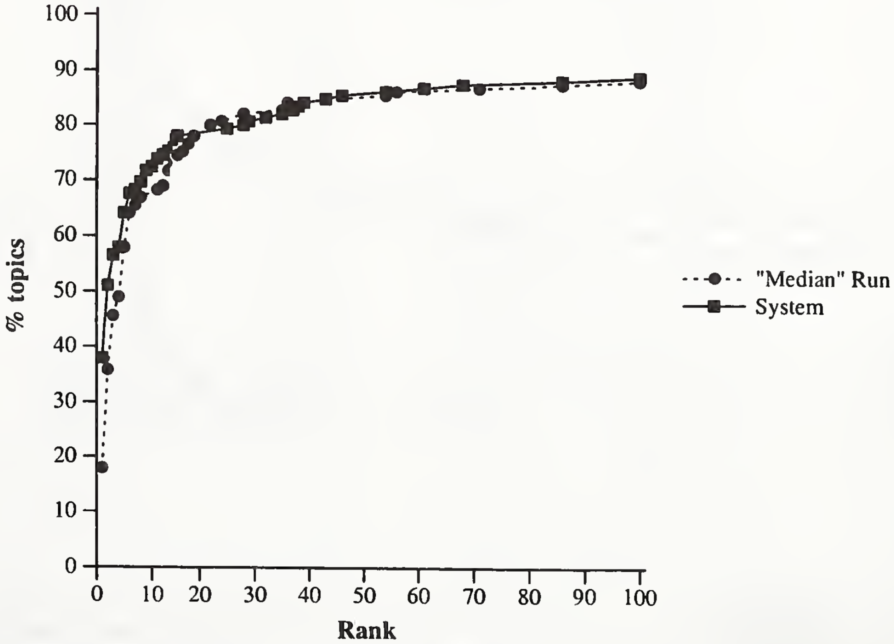
Summary Statistics

Run ID	ajouai0104
Run Description	docstruct-notused, urltext-notused, links-used
Num topics	145
Mean reciprocal rank	0.100
Num found at rank 1	7 (4.8%)
Num found in top 10	34 (23.4%)
Num not found in top 100	72 (49.7%)



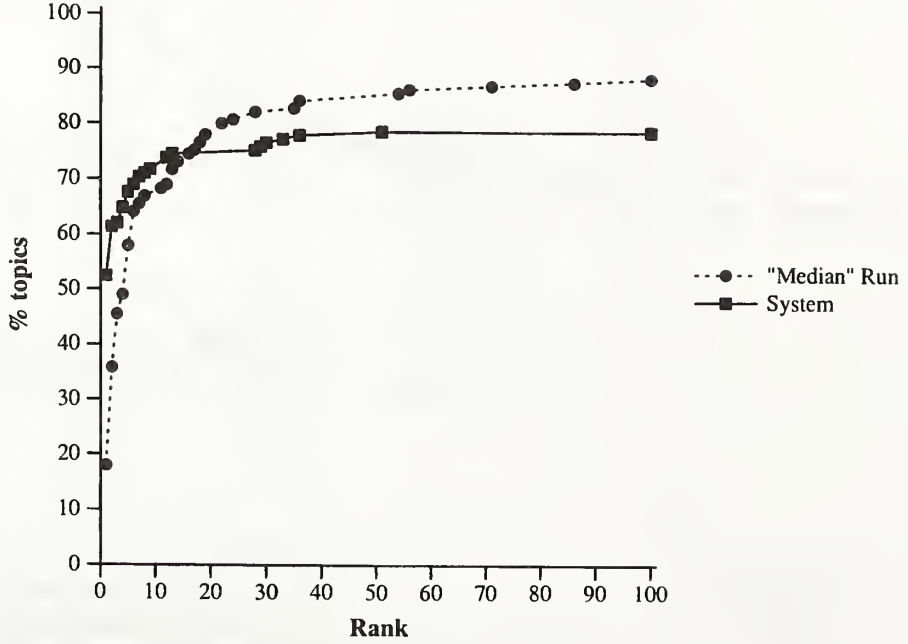
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	csiro0awh1
Run Descriptin	docstruct-used, urltext-used, links-used
Num topics	145
Mean reciprocl rank	0.498
Num found atrank 1	55 (37.9%)
Num found intop 10	105 (72.4%)
Num not founl in top 100	16 (11.0%)



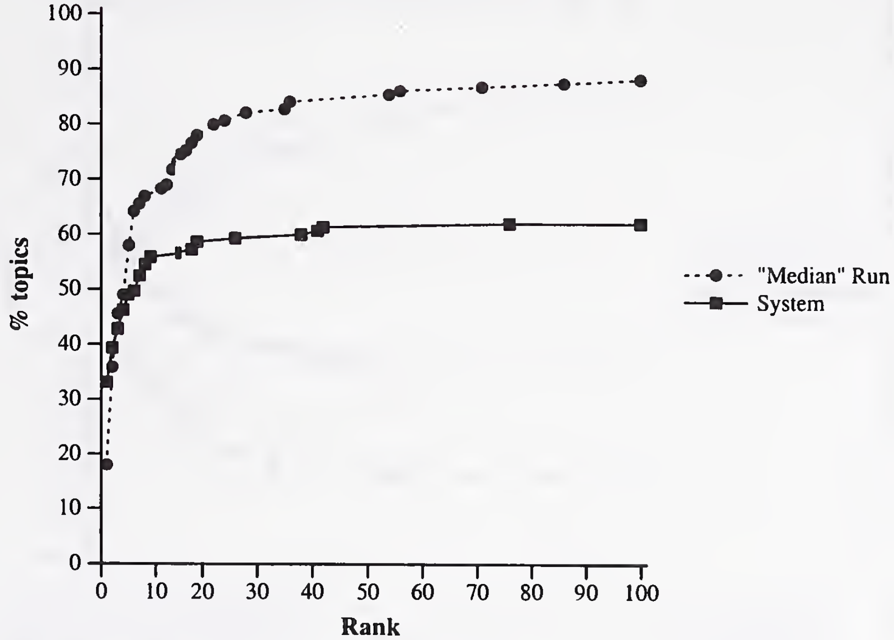
Cumulative %of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	csiro0awh2
Run Description	docstruct-notused, urltext-notused, links-used
Num topics	145
Mean reciprocal rank	0.593
Num found at rank 1	76 (52.4%)
Num found in top 10	104 (71.7%)
Num not found in top 100	31 (21.4%)



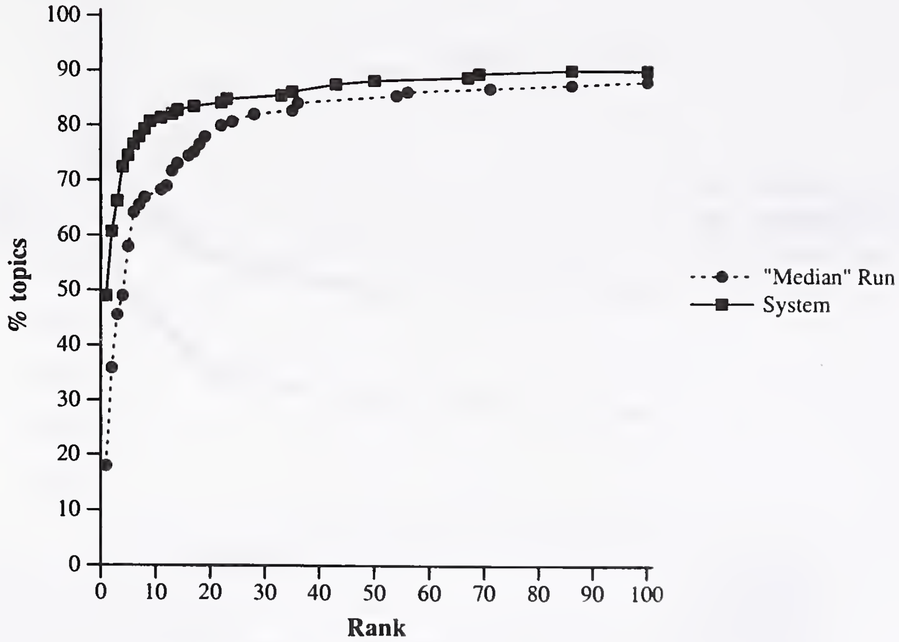
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	flabxe75a
Run Descriptim	docstruct-used, urltext-used, links-used
Num topics	145
Mean reciprocl rank	0.399
Num found at rank 1	48 (33.1%)
Num found in :op 10	81 (55.9%)
Num not founc in top 100	55 (37.9%)



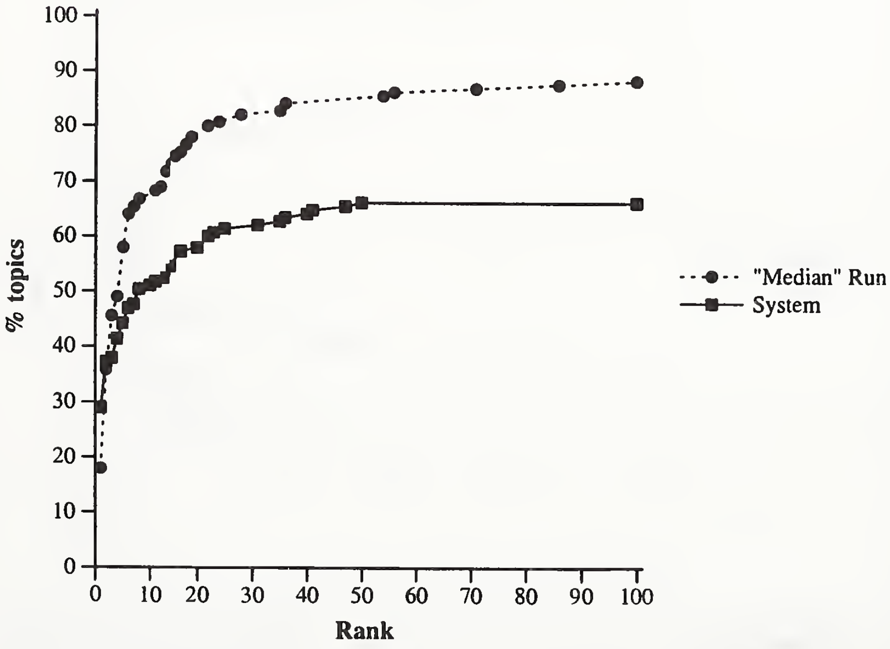
Cumulative %of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	flabxeall
Run Description	docstruct-notused, urltext-notused, links-used
Num topics	145
Mean reciprocal rank	0.599
Num found at rank 1	71 (49.0%)
Num found in top 10	117 (80.7%)
Num not found in top 100	14 (9.7%)



