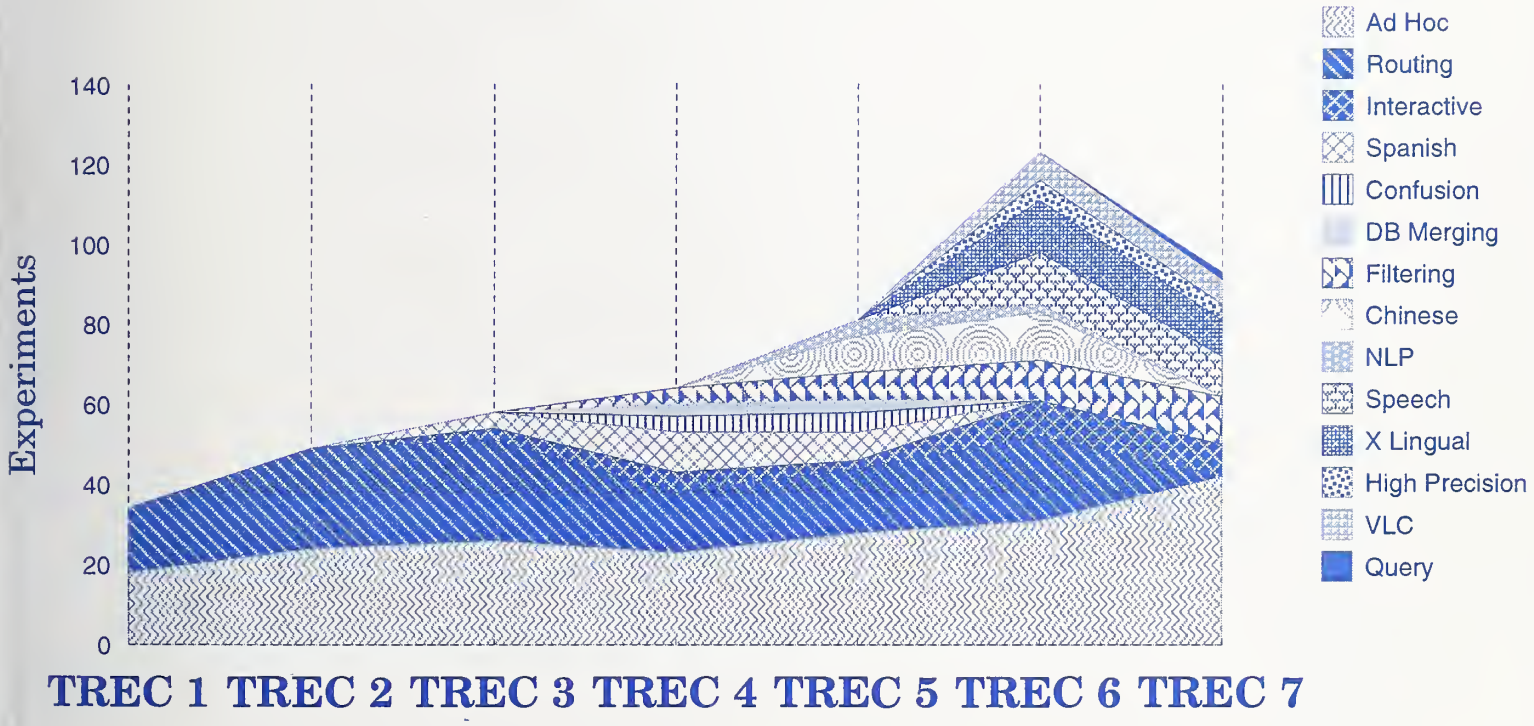
NIST Special Publication 500-242

Information Technology:

The Seventh Text REtrieval Conference (TREC-7)

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



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

NIST

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
- Technology Innovation
- Information Services

Advanced Technology Program
- Economic Assessment
- Information Technology and Applications
- Chemical and Biomedical Technology
- 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
- Electromagnetic Fields[1]
- Electromagnetic Technology[1]
- Optoelectronics[1]

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]

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

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

Building and Fire Research Laboratory
- 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

[1] At Boulder, CO 80303.
[2] Some elements at Boulder, CO.

NIST Special Publication 500-242

Information Technology:

The Seventh Text REtrieval Conference (TREC-7)

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

Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-0001

July 1999

U.S. Department of Commerce
William M. Daley, Secretary

Technology Administration
Gary R. Bachula, Acting Under Secretary for Technology

National Institute of Standards and Technology
Raymond G. Kammer, 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.

National Institute of Standards and Technology
Special Publication 500-242
Natl. Inst. Stand. Technol.
Spec. Publ. 500-242
810 pages (July 1999)
CODEN: NSPUE2

U.S. Government Printing Office
Washington: 1999

For sale by the Superintendent of Documents
U.S. Government Printing Office, Washington, DC 20402-9325

Foreword

This report constitutes the proceedings of the seventh Text REtrieval Conference (TREC-7) held in Gaithersburg, Maryland, November 9–11, 1998. The conference was co-sponsored by the National Institute of Standards and Technology (NIST) and the Defense Advanced Research Projects Agency (DARPA), and was attended by 150 people. Fifty-six groups including participants from 13 different countries and 19 companies were represented. The conference was the seventh in an on-going series of workshops to evaluate new technologies in text retrieval.

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.

The sponsorship of the Information Technology Office of the Defense Advanced Research Projects Agency is gratefully acknowledged, as is the tremendous work of the program committee.

Ellen Voorhees,
Donna Harman
May 4, 1999

TREC-7 Program Committee

Donna Harman, NIST, chair
Nick Belkin, Rutgers University
Chris Buckley, Sabir Research, Inc.
Jamie Callan, University of Massachusetts at Amherst
Susan Dumais, Microsoft
David Hawking, CSIRO
Darryl Howard, U.S. Department of Defense
David Hull, Xerox Research Centre Europe
John Prange, U.S. Department of Defense
Steve Robertson, Microsoft
Peter Schäuble, Swiss Federal Institute of Technology (ETH)
Alan Smeaton, Dublin City University, Ireland
Karen Sparck Jones, University of Cambridge, UK
Tomek Strzalkowski, GE Corporate Research and Development
Howard Turtle, CogiTech
Ellen Voorhees, NIST
Ross Wilkinson, CSIRO

TABLE OF CONTENTS

APPENDICES

ALPHABETICAL INDEX OF TREC-7 PAPERS BY ORGANIZATION

INDEX OF TREC-7 PAPERS BY TASK/TRACK

AD HOC TASK

CROSS-LANGUAGE

FILTERING

HIGH PRECISION

INTERACTIVE

QUERY

SPOKEN DOCUMENT RETRIEVAL

VERY LARGE CORPUS

Abstract

This report constitutes the proceedings of the seventh Text REtrieval Conference (TREC-7) held in Gaithersburg, Maryland, November 9–11, 1998. The conference was co-sponsored by the National Institute of Standards and Technology (NIST) and the Defense Advanced Research Projects Agency (DARPA), and was attended by 150 people. Fifty-six groups including participants from 13 different countries and 19 companies were represented.

The goal of the conference was to bring research groups together to discuss their work on a large test collection. The diversity of the participants meant that a wide variety of retrieval techniques were represented, including new models for retrieval and refined machine learning techniques. Results were scored using a common evaluation package, so groups were able to compare the effectiveness of different techniques, and to discuss how differences between systems affected performance. In addition to the main evaluation, seven additional evaluations, called "tracks," allowed participants to focus on particular common subproblems.

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-7 proceedings web site also contains system descriptions that detail the timing and storage requirements of the different runs.

Overview of the Seventh Text REtrieval Conference (TREC-7)

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

1 Introduction

The seventh Text REtrieval Conference (TREC-7) was held at the National Institute of Standards and Technology (NIST) on November 9–11, 1998. The conference was co-sponsored by NIST and the Information Technology Office of the Defense Advanced Research Projects Agency (DARPA) as part of the TIPSTER Text Program.

TREC-7 is the latest in a series of workshops designed to foster research in text retrieval. For analyses of the results of previous workshops, see Sparck Jones [7], Tague-Sutcliffe and Blustein [9], and Harman [2]. In addition, the overview paper in each of the previous TREC proceedings summarizes the results of that TREC.

The TREC workshop series has the following goals:

- to encourage research in text retrieval 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.

Table 1 lists the groups that participated in TREC-7. Fifty-six groups including participants from 13 different countries and 19 companies were represented. The diversity of the participating groups has ensured that TREC represents many different approaches to text retrieval. The emphasis on individual experiments evaluated within a common setting has proven to be a major strength of TREC.

This paper serves as an introduction to the research described in detail in the remainder of the volume. It concentrates mainly on the main task, *ad hoc retrieval*, which is defined the next section. Details regarding the test collections and evaluation methodology used in TREC follow in sections 3 and 4, while section 5 provides an overview of the ad hoc retrieval results. In addition to the main ad hoc task, TREC-7 contained seven "tracks," tasks that focus research on particular subproblems of text retrieval. Taken together, the tracks represent the bulk of the experiments performed in TREC-7. However, each track has its own overview paper included in the proceedings, so this paper presents only a short summary of each track in section 6. The final section looks forward to future TREC conferences.

2 The Ad Hoc Task

The ad hoc task investigates the performance of systems that search a static set of documents using new questions (called *topics* in TREC). This task is similar to how a researcher might use a library—the collection is known but the questions likely to be asked are not known. NIST provides the participants approximately 2 gigabytes worth of documents and a set of 50 natural language topic statements. The participants produce a set of *queries* from the topic statements and run those queries against the documents. The output from

Table 1: Organizations participating in TREC-7

ACSys Cooperative Research Centre	Management Information Technologies, Inc.
AT&T Labs Research	Massachusetts Institute of Technology
Avignon CS Laboratory/Bertin	National Tsing Hua University
BBN Technologies	NEC Corp. and Tokyo Institute of Technology
Canadian Imperial Bank of Commerce	New Mexico State University
Carnegie Mellon University	NTT DATA Corporation
Commissariat à l'Energie Atomique	Okapi Group (City U./U. of Sheffield/Microsoft)
CLARITECH Corporation	Oregon Health Sciences University
Cornell University/SabIR Research, Inc.	Queens College, CUNY
Defense Evaluation and Research Agency	RMIT/Univ. of Melbourne/CSIRO
Eurospider	Rutgers University (2 groups)
Fondazione Ugo Bordoni	Seoul National University
FS Consulting, Inc.	Swiss Federal Institute of Technology (ETH)
Fujitsu Laboratories, Ltd.	TextWise, Inc.
GE/Rutgers/SICS/Helsinki	TNO-TPD TU-Delft
Harris Information Systems Division	TwentyOne
IBM — Almaden Research Center	Universite de Montreal
IBM T.J. Watson Research Center (2 groups)	University of California, Berkeley
Illinois Institute of Technology	University of Cambridge
Imperial College of Science, Technology and Medicine	University of Iowa
Institut de Recherche en Informatique de Toulouse	University of Maryland
The Johns Hopkins University — APL	University of Massachusetts, Amherst
Kasetsart University	University of North Carolina, Chapel Hill
KDD R&D Laboratories	Univ. of Sheffield/Cambridge/SoftSound
Keio University	University of Toronto
Lexis-Nexis	University of Waterloo
Los Alamos National Laboratory	U.S. Department of Defense

this run is the official test result for the ad hoc task. Participants return the best 1000 documents retrieved for each topic to NIST for evaluation.

Participants are free to use any method they desire to create the 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.

The right answers, called *relevance judgments*, for the ad hoc topics are not known at the time the participants produce their runs, though they may use the documents, topics, and relevance judgments from previous TRECs to develop their systems. Participants are also free to use other sources of training data if they desire. Topics 351–400 were created for the TREC-7 ad hoc task. The set of documents used in the task was those contained on TREC Disks 4 and 5, excluding the *Congressional Record* subcollection. See Section 3.1 for details about this document set.

Participants were allowed to submit up to three ad hoc runs to NIST. The runs could differ as the result of using different query construction techniques, or using different searching methods with the same queries. When submitting a run, participants were required to state whether the queries were produced manually or automatically. If an automatic method was used, participants also stated what parts of the topic statement

Table 2: Document collection statistics. Words are strings of alphanumeric characters. No stop words were removed and no stemming was performed.

	Size (megabytes)	# Docs	Median # Words/Doc	Mean # Words/Doc
Disk 1				
Wall Street Journal, 1987–1989	267	98,732	245	434.0
Associated Press newswire, 1989	254	84,678	446	473.9
Computer Selects articles, Ziff-Davis	242	75,180	200	473.0
Federal Register, 1989	260	25,960	391	1315.9
abstracts of U.S. DOE publications	184	226,087	111	120.4
Disk 2				
Wall Street Journal, 1990–1992 (WSJ)	242	74,520	301	508.4
Associated Press newswire (1988) (AP)	237	79,919	438	468.7
Computer Selects articles, Ziff-Davis (ZIFF)	175	56,920	182	451.9
Federal Register (1988) (FR88)	209	19,860	396	1378.1
Disk 3				
San Jose Mercury News, 1991	287	90,257	379	453.0
Associated Press newswire, 1990	237	78,321	451	478.4
Computer Selects articles, Ziff-Davis	345	161,021	122	295.4
U.S. patents, 1993	243	6,711	4445	5391.0
Disk 4				
the *Financial Times*, 1991–1994 (FT)	564	210,158	316	412.7
Federal Register, 1994 (FR94)	395	55,630	588	644.7
Congressional Record, 1993 (CR)	235	27,922	288	1373.5
Disk 5				
Foreign Broadcast Information Service (FBIS)	470	130,471	322	543.6
the *LA Times*	475	131,896	351	526.5

were used (see Section 3.2).

3 The Test Collections

Like most traditional retrieval collections, there are three distinct parts to the collections used in TREC: the documents, the topics, and the relevance judgments. This section describes each of these pieces for the ad hoc collection.

3.1 Documents

TREC documents are distributed on CD-ROM's with approximately 1 GB of text on each, compressed to fit. For TREC-7, Disks 1–5 were all available as training material (see Table 2) and Disks 4–5 were used for the ad hoc task. The *Congressional Record* subcollection on Disk 4 was excluded from the test document set.

Documents are tagged using SGML to allow easy parsing (see fig. 1). The documents in the different datasets have been tagged with identical major structures, but they have different minor structures. The philosophy in the formatting at NIST is to leave the data as close to the original as possible. No attempt is made to correct spelling errors, sentence fragments, strange formatting around tables, or similar faults.