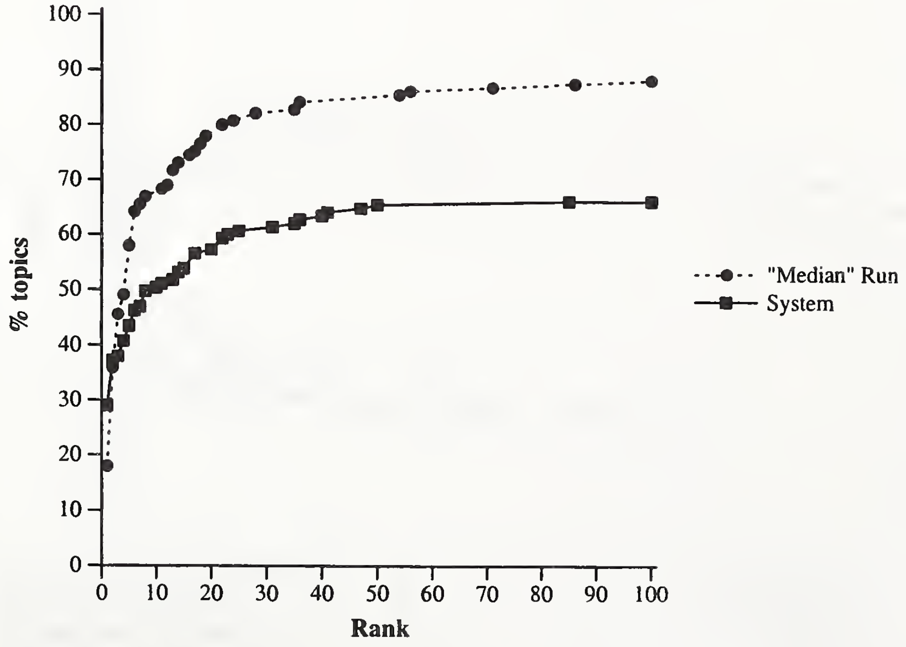
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	flabxemerge
Run Descriptim	docstruct-used, urltext-used, links-used
Num topics	145
Mean reciprocal rank	0.365
Num found at :ank 1	42 (29.0%)
Num found in ,op 10	74 (51.0%)
Num not founc in top 100	49 (33.8%)



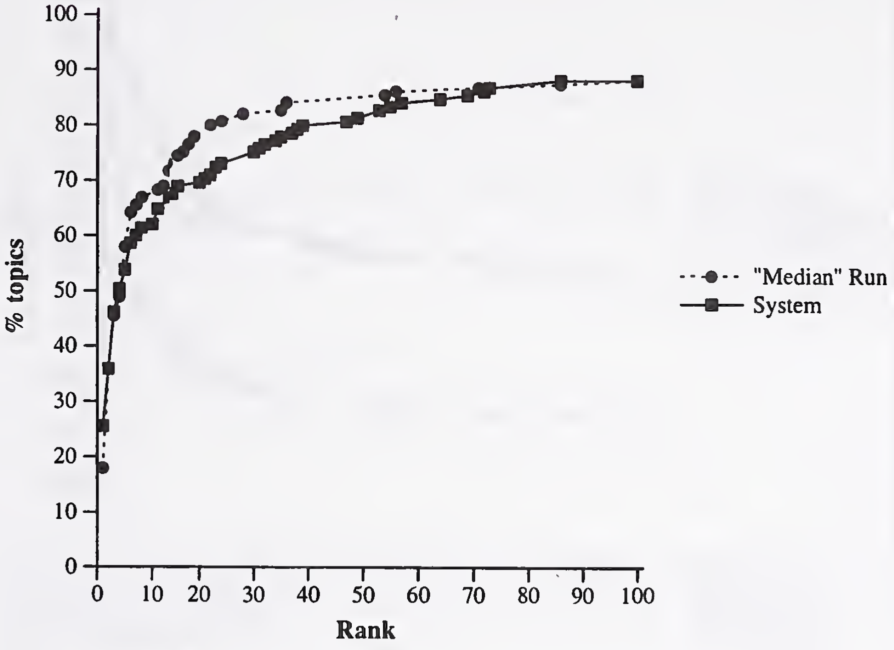
Cumulative %of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	flabxet256
Run Description	docstruct-used, urltext-notused, links-used
Num topics	145
Mean reciprocal rank	0.363
Num found at rank 1	42 (29.0%)
Num found in top 10	73 (50.3%)
Num not found in top 100	49 (33.8%)



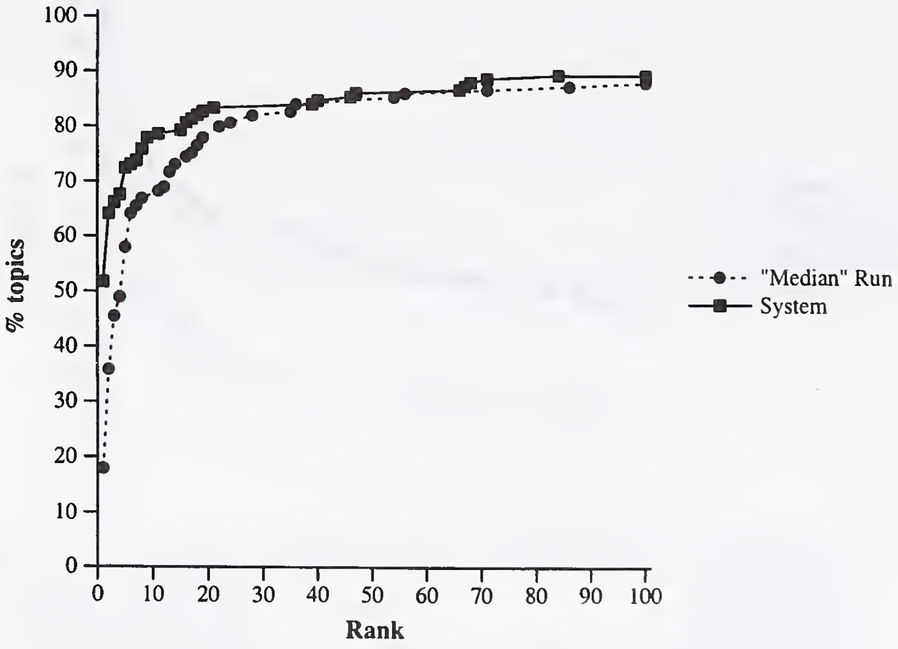
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	IBMHOMENR
Run Descriptin	docstruct-used, urltext-notused, links-notused
Num topics	145
Mean reciprocl rank	0.382
Num found atrank 1	37 (25.5%)
Num found intop 10	90 (62.1%)
Num not founl in top 100	17 (11.7%)



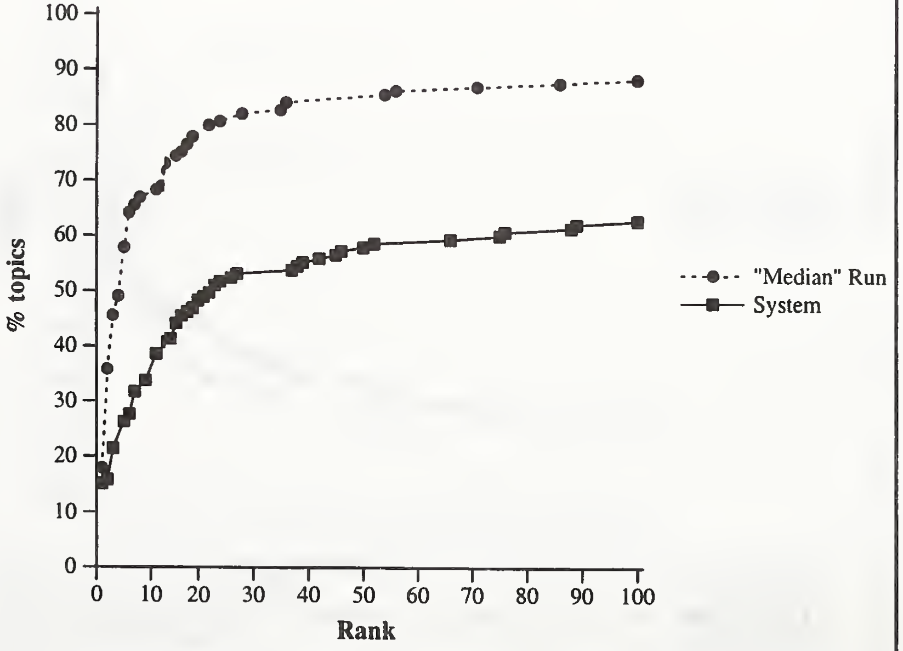
Cumulative %of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	IBMHOMER
Run Description	docstruct-used, urltext-notused, links-used
Num topics	145
Mean reciprocal rank	0.611
Num found at rank 1	75 (51.7%)
Num found in top 10	113 (77.9%)
Num not found in top 100	15 (10.3%)



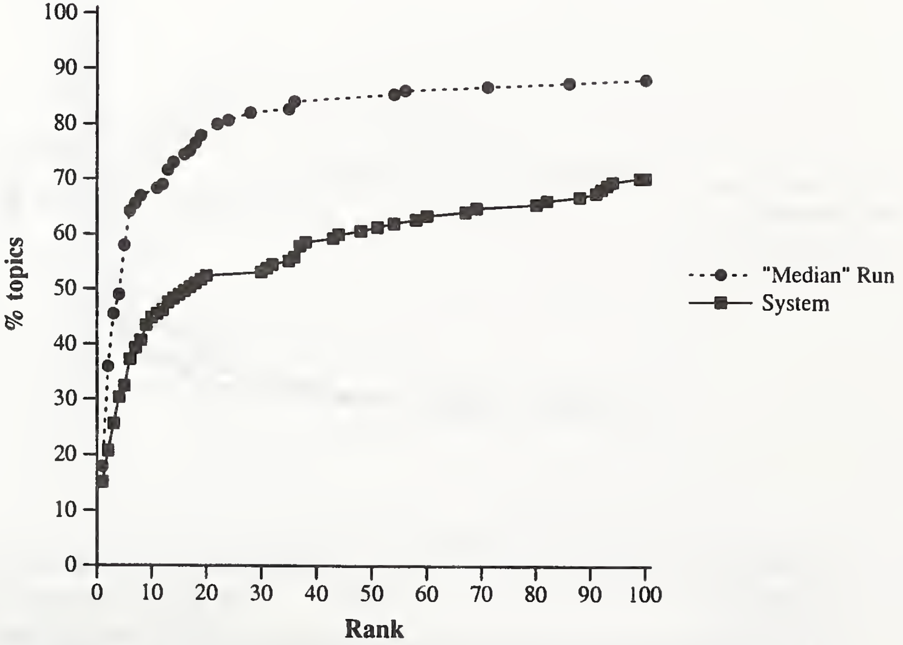
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	ichp1
Run Descriptibn	docstruct-notused, urltext-notused, links-used
Num topics	145
Mean reciprocal rank	0.208
Num found atrank 1	22 (15.2%)
Num found intop 10	49 (33.8%)
Num not founl in top 100	54 (37.2%)



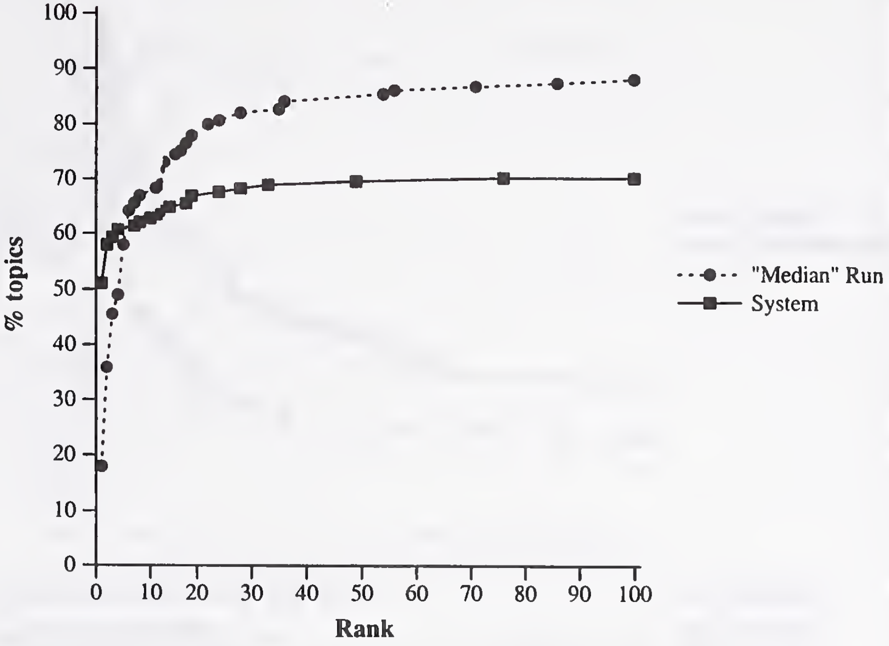
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	ichp2
Run Description	docstruct-notused, urltext-notused, links-notused
Num topics	145
Mean reciprocal rank	0.237
Num found at rank 1	22 (15.2%)
Num found in top 10	65 (44.8%)
Num not found in top 100	43 (29.7%)



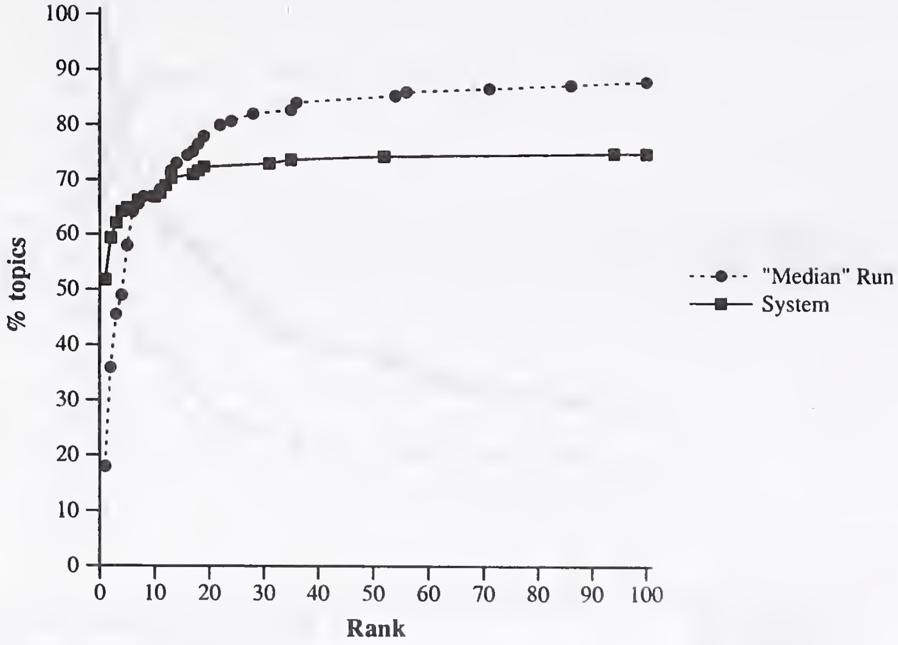
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	iit01st
Run Description	docstruct-used, urltext-used, links-notused
Num topics	145
Mean reciprocal rank	0.559
Num found at rank 1	74 (51.0%)
Num found in top 10	91 (62.8%)
Num not found in top 100	43 (29.7%)



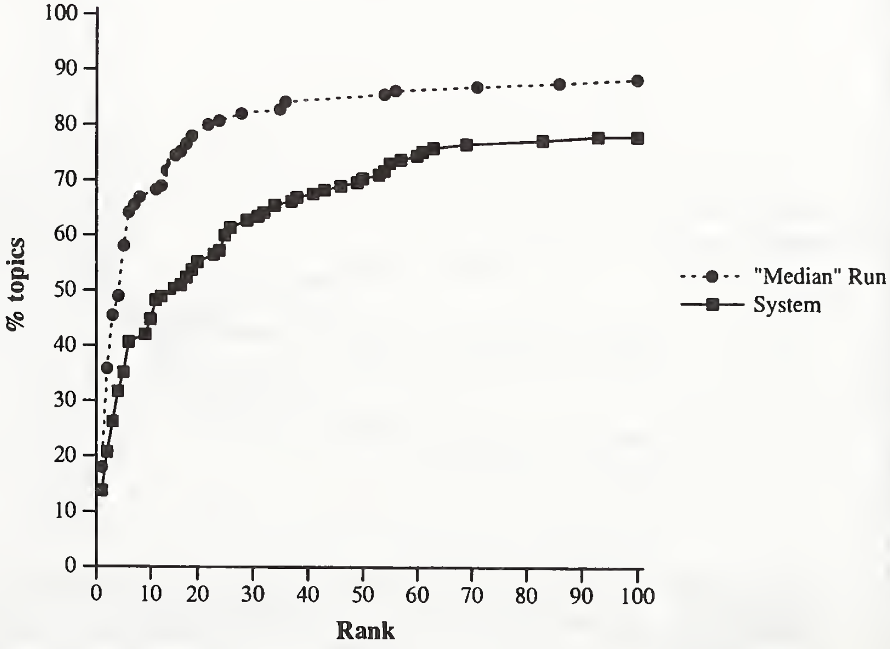
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	iit01stb
Run Description	docstruct-used, urltext-used, links-used
Num topics	145
Mean reciprocal rank	0.578
Num found at rank 1	75 (51.7%)
Num found in top 10	97 (66.9%)
Num not found in top 100	36 (24.8%)



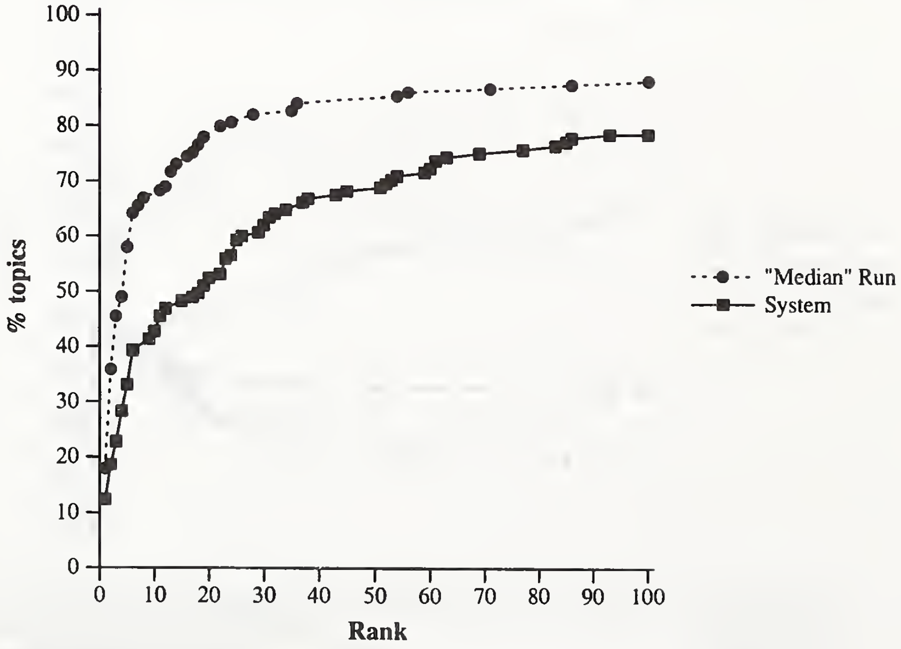
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	apl10ha
Run Description	docstruct-notused, urltext-notused, links-notused
Num topics	145
Mean reciprocal rank	0.238
Num found atrank 1	20 (13.8%)
Num found intop 10	65 (44.8%)
Num not found in top 100	32 (22.1%)



Cumulative %of topics that retrieve homepage by given rank

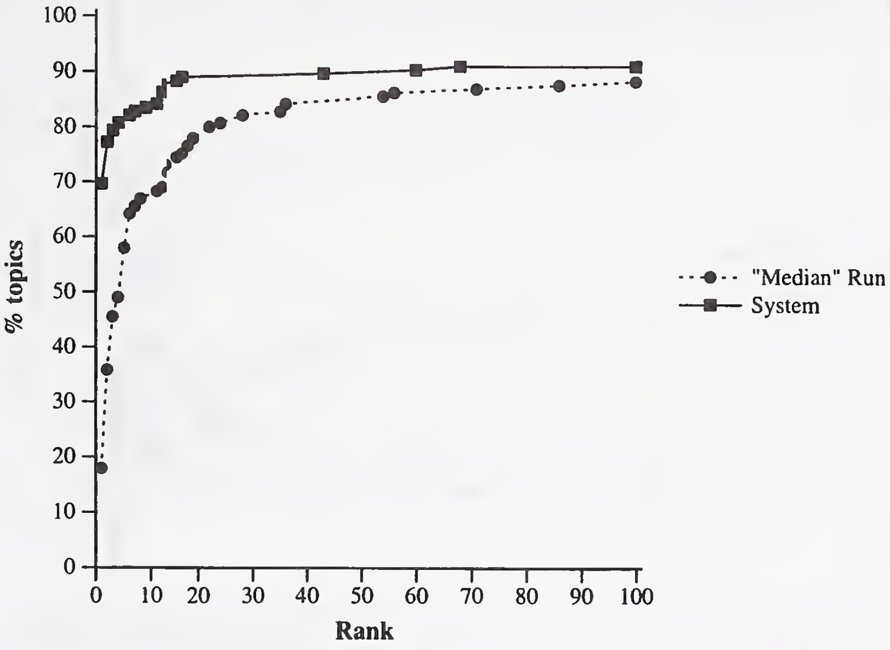
Summary Statistics	
Run ID	apl10hb
Run Description	docstruct-notused, urltext-notused, links-notused
Num topics	145
Mean reciprocal rank	0.220
Num found at rank 1	18 (12.4%)
Num found in top 10	62 (42.8%)
Num not found in top 100	31 (21.4%)



Cumulative % of topics that retrieve homepage by given rank

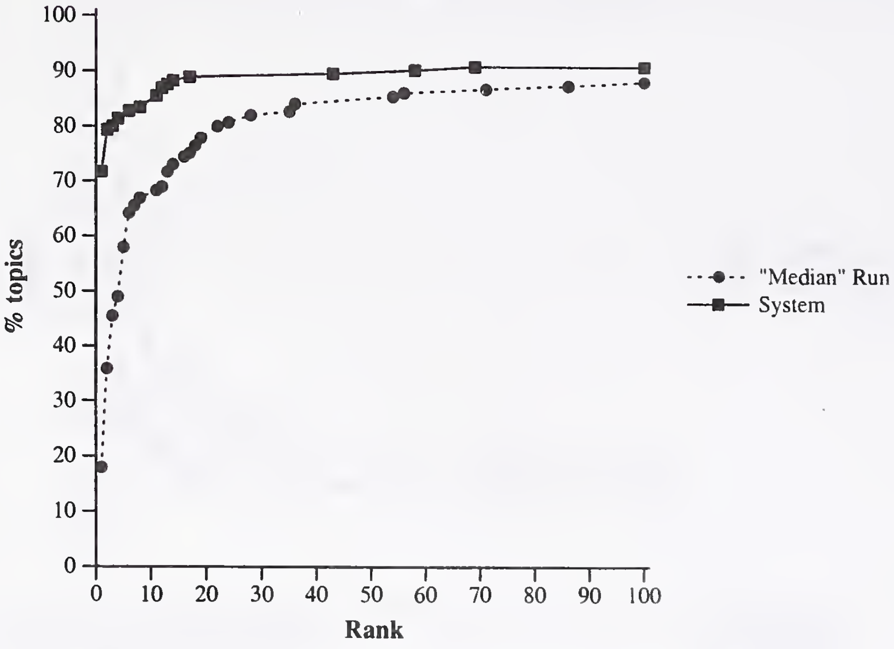
Web track, entry page finding task results — Justsystem Corporation