3.2 Topics

The format of the TREC topics has evolved over time as illustrated in Table 3. The table shows the number

```
<DOC>
<DOCNO>FT911-3</DOCNO>
<PROFILE>AN-BEOA7AAIFT</PROFILE>
<DATE>910514
</DATE>
<HEADLINE>
FT 14 MAY 91 / International Company News:  Contigas plans DM900m east German
project
</HEADLINE>
<BYLINE>
By DAVID GOODHART
</BYLINE>
<DATELINE>
BONN
</DATELINE>
<TEXT>
CONTIGAS, the German gas group 81 per cent owned by the utility Bayernwerk, said
yesterday that it intends to invest DM900m (Dollars 522m) in the next four years
to build a new gas distribution system in the east German state of Thuringia.  ...
</TEXT>
</DOC>
```

Figure 1: A document extract from the *Financial Times*.

```
<num> Number:  396
<title> sick building syndrome

<desc> Description:
Identify documents that discuss sick building syndrome or building-related
illnesses.

<narr> Narrative:
A relevant document would contain any data that refers to the sick building
or building-related illnesses, including illnesses caused by asbestos, air
conditioning, pollution controls.  Work-related illnesses not caused by the
building, such as carpal tunnel syndrome, are not relevant.
```

Figure 2: A sample TREC-7 topic.

of words included in the different parts of the topic statements for each TREC. The original ad hoc topics (51–150) were very detailed, containing multiple fields and lists of concepts related to the topic subject. The ad hoc topics used in TREC-3 (151–200) did not contain the concept lists and and the remaining fields were generally shorter than in earlier topics. Nonetheless, participants in TREC-3 felt that the topics were still too long compared with what users normally submit to operational retrieval systems. The TREC-4 topics (201–250) were therefore made even shorter: a single field consisting of a one sentence description of the information need. However, the one-sentence topic eliminated from the topic the statement of the criteria used to judge a document as relevant—which was one of the motivating factors for providing topic statements rather than queries. The last three sets of ad hoc topics (251–400) have therefore all had the same format as in TREC-3, consisting of a title, description, and narrative. A sample TREC-7 topic is shown in Figure 2.

The different parts in the most recent TREC topics allow participants to investigate the effect of different

Table 3: Topic length statistics by topic section. Lengths count number of tokens in topic statement including stop words.

	Min	Max	Mean
TREC-1 (51–100)	44	250	107.4
title	1	11	3.8
description	5	41	17.9
narrative	23	209	64.5
concepts	4	111	21.2
TREC-2 (101–150)	54	231	130.8
title	2	9	4.9
description	6	41	18.7
narrative	27	165	78.8
concepts	3	88	28.5
TREC-3 (151–200)	49	180	103.4
title	2	20	6.5
description	9	42	22.3
narrative	26	146	74.6
TREC-4 (201–250)	8	33	16.3
description	8	33	16.3
TREC-5 (251–300)	29	213	82.7
title	2	10	3.8
description	6	40	15.7
narrative	19	168	63.2
TREC-6 (301–350)	47	156	88.4
title	1	5	2.7
description	5	62	20.4
narrative	17	142	65.3
TREC-7 (351–400)	31	114	57.6
title	1	3	2.5
description	5	34	14.3
narrative	14	92	40.8

query lengths on retrieval performance. The "titles" in topics 301–400 were specially designed to allow experiments with very short queries. The titles consist of up to three words that best describe the topic. The description field is a one sentence description of the topic area. For TREC-7 (topics 351–400), the description field contains all of the words in the title field, to remove the confounding effects of word choice on length experiments as was exhibited in TREC-6 [11]. The narrative gives a concise description of what makes a document relevant.

Ad hoc participants who used automatic query construction techniques were required to use particular parts of the topics in TREC-5 and TREC-6. The TREC-7 task description had no such requirements, but participants did have to report what parts they used when they submitted their runs.

Ad hoc topics have been constructed by the same person who performed the relevance assessments for that topic since TREC-3. Each assessor comes to NIST with ideas for topics based on his or her own interests, and searches the ad hoc collection (looking at approximately 100 documents per topic) to estimate the likely number of relevant documents per candidate topic. NIST personnel select the final 50 topics from among the candidates based on having a range of estimated number of relevant documents and balancing the load across assessors.

Table 4: Overlap of submitted results

	Possible	Actual	Relevant
TREC-1	3300	1279 (39%)	277 (22%)
TREC-2	4000	1106 (28%)	210 (19%)
TREC-3	2700	1005 (37%)	146 (15%)
TREC-4	7300	1711 (24%)	130 (08%)
ad hoc	4000	1345	115
confusion	900	205	0
dbmerge	800	77	2
interactive	1600	84	13
TREC-5	10,100	2671 (27%)	110 (04%)
ad hoc	7700	2310	104
dbmerge	600	72	2
NLP	1800	289	3
TREC-6	3,430	1445 (42%)	92 (06%)
ad hoc	3100	1326	89
NLP	200	113	2
HP	130	6	1
TREC-7	7,805	1611 (21%)	93 (06%)
ad hoc	7700	1605	92
HP	105	6	.5

3.3 Relevance assessments

Relevance judgments are of critical importance to a test collection. For each topic it is necessary to compile a list of relevant documents—as comprehensive a list as possible. All TRECs have used the pooling method [8] to assemble the relevance assessments. In this method a pool of possible relevant documents is created by taking a sample of documents selected by the various participating systems. This pool is then shown to the human assessors. The particular sampling method used in TREC is to take the top 100 documents retrieved in each submitted run for a given topic and merge them into the pool for assessment. This is a valid sampling technique since all the systems used ranked retrieval methods, with those documents most likely to be relevant returned first.

3.3.1 Overlap

Table 4 summarizes the amount of overlap in the ad hoc pool for each of the seven TRECs. The first data column in the table gives the maximum possible size of the pool. Since the top 100 documents from each run are judged, this number is usually 100 times the number of runs used to form the pool. However, high precision track runs contribute fewer documents. The next column shows the number of documents that were actually in the pool (i.e., the number of unique documents retrieved in the top 100 across all judged runs) averaged over the number of topics. The percentage given in that column is the size of the actual pool relative to the possible pool size. The final column gives the average number of relevant documents in the pool and the percentage of the actual pool that was relevant. Starting in TREC-4, various tracks also contributed documents to the ad hoc pool. These are broken out in the appropriate rows within Table 4. The order of the tracks is significant in the table—a document retrieved in a track listed later is not counted for that track if the document was also retrieved by a track listed earlier.

TREC-6 is clearly an outlier in Table 4. The tremendous drop in the size of the ad hoc pool reflects the difference in the number of runs NIST was able to assess that year. The overlap for the TREC-6 runs was less than in previous years, and this coupled with less time for assessing meant that NIST could only judge one ad hoc run per group. The overlap in TREC-7 was as high as in earlier years, so NIST was able to judge two ad hoc runs per group.

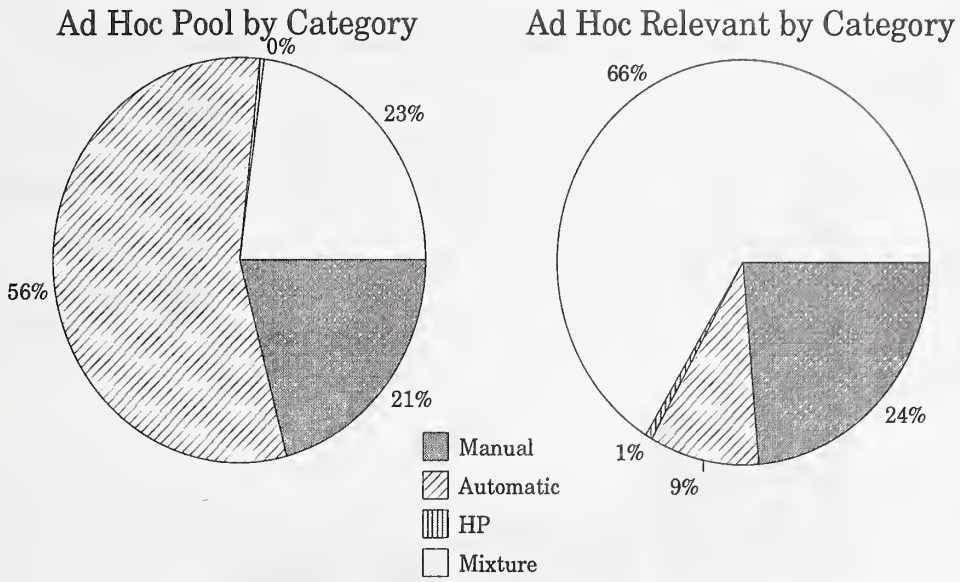


Figure 3: Distribution of categories in judged and relevant document pools.