Summary Statistics	
Run ID	jscbtawep1
Run Descriptiın	docstruct-used, urltext-used, links-used
Num topics	145
Mean reciprocal rank	0.754
Num found atrank 1	101 (69.7%)
Num found intop 10	121 (83.4%)
Num not founl in top 100	13 (9.0%)



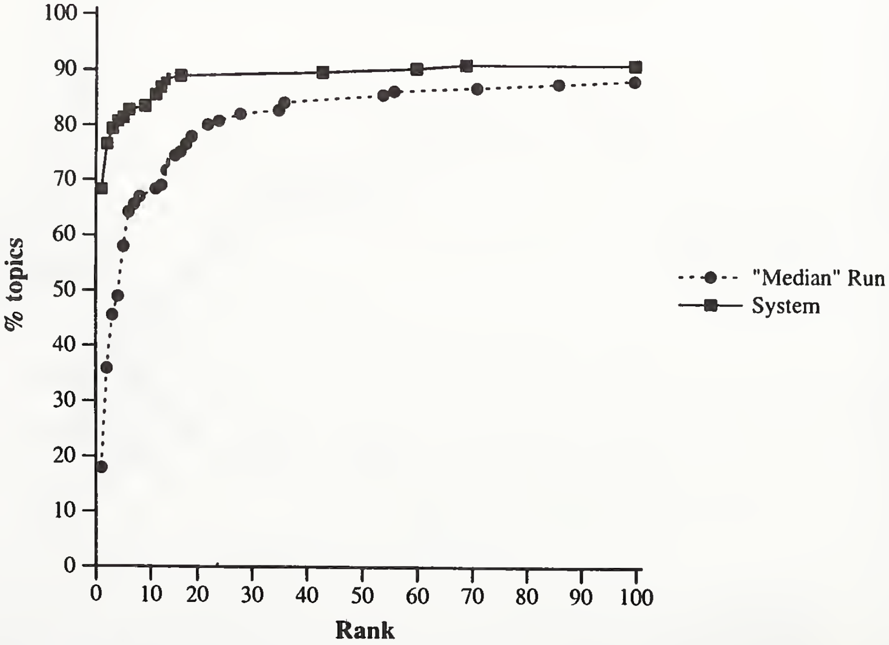
Cumulative %of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	jscbtawep2
Run Description	docstruct-used, urltext-used, links-used
Num topics	145
Mean reciprocal rank	0.769
Num found at rank 1	104 (71.7%)
Num found in top 10	121 (83.4%)
Num not found in top 100	13 (9.0%)



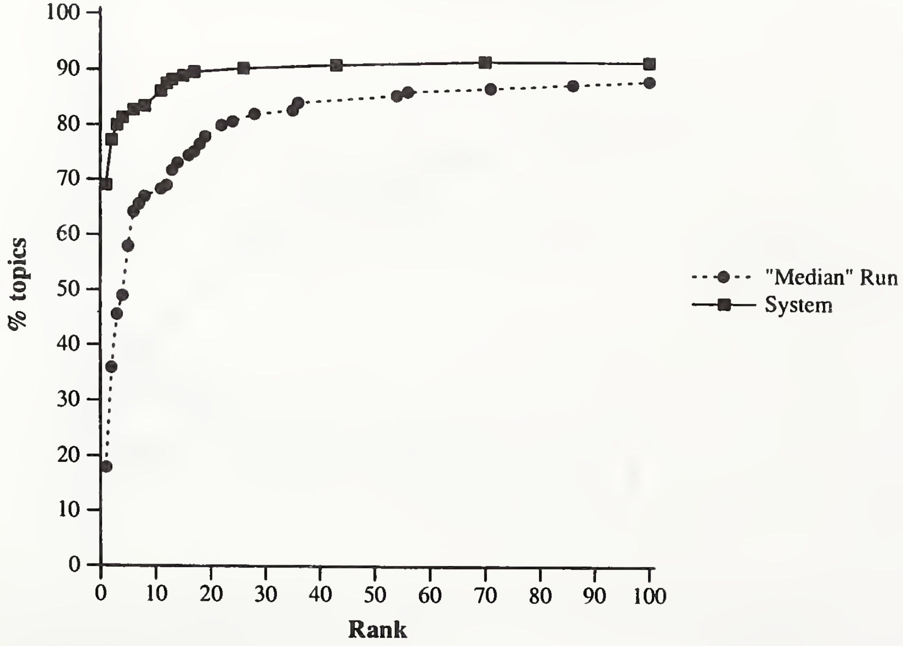
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	jscbtawep3
Run Descriptin	docstruct-used, urltext-used, links-used
Num topics	145
Mean reciprocl rank	0.746
Num found atrank 1	99 (68.3%)
Num found intop 10	121 (83.4%)
Num not founl in top 100	13 (9.0%)



Cumulative % of topics that retrieve homepage by given rank

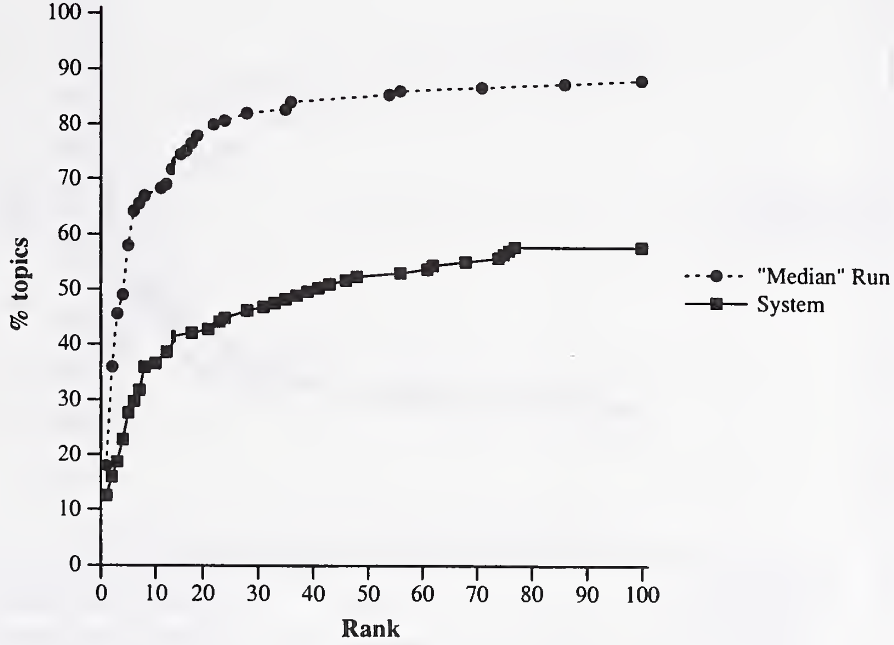
Summary Statistics	
Run ID	jscbtawep4
Run Description	docstruct-used, urltext-used, links-used
Num topics	145
Mean reciprocal rank	0.752
Num found at rank 1	100 (69.0%)
Num found in top 10	121 (83.4%)
Num not found in top 100	12 (8.3%)



Cumulative % of topics that retrieve homepage by given rank

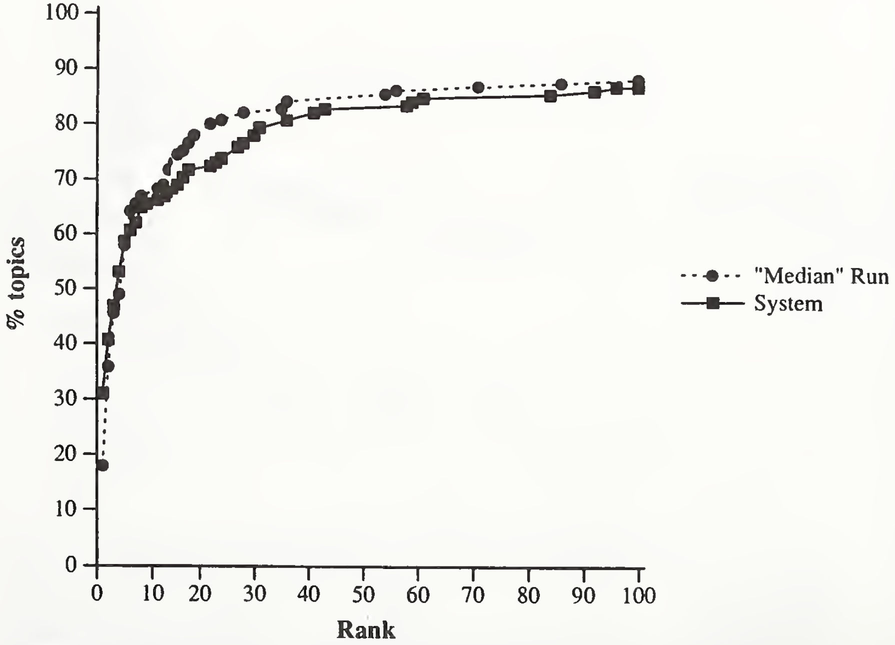
Web track, entrypage finding task results — Kasetsart University

Summary Statistics	
Run ID	kuhpf2001
Run Descriptim	docstruct-notused, urltext-notused, links-notused
Num topics	145
Mean reciproc:l rank	0.191
Num found at :ank 1	18 (12.4%)
Num found in :op 10	53 (36.6%)
Num not founc in top 100	61 (42.1%)