Table 4 also shows that the average number of relevant documents per topic has decreased over the years to its current number. NIST has deliberately chosen more tightly focused topics to better guarantee the completeness of the relevance assessments.

3.3.2 Uniquely retrieved documents

The average overlap figures given in Table 4 hide details about the source of the documents in the pool. Figures 3 and 4 show two breakdowns of the sources of the relevant documents.

Figure 3 shows the percentages contributed by each type of ad hoc run (and high precision track) to the judgment pool, and to the relevant documents. For example, whereas 56% of the pool for relevance judgments came from the automatic systems, 9% of the relevant documents were found by only automatic systems. The manual systems contributed 21% of the pool, and 24% of the relevant documents were found only by manual systems.

Figure 4 gives a different view of the same issue by looking at the systems that retrieved the most unique relevant documents (i.e., relevant documents that were contributed to the pool by exactly one group). Almost all of the unique documents were retrieved by manual runs. Note that the pattern of the sources of unique relevant documents is very similar to the pattern found for TREC-5 [10].

4 Evaluation

The entire purpose of building a test collection is to be able to evaluate the effectiveness of retrieval systems. Providing a common evaluation scheme is an important element of TREC.

4.1 Current practice

All TREC tasks that involve returning a ranked list of documents are evaluated using the `trec_eval` package. This package, written by Chris Buckley, reports about 85 different numbers for a run. The measures reported include *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

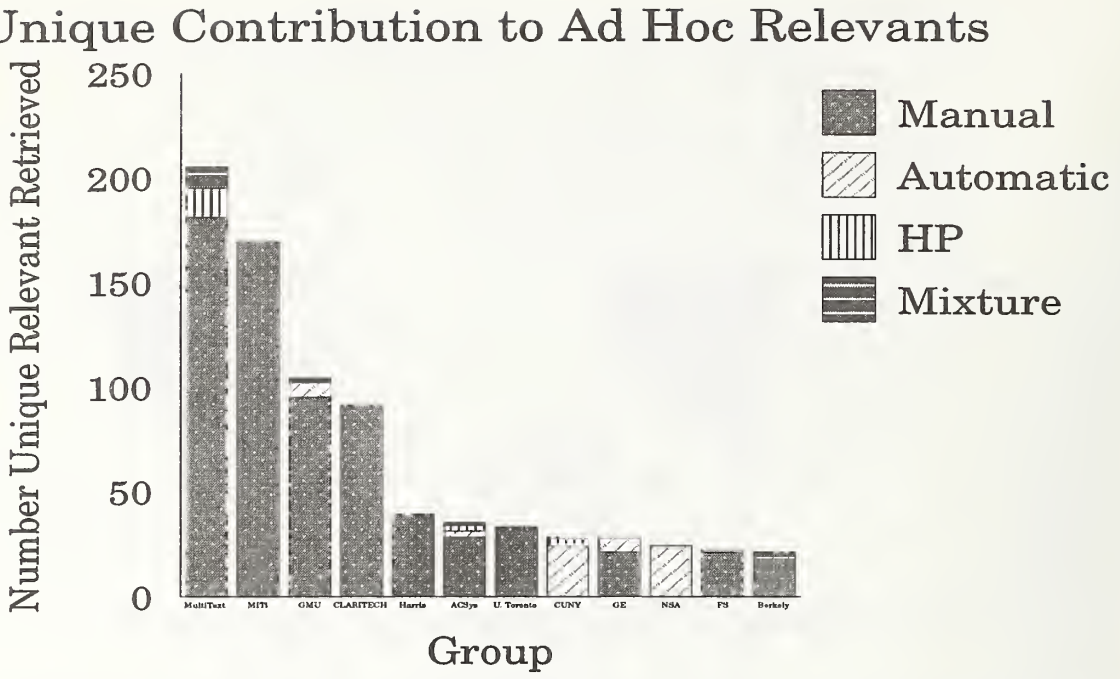


Figure 4: Percentage of unique relevant documents by category for groups retrieving more than 20 unique relevant documents.

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.

This overview paper generally uses two evaluation measures when discussing retrieval results, the recall-precision curve and mean (non-interpolated) average precision. A recall-precision curve plots precision as a function of recall as shown, for example, in Figure 5. 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.

Table 5: Kendall's tau correlations between pairs of system rankings.

	P(30)	R-Prec	Mean Ave Precision	.5 Prec	Recall R(1000)	Total Rel Ret	Rank of 1st Rel
P(10)	.8851	.8151	.7899	.7855	.7817	.7718	.6378
P(30)		.8676	.8446	.8238	.7959	.7915	.6213
R-Prec			.9245	.8654	.8342	.8320	.5896
Mean Ave Prec				.8840	.8473	.8495	.5612
Recall at .5 Prec					.7707	.7762	.5349
R(1000)						.9212	.5891
Total Rel Ret							.5880

The (reformatted) output of `trec_eval` for each submitted run is given in Appendix A. In addition to the ranked results, participants are also asked to submit data that describes their system features and timing figures to allow a primitive comparison of the amount of effort needed to produce the corresponding retrieval results. These system descriptions are not included in the printed version of the proceedings due to their size, but they are available on the TREC web site (http://trec.nist.gov).

4.2 Comparison of evaluation measures

The `trec_eval` program reports so many different numbers as the evaluation of a single run because there are so many different features of a run that might be of interest. To better understand what aspect of retrieval behavior different effectiveness measures capture, NIST used the TREC-7 automatic ad hoc results to compute correlations between pairs of measures.

The correlations are given in Table 5 and were computed in the following way. Eight different measures (described below) were used in the study. Each run was evaluated using each measure, where the score for a measure was usually the average score for that measure over the 50 topics. The runs were then ranked by score for each measure. The correlation between two different measures was defined as the Kendall's tau correlation between the respective rankings. Kendall's tau computes the distance between two rankings as the minimum number of pairwise adjacent swaps to turn one ranking into the other. The distance is normalized by the number of items being ranked such that two identical rankings produce a correlation of 1.0, the correlation between a ranking and its perfect inverse is -1.0, and the expected correlation of two rankings chosen at random is 0.0.

The following measures were used in the study:

P(10): The precision after the first 10 documents are retrieved.

P(30): The precision after the first 30 documents are retrieved.

R-Prec: The precision after the first R documents are retrieved, where R is the number of relevant documents for the current topic.

Mean Ave Precision: Mean (non-interpolated) average precision as defined above.

Recall at .5 Prec: Recall at the rank where precision first dips below .5 (after at least 10 documents have been retrieved). This measure reflects the heuristic that users will keep looking at a result set while there are more relevant than non-relevant documents being retrieved.

R(1000): The recall after 1000 documents are retrieved.

Total Rel Ret: The total number of relevant documents retrieved across all 50 topics (not an average). The difference between this measure and R(1000) is in the averaging. R(1000) is averaged such that each topic is weighted equally, while the total number of relevant retrieved is dominated by topics that have many relevant documents.

Rank 1st Rel: The rank at which the first relevant document is retrieved.

The correlations between the different measures are all at least .5, showing that each pair of measures is at least somewhat correlated. This is not surprising since all the measures were designed to reflect the quality of a retrieval run. The very high correlation between R(1000) and Total Rels Ret is also not surprising, though the fact the correlation is not 1.0 demonstrates that averaging does have an effect. The weakest correlations are between the Rank 1st Rel measure and each of the others. This is an indication that the Rank 1st Rel measure is in fact a poor measure of retrieval performance. The measure is unstable both because a single topic can have an unreasonable effect on the average score, and because large differences in a score do not reflect the importance of that difference to the user. For example, ranking the first relevant document at rank 103 vs. ranking the first relevant document at rank 957 will cause a large difference in the average score while being essentially meaningless to a user.