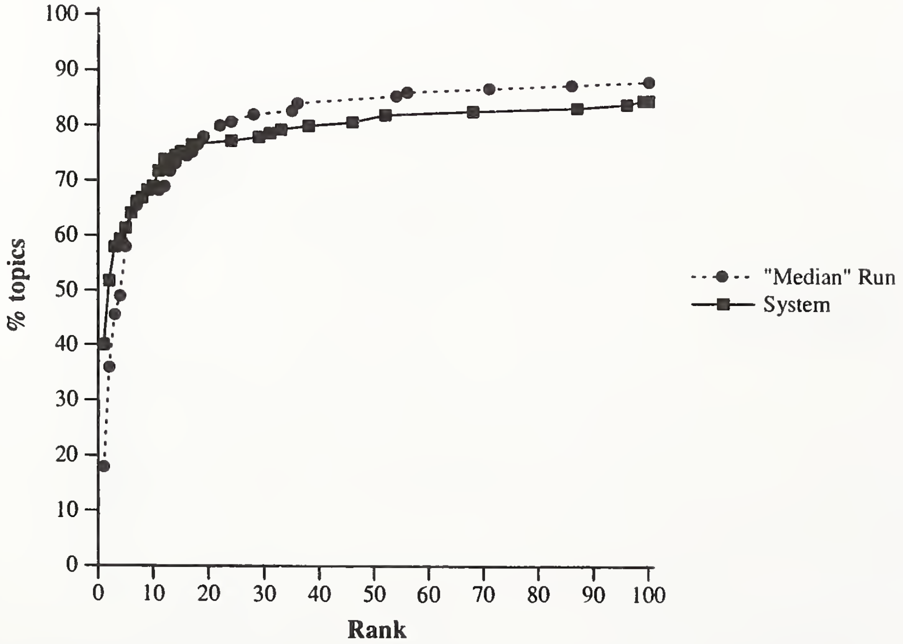
Cumulative %of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	msrcnp1
Run Descriptim	docstruct-used, urltext-used, links-notused
Num topics	145
Mean reciprocd rank	0.424
Num found atrank 1	45 (31.0%)
Num found in :op 10	95 (65.5%)
Num not founc in top 100	19 (13.1%)



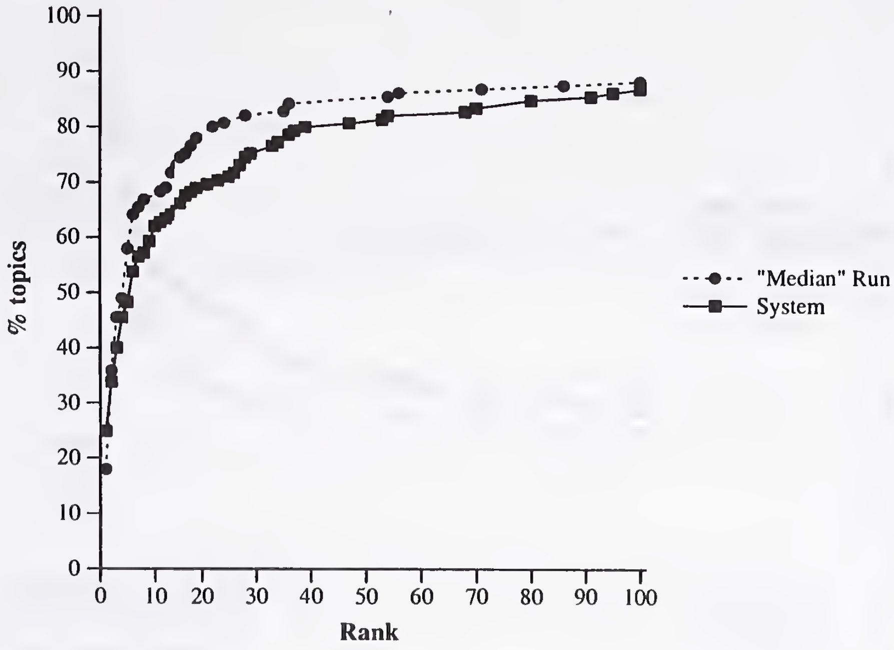
Cumulative %of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	msrcnp2
Run Description	docstruct-used, urltext-used, links-used
Num topics	145
Mean reciprocal rank	0.505
Num found at rank 1	58 (40.0%)
Num found in top 10	100 (69.0%)
Num not found in top 100	22 (15.2%)



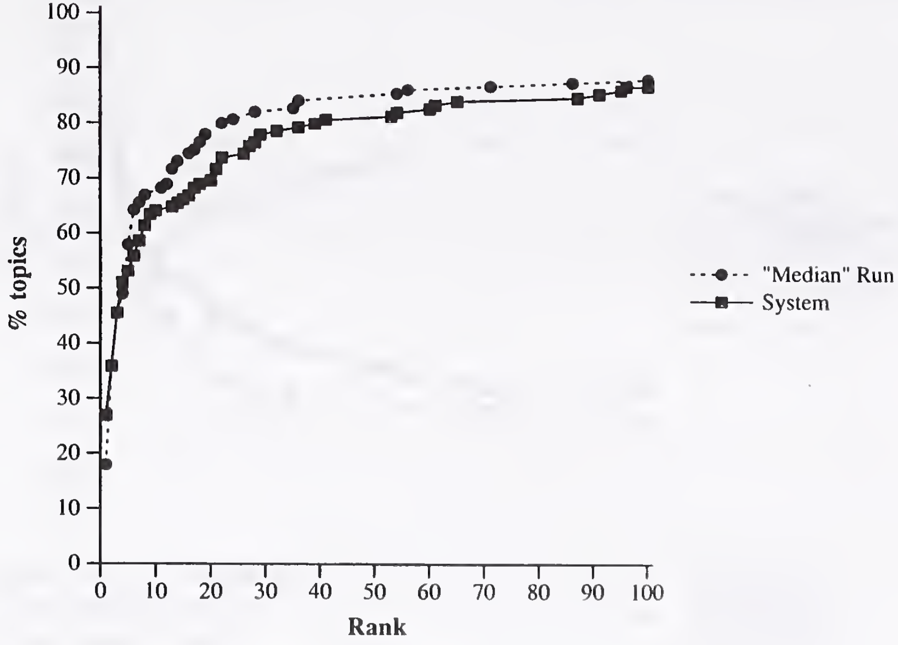
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	ok10wahd0
Run Description	docstruct-notused, urltext-used, links-used
Num topics	145
Mean reciprocal rank	0.362
Num found at rank 1	36 (24.8%)
Num found in top 10	90 (62.1%)
Num not found in top 100	19 (13.1%)



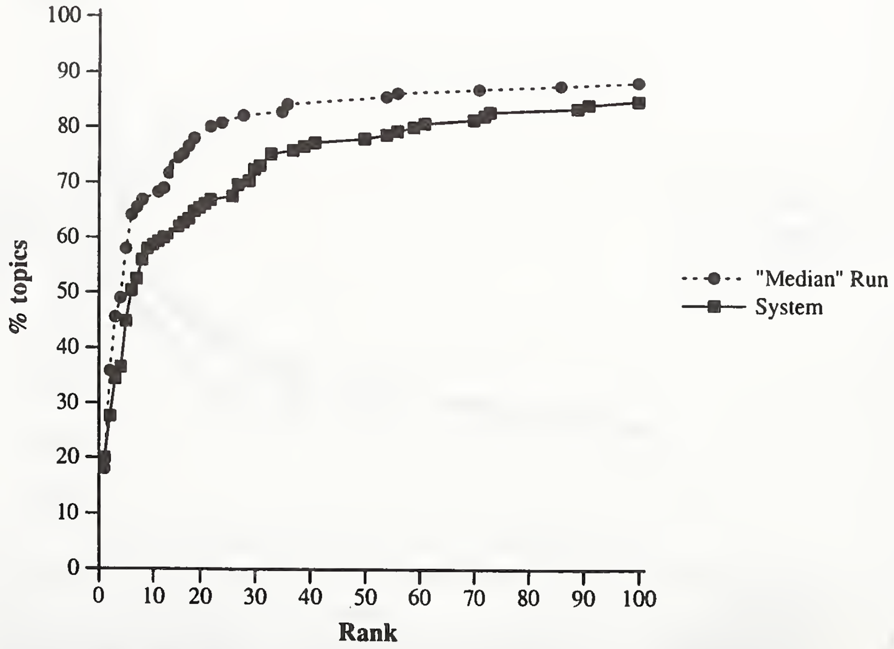
Cumulative %of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	ok10wahd1
Run Description	docstruct-notused, urltext-used, links-used
Num topics	145
Mean reciprocal rank	0.387
Num found at rank 1	39 (26.9%)
Num found in top 10	93 (64.1%)
Num not found in top 100	19 (13.1%)



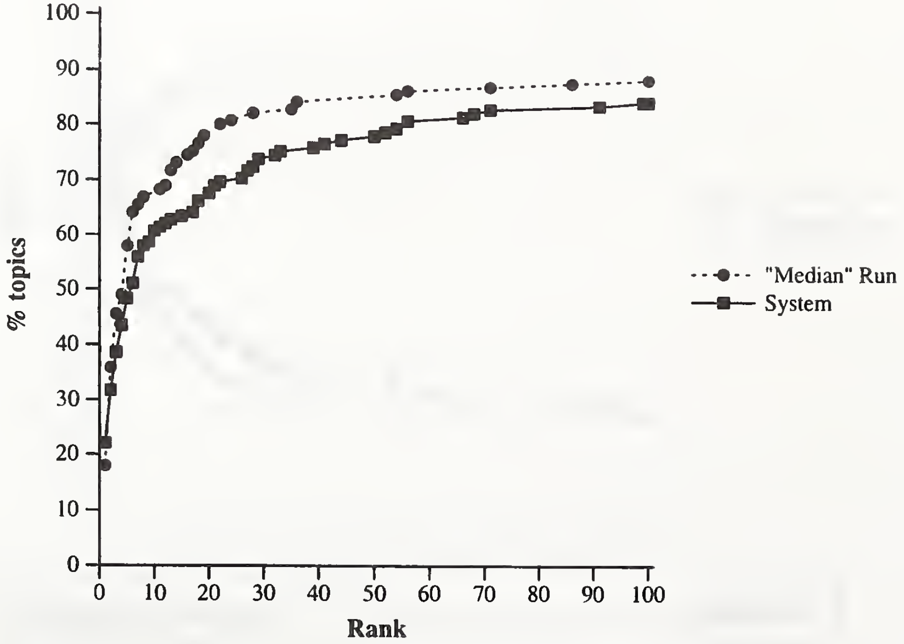
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	ok10whd0
Run Descriptim	docstruct-notused, urltext-used, links-notused
Num topics	145
Mean reciproc:l rank	0.312
Num found at :ank 1	29 (20.0%)
Num found in ,op 10	85 (58.6%)
Num not founc in top 100	22 (15.2%)



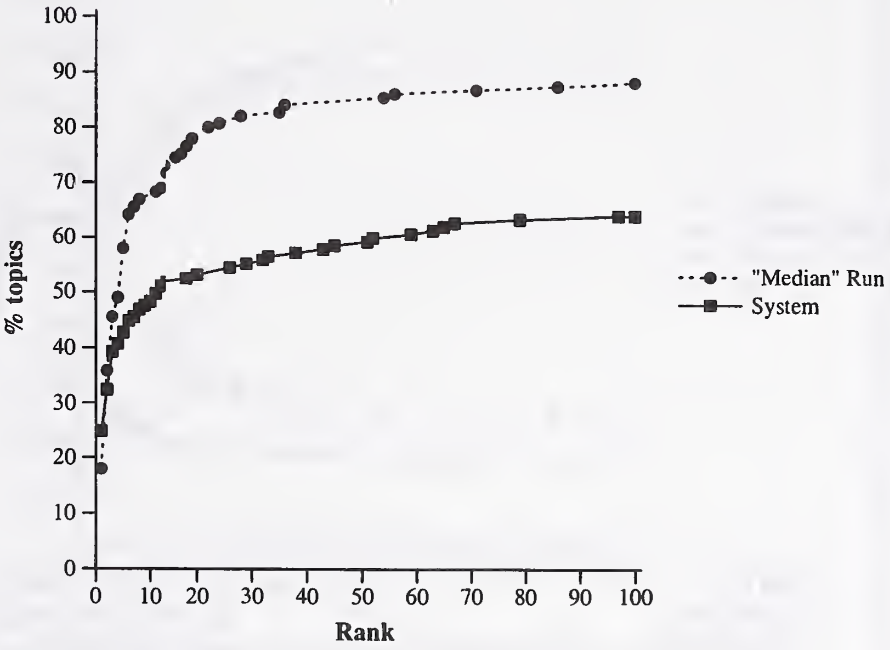
Cumulative %of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	ok10whd1
Run Description	docstruct-notused, urltext-used, links-notused
Num topics	145
Mean reciprocal rank	0.340
Num found at rank 1	32 (22.1%)
Num found in top 10	88 (60.7%)
Num not found in top 100	23 (15.9%)