One of the current debates in IR is whether recall is important outside a few specific applications such as patent searching. Those who question the utility of recall argue that users never look beyond the "first screen" of results and therefore the only measure that matters is precision at some small cut-off level. Proponents of recall point out that a measure such as P(10) is too coarse-grained for system tuning, even when P(10) is the final measure of interest. The only change in a document ranking that affects P(10) is a relevant document entering or leaving the top 10, while the mean average precision measure is sensitive to the entire ranking. The correlation between P(10) and mean average precision cannot answer which is a better measure, but does show that they measure different things. (The correlation of .7899 represents 384 swaps out of a maximum possible swaps of 3655 since the rankings consist of 86 different runs.)

5 Ad Hoc Retrieval Results

One of the important goals of the TREC conferences is that the participating groups freely devise their own experiments within the TREC task. For some groups this means doing the ad hoc task with the goal of achieving high retrieval effectiveness. For other groups, however, the goals are more diverse and may mean experiments in efficiency or unusual ways of using the data.

This overview of the results discusses the effectiveness of the systems and analyzes some of the similarities and differences in the approaches that were taken. In all cases, readers are referred to the system papers in this proceedings for more details.

The TREC-7 ad hoc evaluation used new topics (topics 351–400) against the documents on Disks 4 and 5 minus the *Congressional Record* documents. There were 103 sets of official results for ad hoc evaluation in TREC-7. Of these, 86 used automatic construction of queries and 17 used manual query construction.

5.1 Automatic runs

Figure 5 shows the recall/precision curves for the eight TREC-7 groups with the highest mean average precision using automatic construction of queries. The runs are ranked by average precision and only one run is shown per group. These graphs (and others in this section) are not intended to show specific comparison of results across sites but rather to provide a focal point for discussion of methodologies used in TREC. For more details on the various runs and procedures, please see the cited papers in this proceedings.

ok7ax – OKAPI group ("Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive" by S.E. Robertson, S. Walker, and M. Beaulieu) continued their experiments with the BM25 weighting technique that has been so successful. They tried some experiments with term proximity in the topic, with little success. This particular run is a weighted linear combination of runs made using title, title + description, and full versions of the topic, with pseudo-feedback expansion used in each of these runs before combining. A corrected version of the run (minus the effects of manual index terms in the LA Times) had only slightly degraded performance (see paper for corrected table).

att98atdc – AT&T Labs Research ("AT&T at TREC-7" by A. Singhal, J. Choi, D. Hindle, D. Lewis, and F. Pereira) made two major changes to their TREC-6 algorithms. The first one involved some changes in the term weighting to better accommodate a mix of single terms and phrases (a new phrase list was

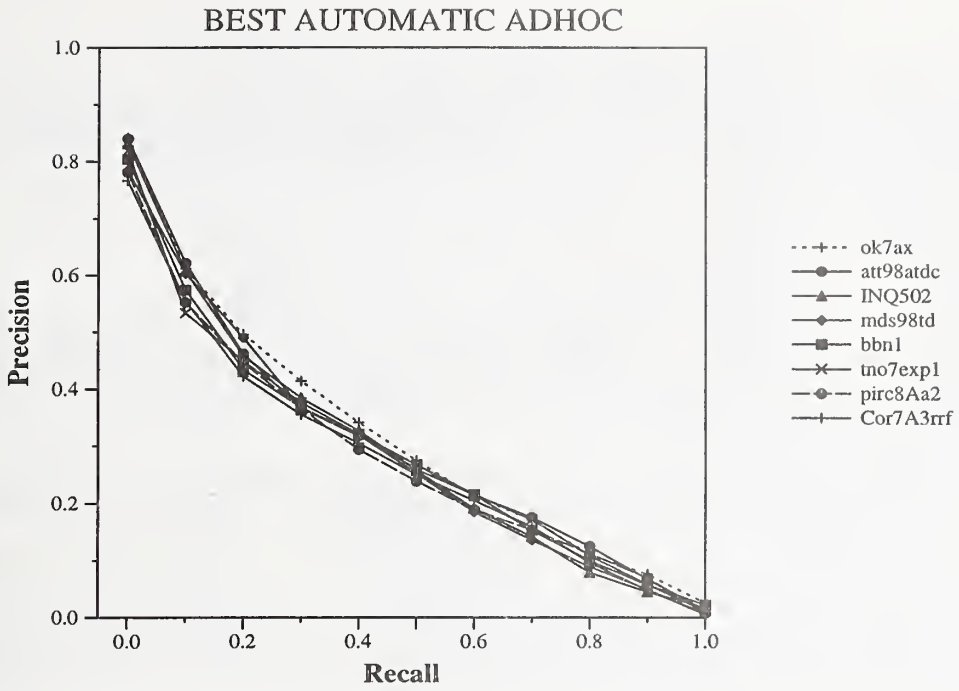


Figure 5: Recall/Precision graph for the top eight automatic ad hoc runs.

used). The second was a change in the automatic expansion methods that allows the use of a larger corpus (all 5 disks) without incurring too much query drift. Whereas the average precision increases by only 3% using this new expansion algorithm, there are fewer topics that are hurt than when using the older expansion method, therefore making it more predictable in actual operational settings.

INQ502 – University of Massachusetts ("INQUERY and TREC-7" by J. Allan, J. Callan, M. Sanderson, J. Xu, and S. Wegmann) ran both the INQUERY phrase recognition and the LCA query expansion (with passages) that had been used in TREC-6. The phrases improved performance by 3.6% in TREC-7, consistent with past work, and the LCA expansion continued to work well, improving performance by 18.5% over non-expanded queries. New for TREC-7 was the use of title terms as a filter to rerank the passages selected for expansion. This filter improved results by 2.8%, but had significant performance effects (positive and negative) on specific topics.

mds98td – MDS/CSIRO ("TREC-7 Ad Hoc, Speech, and Interactive tracks at MDS/CSIRO" by M. Fuller, M. Kaszkiel, D. Kim, C. Ng, J. Robertson, R. Wilkinson, M. Wu and J. Zobel) continued their explorations with the MG system. This year they used two-term statistical phrases, Rocchio relevance feedback, and also tested using passages vs documents. The OKAPI weighting was used for document ranking, and a cosine similarity function was used for passage matching. Their top ranked run used the title and description as input, used documents but not passages, and incorporated phrases and Rocchio expansion. They report a 4.1% improvement for use of phrases, and a 36% gain for expansion. The use of passages instead of documents gave the same results for titles but decreased performance (7.1%) for titles + descriptions.

bbn1 – BBN Technologies ("BBN at TREC-7: Using Hidden Markov Models for Information Retrieval" by D. Miller, T. Leek, and R. Schwartz) based their work on their experience with Hidden Markov models (HMMs) in speech and other language-related recognition problems. This model adapted well to the information retrieval task, working better than the basic OKAPI model without expansion, and still remaining competitive when compared to the full OKAPI results. This was the first entry of BBN to TREC, and they creatively refined the HMM model to handle phrases (bigrams) and pseudo-relevance feedback. These refinements did not improve their results as much as for other IR models; this may

Table 6: Characteristics of best automatic ad hoc runs.