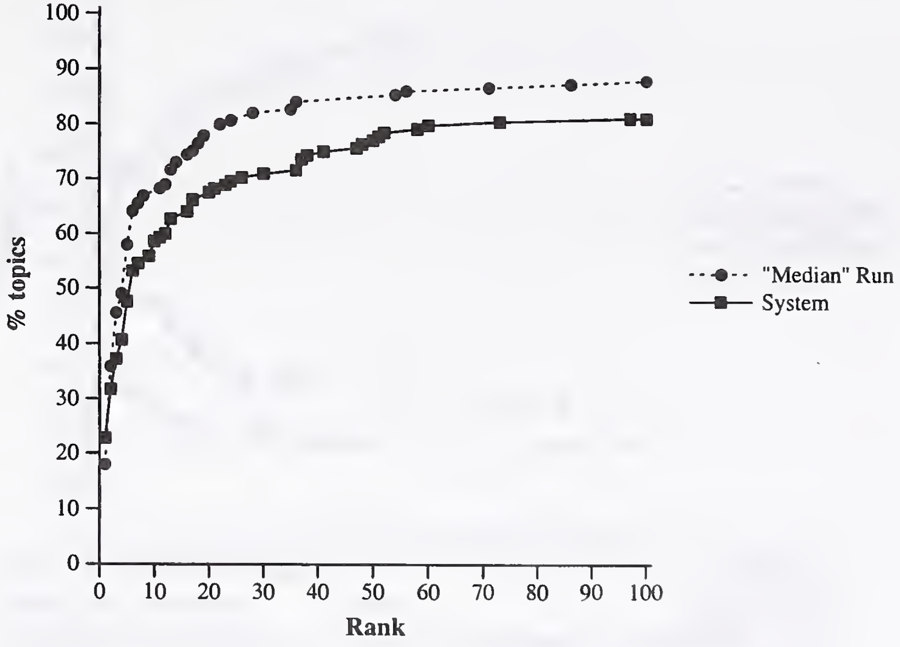
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	tnout10epA
Run Descriptin	docstruct-notused, urltext-notused, links-used
Num topics	145
Mean reciprocl rank	0.331
Num found atrank 1	36 (24.8%)
Num found intop 10	70 (48.3%)
Num not founl in top 100	52 (35.9%)



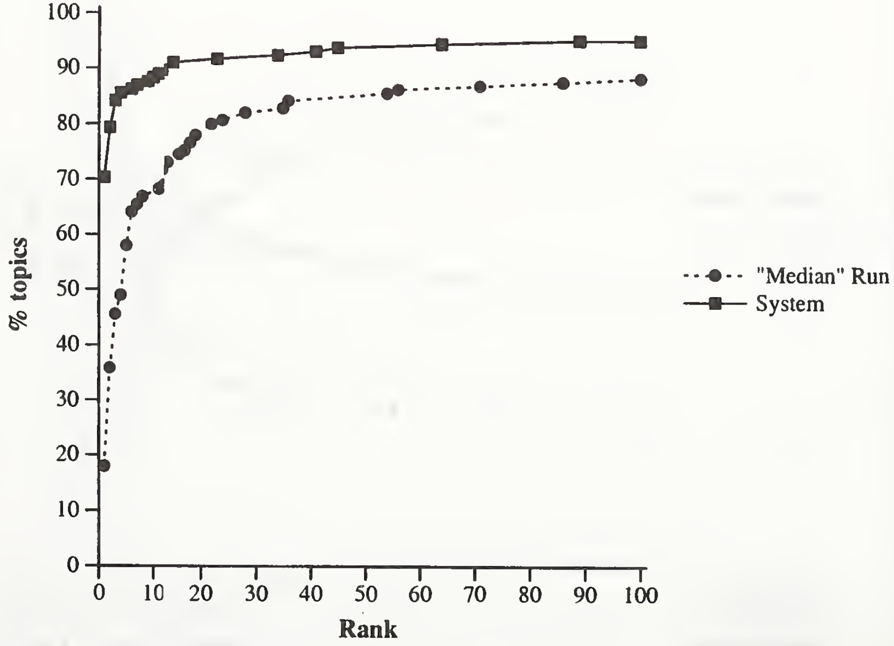
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	tnout10epC
Run Description	docstruct-notused, urltext-notused, links-notused
Num topics	145
Mean reciprocal rank	0.338
Num found at rank 1	33 (22.8%)
Num found in top 10	85 (58.6%)
Num not found in top 100	27 (18.6%)



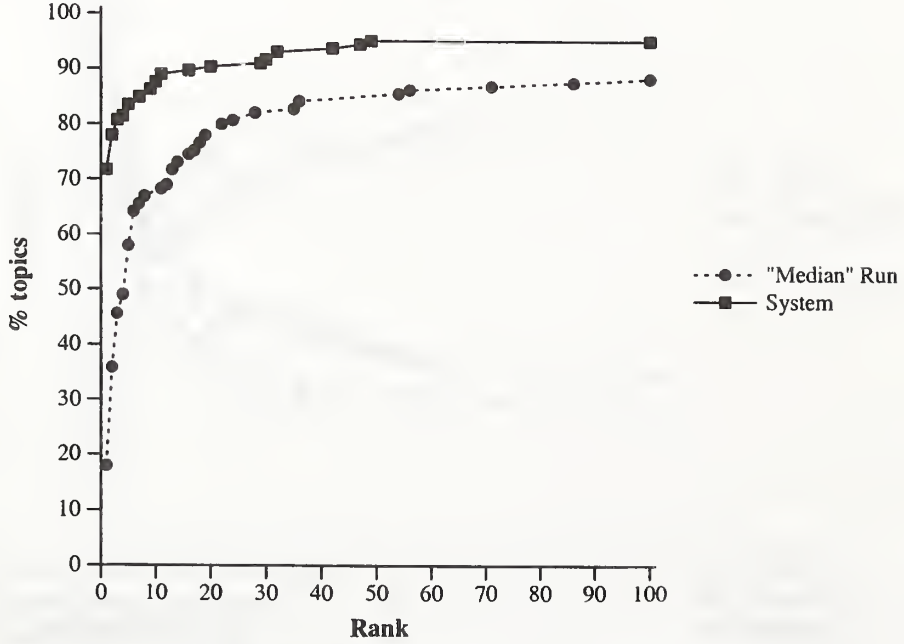
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	tnout10epCAU
Run Descriptbn	docstruct-notused, urltext-used, links-used
Num topics	145
Mean reciprocal rank	0.774
Num found atrank 1	102 (70.3%)
Num found intop 10	128 (88.3%)
Num not found in top 100	7 (4.8%)



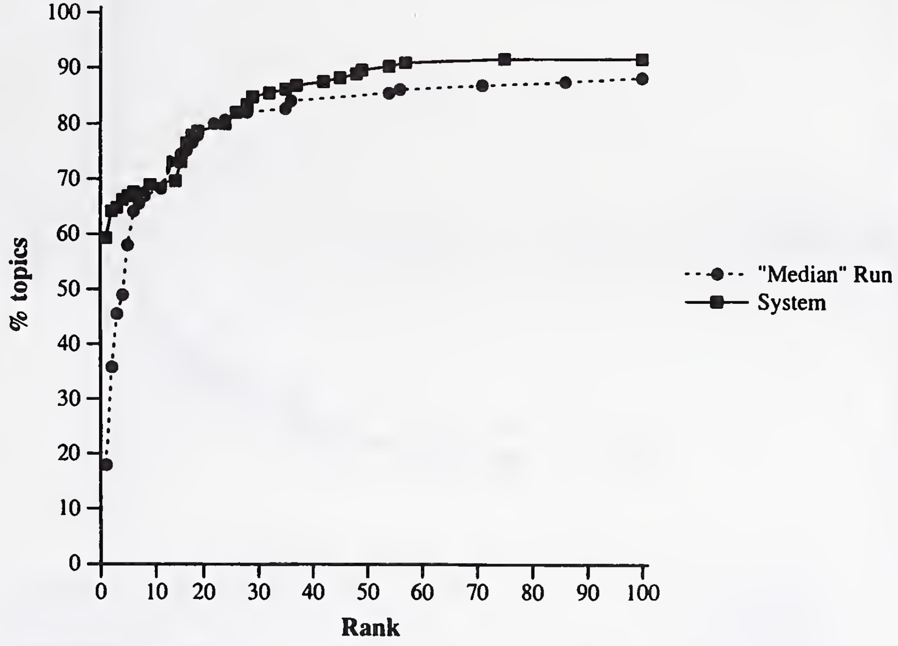
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	tnout10epCU
Run Description	docstruct-notused, urltext-used, links-notused
Num topics	145
Mean reciprocal rank	0.772
Num found at rank 1	104 (71.7%)
Num found in top 10	127 (87.6%)
Num not found in top 100	7 (4.8%)



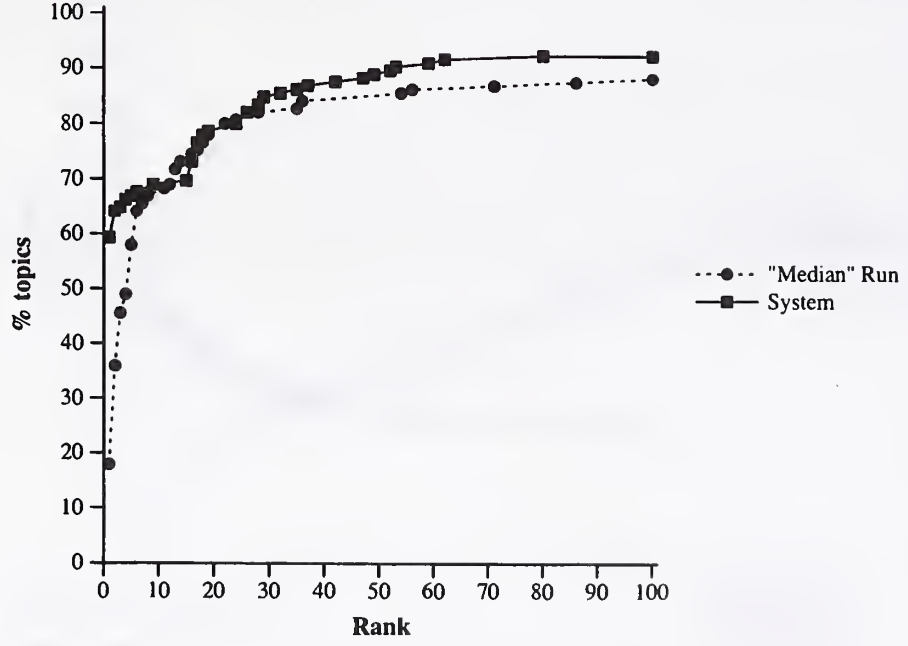
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	UniNEep1
Run Description	docstruct-notused, urltext-used, links-notused
Num topics	145
Mean reciprocal rank	0.637
Num found at rank 1	86 (59.3%)
Num found in top 10	100 (69.0%)
Num not found in top 100	12 (8.3%)