Organization	Topic Parts	D + T	T only	Full Topic	Comments
Okapi group	T,D,N	0.281	0.253 (-10%)	0.284 (1%)	fused run-0.296
AT&T Labs Research	T,D	0.296	0.249 (-16%)		
U. Mass	T,D,N	0.252		0.274 (9%)	title filtered run-0.282
RMIT/UM/CSIRO	T,D	0.281	0.220 (-22%)	0.285 (1%)	
BBN	T,D,N			0.280	
TwentyOne	T,D,N			0.279	
CUNY	T,D,N	0.254	0.243 (-4%)	0.266 (5%)	with phrases-0.272
Cornell/SabIR	T,D,N	0.254*	0.239 (-6%)	0.267 (5%)	*description only

be because the basic HMM model already incorporates to some extent these techniques, or may mean that there are larger improvements to be expected after further work with this new model.

tno7exp1 – Twenty-One group ("Twenty-One at TREC-7: Ad-hoc and Cross-Language track" by D. Hiemstra and W. Kraaij) based their work on the vector-space model but developed a new weighting algorithm using a linguistically motivated probabilistic model. Their new algorithm in its basic form outperformed the basic OKAPI/SMART algorithm by 8%; incorporation of Rocchio-based feedback improved their results by 12%. All runs were made using the full topic.

pirc8Aa2 – Queens College, CUNY ("TREC-7 Ad-Hoc, High Precision and Filtering Experiments using PIRCS" by K.L. Kwok, L. Grunfeld, M. Chan and N. Dinstl) continued work with their spreading activation model using a sequence of 5 different methods to improve their basic results. These methods included the avtf weighting used in TREC-6, a variable Zipf threshold for selecting indexing terms, collection enrichment by using all 5 disks for query expansion, use of the LCA algorithm for expansion, and a reweighting of the query terms using the documents retrieved from this expansion. This group also made extensive investigations into the effects of input query length on results from different methods.

Cor7A3rrf – Cornell/SabIR Research ("SMART High Precision: TREC 6" by C. Buckley, M. Mitra, J. Walz and C. Cardie) tried many variations on their algorithms, including re-examining stemming, phrases, or alternative document clustering methods to do the final term selection from the top retrieved documents. They also investigated differential weighting for titles or for beginnings of documents. None of these variations improved performance significantly and this run uses the TREC-6 clustering approach.

Note that most of these runs use all parts of the topic (att98atdc and mds98td use only the title and description). However there is now a smaller performance difference between runs that use the full topic and runs that use only the title and description sections than was seen in earlier TRECs. This is most likely due to improved query expansion methods, but could be due to variations across topic sets. Table 6 shows the results (official and unofficial as reported in the papers) of these groups using the different topic parts. It should be noted that the improvement going to the full topic is only 1% for several groups. The decrease in performance using only the title is more marked, ranging from 4% to 22%. The TREC-7 title results should be a truer measure of the effects of using the title only than TREC-6, where the descriptions were often missing key terms. However, it is not clear how representative these titles are with respect to very short user inputs and therefore title results should best be viewed as how well these systems could perform on very short, but very good user input.

Looking at individual topic results shows a less consistent picture. Table 7 shows the number of topics that had the best performance from among a group's three runs using different input lengths. (Note that the "long" run for the Okapi system is actually their fused run, and that these are their "official" results as opposed to their corrected ones.) Not only is there a wide variation across topics, there is also a wide

Table 7: Number of topics performing best by topic length.

	Long	Desc	Title
Okapi	28	13	9
CUNY	27	10	13
Cornell	22	17	11

Table 8: More characteristics of best automatic ad hoc runs.

Organization	Model	Weighting/Similarity	Phrase Imp.	Comments
Okapi group	probabilistic	BM25	minimal*	*last reported in TREC-5
AT&T Labs Research	vector	pivot*		*byte normalization
U. Mass	inference net	belief function	3.6%	
RMIT/UM/CSIRO	vector	BM25/cosine		phrases used
BBN	HMM	probabilistic	2%	bigram phrases
TwentyOne	vector	new probabilistic		no phrases used
CUNY	spread. act.	avtf/RSV	2%	phrases used for reranking
Cornell/SabIR	vector	pivot		

variation across systems in that topics that work best at a particular length for one group did not necessarily work best at that length for the other groups.

The merged run from Okapi is taking advantage of this variation by fusing the results from the final runs for each topic length, and they gain a 4% improvement from this fusion. The group from Lexis-Nexis ("Experiments in Query Processing at LEXIS-NEXIS for TREC-7" by A.G. Rao, T. Humphrey, A. Parhizgar, C. Wilson, and D. Pliske) has also worked for several years using fusion between different methods of processing the initial topic, various term weighting algorithms, and different query expansion methods.

Table 8 shows additional characteristics of the systems. These top eight systems are derived from many models and use different term weighting algorithms and similarity measures. Of particular note here is that new models and term weighting algorithms are still being developed, and these are competitive with the more established methods. This applies both to new variations on old weighting algorithms, such as the double log tf weighting from AT&T, and to more major variations such as the new weighting algorithm from TNO, and the completely new retrieval model from BBN.

The fourth column of the table shows the widespread use of phrases in addition to single terms, but the minimal improvement from their use. The biggest improvement reported in the papers was 3.6% from UMass. Whereas most of the other groups are also using phrases, many did not bother to test for differences due to minimal results in earlier years. Cornell reported 7.7% improvement in TREC-6, but this is the improvement on top of the initial baseline, not the improvement after expansion. Private conversations with several of these groups indicate that these improvements are likely to be much less if measured after expansion. As is often the case, these minimal changes in the averages cover a wide variation in phrase performance across topics. A special run by the Okapi group (many thanks) showed less than a 1% average difference in performance, but 19 topics helped by phrases, 14 hurt, and the rest unchanged. Whereas the benefit of phrases is not proven, they are likely to remain a permanent tool in the retrieval systems in a manner similar to the earlier adoption of stemming.

It is interesting to note that many of these groups are using different phrase "gathering" techniques. The Okapi group has a manually-built phrase list with synonym classes that has slowly grown over the years based on mostly past TREC topics. The automatically-produced UMass phrase list was new for TREC-6, the Cornell list was basically unchanged from early TRECs, and the BBN list was based on a new bigram model.

Table 9 shows characteristics of the expansion tools used in these systems. The second column gives the basic expansion model, with the vector-based systems using the Rocchio expansion and other systems using

Table 9: Characterization of query expansion used in best automatic ad hoc runs.

Organization	Expansion/Feedback	Top Docs/Terms added	Disks used	Comments
Okapi group	probabilistic	Full–15/30 T+D–10/30 T only–6/20+title	1-5	
AT&T Labs Research	Rocchio	10/20+5 phrases	1-5	conservative enrichment
U. Mass	LCA	30P/50	1-5	reranking using title terms before expansion
RMIT/UM/CSIRO	Rocchio	10/40+5 phrases	?	additional experiments with passages
BBN	HMM-based	6/?	?	differential weighting on topic parts
TwentyOne	Rocchio	3/200	?	
CUNY	LCA	200P/?	1-5	
Cornell/SabIR	Rocchio	30/25	4-5	clustering, reranking

expansion models more suitable to their retrieval model. The third column shows the number of top-ranked documents (P if passages were used), and the number of terms added from these documents. It should be noted that these numbers are more similar than in earlier TRECs. The fourth column shows the source of the documents being mined for terms; note that most groups have now moved to retrieval from a wider range of documents. Of particular note is the AT&T specific investigation into "conservative enrichment" to avoid the additional noise caused by such wide searching.

Almost all groups use some type of query expansion. The group from NEC and the Tokyo Institute of Technology ("Ad Hoc Retrieval Experiments Using WordNet and Automatically Constructed Thesauri" by R. Mandala, T. Tokunaga, H. Tanaka, A. Okumura, and K. Satoh) experimented with three different thesauri, including WordNet, a simple co-occurrence-based thesaurus, and a new thesaurus that was automatically built using predicate-argument structures. The University of North Carolina ("IRIS at TREC-7" by K. Yang, K. Maglaughlin, L. Meho, and R.G. Sumner, Jr.) tried two types of relevance feedback approaches and discovered that they performed very differently, particularly when used in the various subcollections of TREC. Fujitsu Laboratories ("Fujitsu Laboratories TREC7 Report" by I. Namba, N. Igata, H. Horai, K. Nitta and K. Matsui) experimented with expansion using the top N documents, and alternatively with expansion using the top M clusters. A final example of work in query expansion is a new method based on relative entropy ("Information term selection for automatic query expansion" by C. Carpineto and G. Romano from Fondazione Ugo Bordoni, Rome and R. De Mori from the University of Avignon).

Although some type of query expansion is clearly necessary for top results, several groups did not use expansion in order to investigate some particular component of retrieval. The SPIDER system ("SPIDER Retrieval System at TREC7" by M. Braschler and M. Wechsler from Eurospider Information Technology and B. Mateev, E. Mittendorf, and P. Schäuble from the Swiss Federal Institute of Technology (ETH)) specifically explored the use of proximity and co-occurrence of terms within the description as an alternative approach to using expansion. NTT DATA ("NTT DATA at TREC-7: system approach for ad-hoc and filtering" by H. Nakajima, T. Takaki, T. Hirao, and A. Kitauchi) also used a new scoring method based on coordination and what they called the "degree of importance" for various co-occurring terms.

Two groups did experiments with various indexing methods. Johns Hopkins University ("Indexing Using Both N-Grams and Words" by J. Mayfield and P. McNamee) worked with both 5-grams and words, including adaptation of the 5-gram method to properly handle stopwords. A joint project led by Tomek Strzalkowski

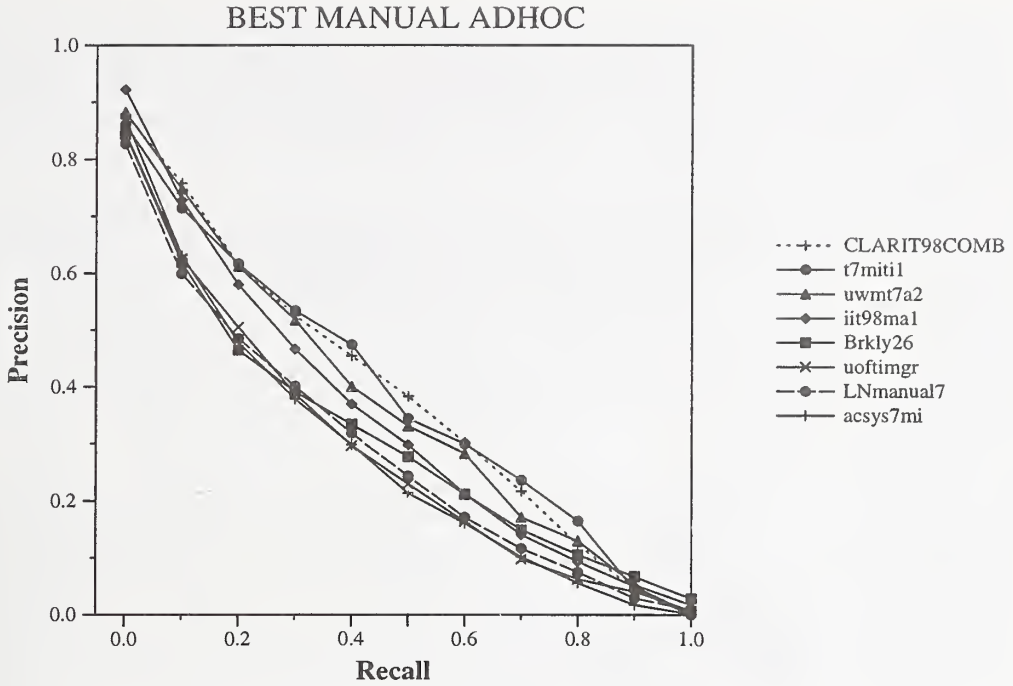


Figure 6: Recall/Precision graph for the top eight manual ad hoc runs.

continued the investigation of merging results from multiple streams of input using different indexing methods ("Natural Language Information Retrieval: TREC-7 Report" by T. Strzalkowski, G. Stein, and G. Bowden Wise from GE Research & Development, J. Perez-Carballo from Rutgers University, P. Tapananinen, T. Jarvinen, and A. Voutilainen from the University of Helsinki and J. Karlgren from the Swedish Institute of Computer Science).

5.2 TREC-6 ad hoc manual results

Figure 6 shows the recall/precision curves for the eight TREC-7 groups with the highest mean average precision scores using manual construction of queries.

CLARIT98COMB – CLARITECH Corp. ("Effectiveness of Clustering in Ad-Hoc Retrieval" by D.A. Evans, A. Huettner, X. Tong, P. Jansen, and J. Bennett) performed a user experiment measuring the difference in performance between presentation modes: a ranked list vs. a clustered set of documents. Starting from a fixed set of initial queries, users spent 30 minutes viewing documents (using the specified presentation mode), marking some as relevant and modifying the queries. The set of "known" relevant documents was used in 2 ways: as input for a Rocchio feedback run to get a 1000-document ranking for each topic and as a fixed set that were then shuffled to the top of each list.

m7miti1 – Management Information Technologies, Inc. ("Readware© Text Analysis and Retrieval in TREC-7" by T. Adi, O.K. Ewell, and P. Adi) used one analyst to manually formulate many queries per topic (an average of 18). These queries used various Readware tools and the user continued to formulate/modify these queries based on information in the retrieved documents. Since this system does no ranking of documents, the final ranked list was composed of those documents judged relevant by the searcher (5898 in all), ordered by the "complexity" of the queries that were used to retrieve the documents.

uwmt7a2 – University of Waterloo ("Deriving Very Short Queries for High Precision and Recall (MultiText Experiments for TREC-7)" by G.V. Cormack, C.R. Palmer, and M. Van Biesbrouck) investigated the building of the ideal very short query. They first spent less than 30 minutes per topic making relevance

judgments, in the method used in TREC-6. These 5529 documents were then used to automatically generate a series of Boolean queries and the query yielding the highest average precision for a given topic was used. The queries were also constrained to having 3 terms or less, with the average number of terms being 1.86. The results of these "ideal" queries were what was submitted to NIST, both in the ad hoc task and in the Very Large Corpus track.

iit98ma1 – ("Use of Query Concepts and Information Extraction to Improve Information Retrieval Effectiveness" by D.O. Holmes from NCR Corporation, D.A. Grossman from U.S. Government, O. Frieder and A. Chowdhury from the Illinois Institute of Technology, and M.C. McCabe from Advanced Analytic Tools) continued work with their parallel database retrieval model. This manual run was done to test the use of phrases and proper nouns. The queries were constructed manually in 15-30 minutes per topic, using the system to retrieve documents to "mine" for good terms. The emphasis was on selecting good phrases and/or proper nouns. The full set of 50 queries consisted of 908 phrases, and 389 single terms. Of these, 541 were proper nouns including 235 names of people.

Brkly26 – University of California at Berkeley ("Manual Queries and Machine Translation in Cross-Language Retrieval and Interactive Retrieval with Cheshire II at TREC-7" by F.C. Gey, H. Jiang, A. Chen and R.R. Larson) explored the possibilities of using manually-constructed Boolean queries to improve performance. For 17 of the 50 topics they constructed these queries; note that for the rest it seemed doubtful that there would be any improvement. For all topics they did a standard automatic logistic regression ranking, but for the 17 Boolean queries they merged the results of the logistic run and the Boolean results. This method improved results for 14 of the 17 topics, 3 very dramatically.

uoftimgr – University of Toronto ("ClickIR: Text Retrieval using a Dynamic Hypertext Interface" by R.C. Bodner and M.H. Chignell) used their dynamic hypertext model to build the queries. Users took approximately 15 minutes to browse the collection, clicking on sentences that then linked to one or two new documents. These links were created automatically from terms used in previous queries from the same user. The final query that created the results sent to NIST was then assembled from these "clicked on" sentences (which were logged). The INQUERY system was used for production of the ranked list, and also to make an additional run in which "known" relevant documents were used for relevance feedback.