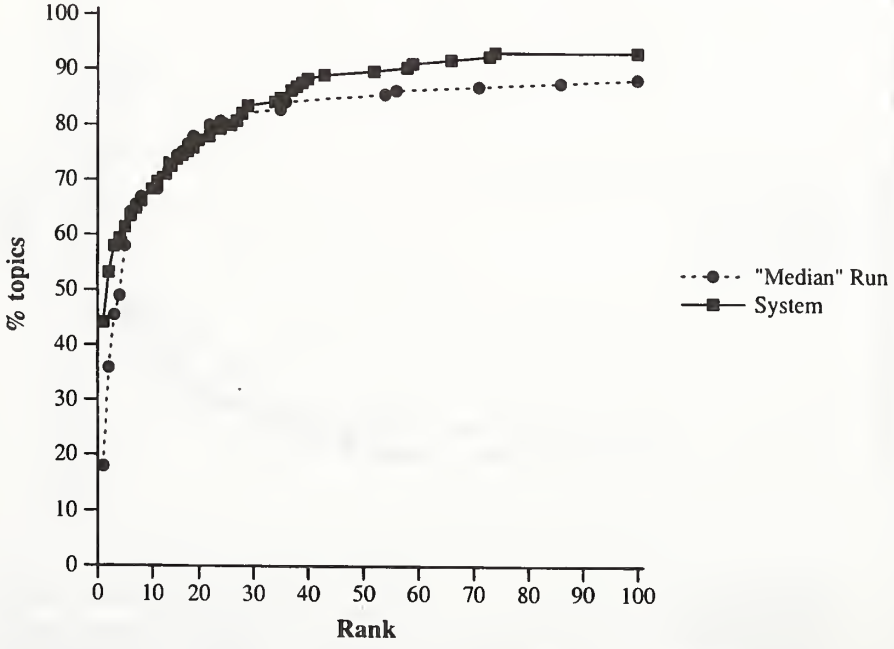
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	UniNEep2
Run Description	docstruct-notused, urltext-used, links-notused
Num topics	145
Mean reciprocal rank	0.637
Num found at rank 1	86 (59.3%)
Num found in top 10	100 (69.0%)
Num not found in top 100	11 (7.6%)



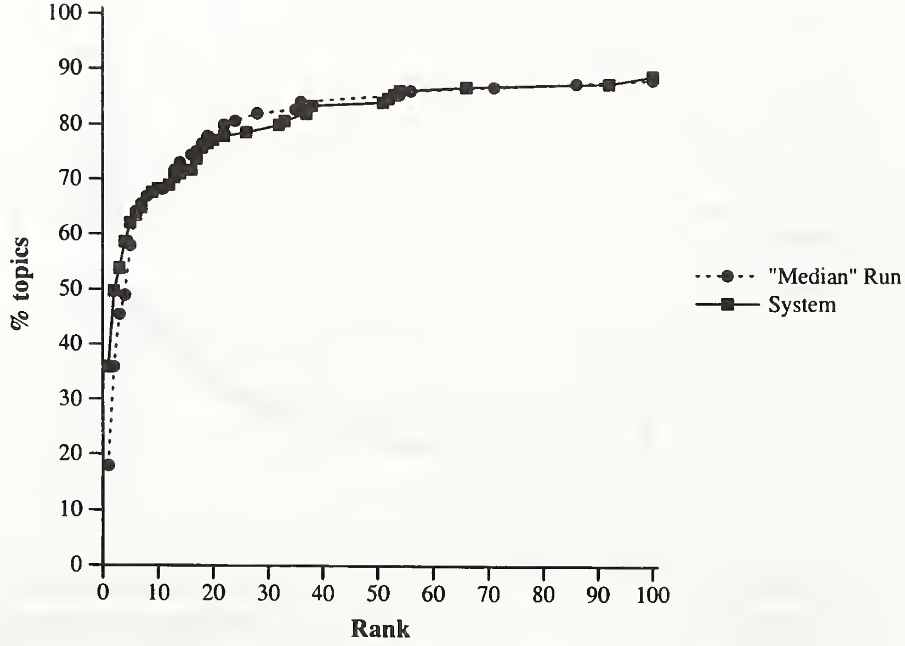
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	UniNEep3
Run Descriptbn	docstruct-notused, urltext-notused, links-notused
Num topics	145
Mean reciproal rank	0.530
Num found atrank 1	64 (44.1%)
Num found intop 10	99 (68.3%)
Num not founl in top 100	10 (6.9%)



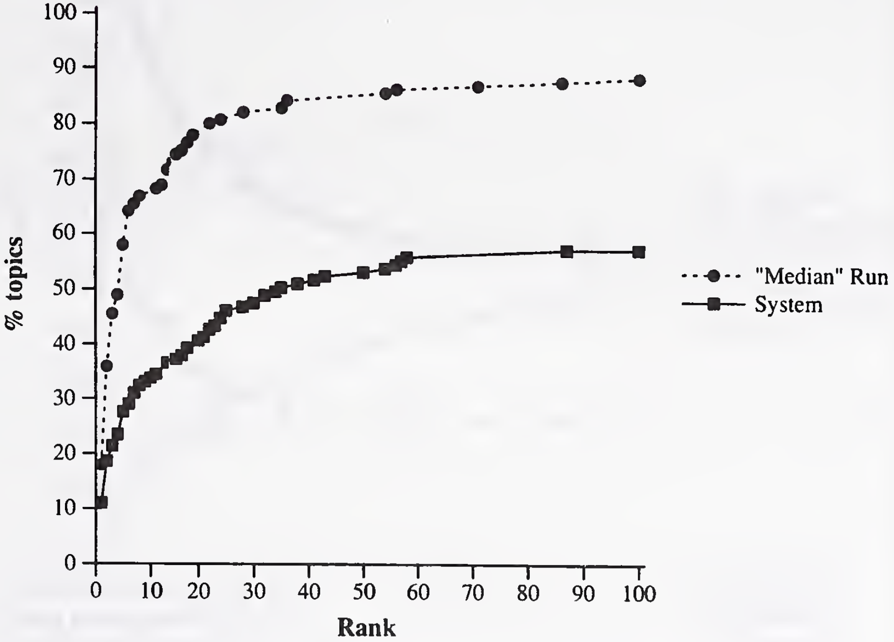
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	UniNEep4
Run Description	docstruct-notused, urltext-used, links-notused
Num topics	145
Mean reciprocal rank	0.477
Num found at rank 1	52 (35.9%)
Num found in top 10	99 (68.3%)
Num not found in top 100	16 (11.0%)



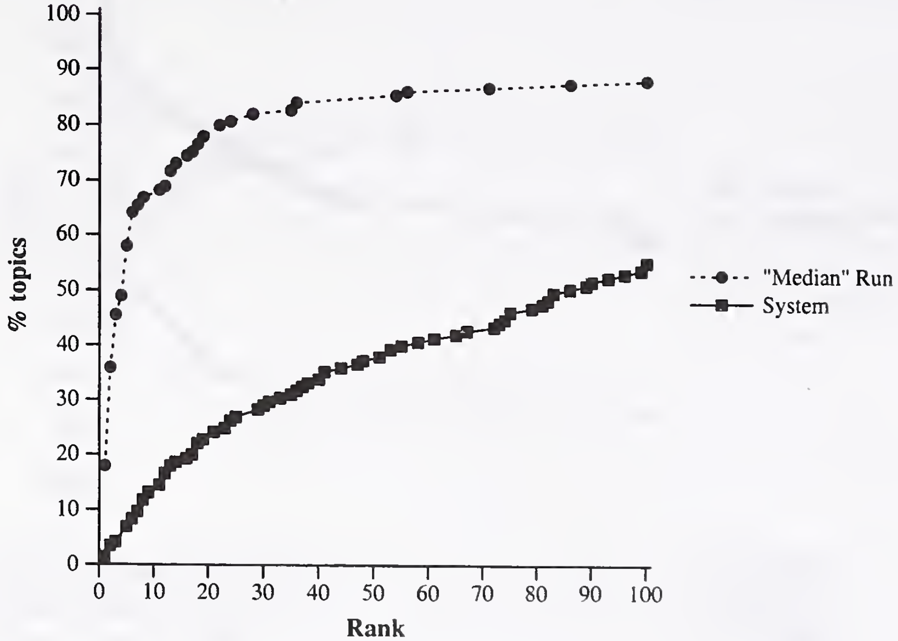
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	PDWTEPDR
Run Descriptin	docstruct-notused, urltext-notused, links-notused
Num topics	145
Mean reciprocl rank	0.189
Num found atrank 1	16 (11.0%)
Num found intop 10	49 (33.8%)
Num not founl in top 100	62 (42.8%)



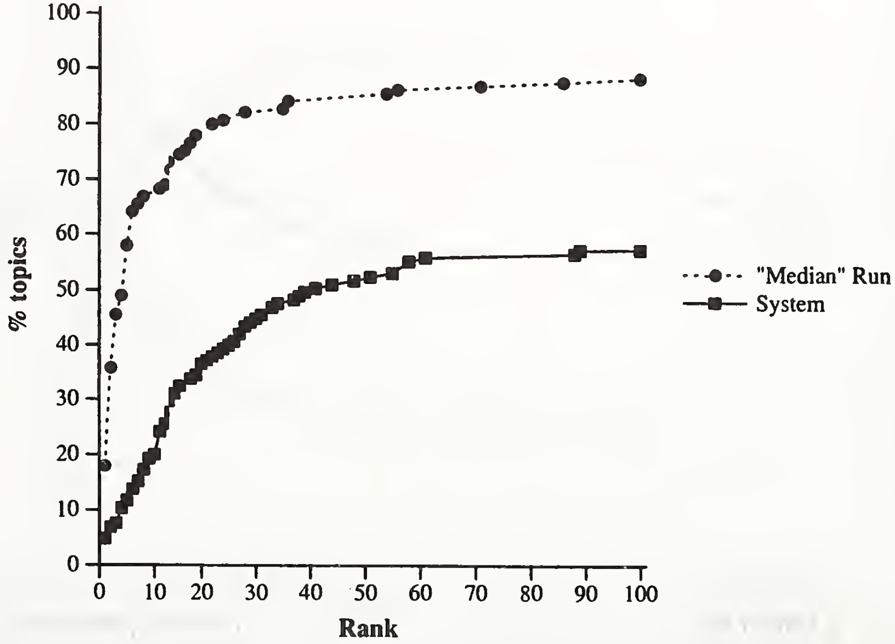
Cumulative %of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	PDWTEPPR
Run Description	docstruct-notused, urltext-notused, links-notused
Num topics	145
Mean reciprocal rank	0.054
Num found at rank 1	2 (1.4%)
Num found in top 10	19 (13.1%)
Num not found in top 100	65 (44.8%)



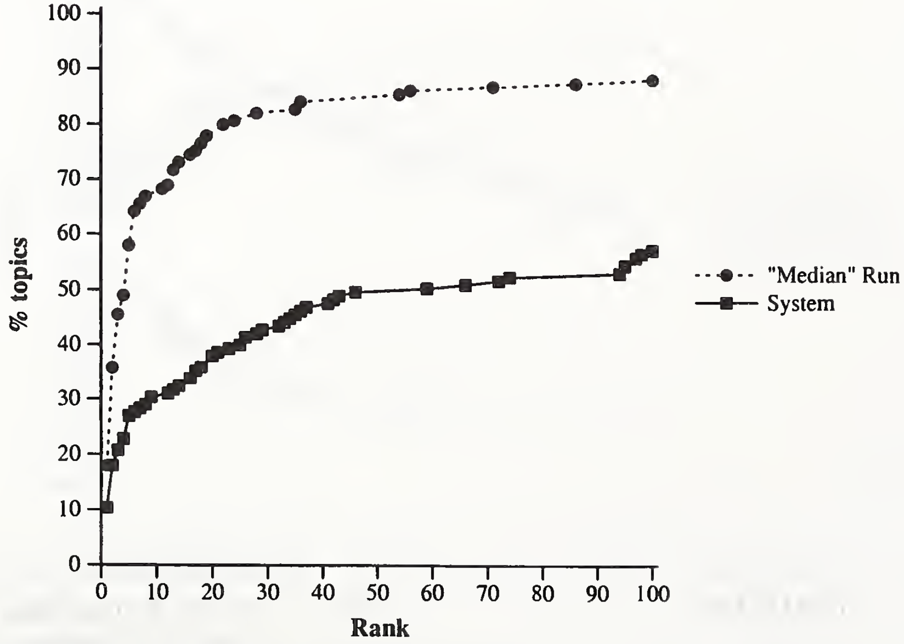
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	PDWTEPTL
Run Descriptin	docstruct-notused, urltext-notused, links-used
Num topics	145
Mean reciprocal rank	0.099
Num found atrank 1	7 (4.8%)
Num found intop 10	29 (20.0%)
Num not founl in top 100	62 (42.8%)



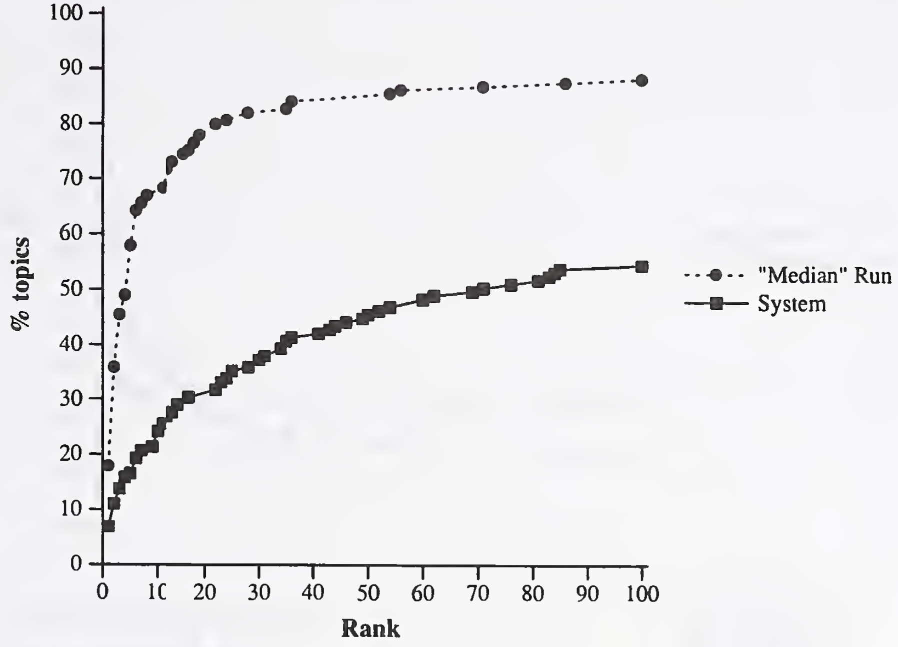
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	PDWTEPWL
Run Description	docstruct-notused, urltext-notused, links-used
Num topics	145
Mean reciprocal rank	0.178
Num found at rank 1	15 (10.3%)
Num found in top 10	44 (30.3%)
Num not found in top 100	62 (42.8%)



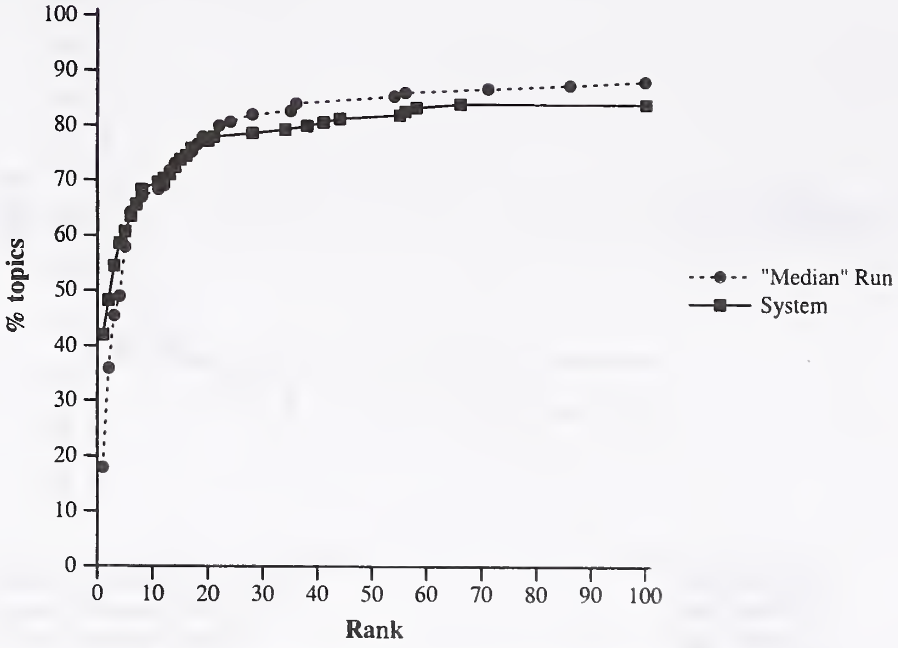
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	VTBASE
Run Descriptbn	docstruct-notused, urltext-notused, links-notused
Num topics	145
Mean reciproal rank	0.126
Num found a rank 1	10 (6.9%)
Num found ir top 10	35 (24.1%)
Num not fourd in top 100	66 (45.5%)



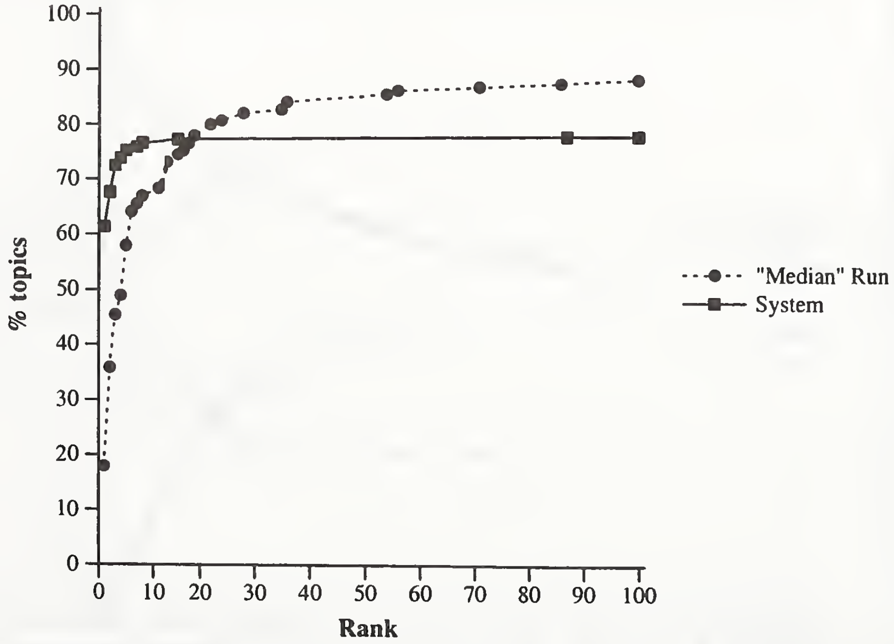
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	VTEP
Run Description	docstruct-notused, urltext-used, links-used
Num topics	145
Mean reciprocal rank	0.506
Num found at rank 1	61 (42.1%)
Num found in top 10	99 (68.3%)
Num not found in top 100	23 (15.9%)



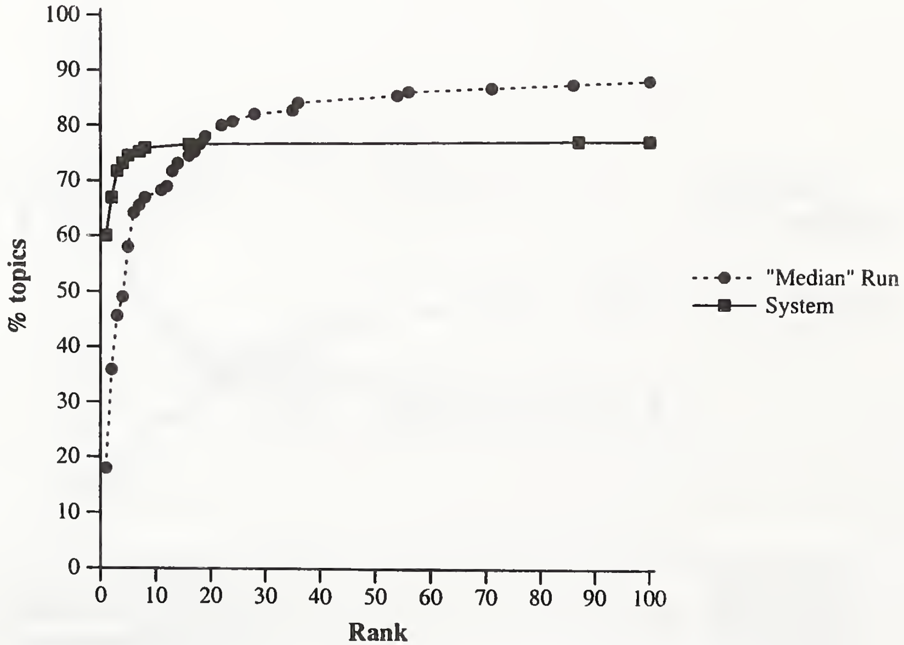
Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	yehp01
Run Descriptbn	docstruct-used, urltext-used, links-used
Num topics	145
Mean reciprocal rank	0.669
Num found atrank 1	89 (61.4%)
Num found intop 10	111 (76.6%)
Num not founl in top 100	32 (22.1%)



Cumulative % of topics that retrieve homepage by given rank

Summary Statistics	
Run ID	yehpb01
Run Description	docstruct-used, urltext-used, links-notused
Num topics	145
Mean reciprocal rank	0.659
Num found at rank 1	87 (60.0%)
Num found in top 10	110 (75.9%)
Num not found in top 100	33 (22.8%)



Cumulative % of topics that retrieve homepage by given rank

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