LNmanual7 – Lexis-Nexis ("Experiments in Query Processing at LEXIS-NEXIS for TREC-7" by A.G. Rao, T. Humphrey, A. Parhizgar, C. Wilson, and D. Pliske) experimented with human relevance feedback as opposed to automatic feedback from the top 20 documents. The users read the top 20 documents and then modified the initial automatic query, adding terms and phrases, but only if those phrases and terms were in the index. The users also had to supply the new term weighting. This hand-picking of the expansion terms improved performance by 28% over the baseline shown in the paper and 14% over the best automatic Lexis-Nexis run, with most improvements in the higher-precision areas of the graph.

acsys7mi – CSIRO, Australian National University ("ACSys TREC-7 Experiments" by D. Hawking from CSIRO Mathematics and Information Sciences and N. Craswell and P. Thistlewaite from the Australian National University) explored both manual editing of the query before feedback, additional editing of the query after viewing documents, and the use of concept scoring to better balance terms representing different concepts. The new Quokka GUI system displayed the concepts in various colors to aid in this task, and a median of 10.6 minutes was taken per topic to produce the final query used in ranking the documents. A major system error affected the results and corrected tables can be seen in the paper.

6 The Tracks

One of the goals of TREC is to provide a common task evaluation that allows cross-system comparisons, which has proven to be a key strength in TREC. A second major strength is the loose definition of the ad hoc task, which allows a wide range of experiments. The addition of secondary tasks (called tracks) in TREC-4 combined these strengths by creating a common evaluation for retrieval subproblems.

Table 10: Number of track participants.

	TREC-6	TREC-7
CLIR	13	9
filtering	10	12
HP	5	4
interactive	9	8
query	0	2
SDR	13	10
VLC	7	6

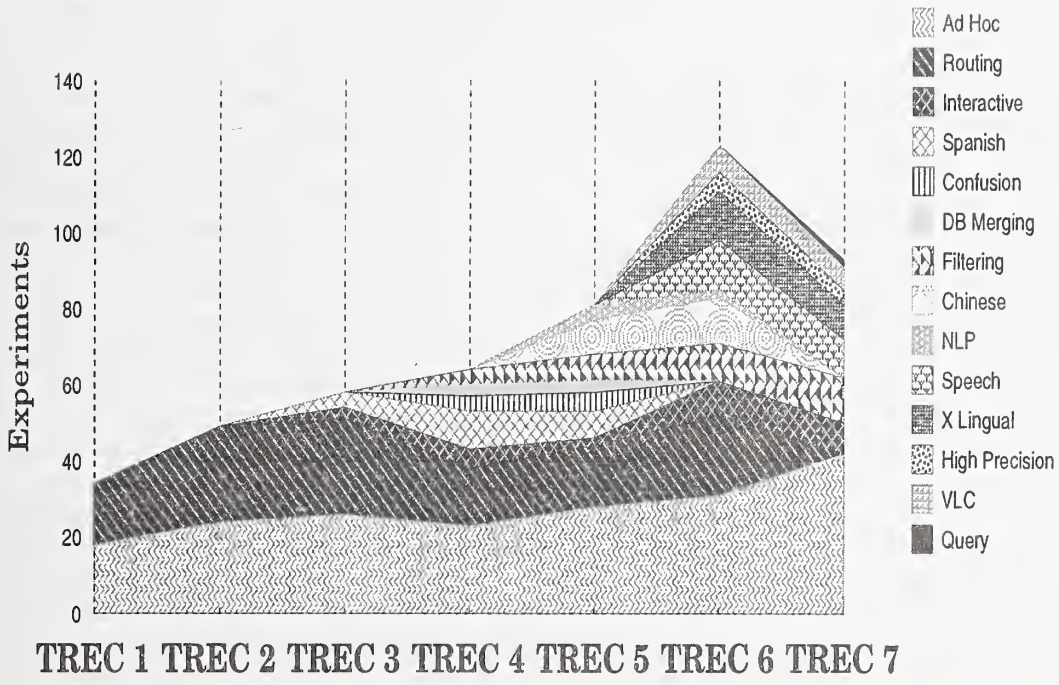


Figure 7: Number of TREC experiments by TREC task.

The tracks have had a significant impact on TREC participation. Figure 7 shows the number of experiments performed in each TREC, where the set of runs submitted for a track by one group is counted as one experiment. The number of experiments increased each year through TREC-6 then decreased in TREC-7, mostly due to the elimination of the routing main task and the Chinese track. The number of participants performing the ad hoc task continues to grow, with 42 groups taking part in TREC-7 compared to 31 in TREC-6. Table 10 gives the number of participants in each of the TREC-7 tracks for both TREC-6 and TREC-7.

The main ad hoc task provides an entry point for new participants and provides a baseline of retrieval performance. The tracks invigorate TREC by focusing research on new areas or particular aspects of text retrieval. To the extent the same retrieval techniques are used for the different tasks, the tracks also validate the findings of the ad hoc task.

Each track has a set of guidelines developed under the direction of the track coordinator. Participants are free to choose which, if any, of the tracks they will join. This section describes the TREC-7 tracks. The overall goals, the experimental design used, and a very brief summary of the results are listed for each track. See the track reports elsewhere in this proceedings for a more complete description of each track.

6.1 The Cross Language (CLIR) track

The CLIR task focuses on searching for documents in one language using topics in a different language. The first CLIR track was held in TREC-6 [6]. Three document sets were used in TREC-6: a set of French documents from the Swiss news agency *Schweizerische Depeschen Agentur* (SDA); a set of German documents from SDA plus a set of articles from the newspaper *New Zurich Newspaper* (NZZ); and a set of English documents from the AP newswire. All of the document sets contain news stories from approximately the same time period, but are not aligned or specially coordinated with one another. A set of 25 topics that were translated into each of the languages was also provided. Participants searched for documents in one target language using topics written in a different language.

The TREC-7 task expanded on this beginning. The document set for the TREC-7 track consisted of all the documents used in the TREC-6 track plus the Italian version of the SDA for the same time period. Participants were provided with a new set of 28 topics (with translations available in English, French, German, and Italian), and used one topic language to search the combined document set. That is, a single run retrieved documents written in different languages. Since some participants were not able to process all four languages, a second task in which English topics were run against the combined French and English document set was also run.

The TREC-7 track also defined an optional subtask. The subtask used a different document collection, a 31,000 document structured database (formatted as SGML fielded text data) from the field of social science plus the NZZ articles, and a separate set of 28 topics. The rationale of the subtask was to study CLIR in a vertical domain (i.e., social science) where a German/English thesaurus is available.

Nine groups participated in the TREC-7 CLIR track, with five groups performing the test on the full four language collection, and seven groups performing the test on the English and French collection. No runs were submitted for the optional subtask, however this subtask is planned to be repeated in TREC-8 now that groups have more experience with cross language retrieval. The results of the track demonstrate that very different approaches to cross-language retrieval can lead to comparable retrieval effectiveness.

The construction of the cross language test collection differed from the way any other TREC collection has been created. Candidate topics in the native language were created in each of four different institutions: NIST (English); EPFL Lausanne, Switzerland (French); Informationszentrum Sozialwissenschaften, Bonn, Germany (German); and CNR, Pisa, Italy (Italian). Each institution created topics that would target documents in its corresponding language. NIST selected the final set of 28 topics, balancing the set so that each institution contributed seven topics. Each of the final topics was then translated into the three remaining languages so that the entire set of topics was available in each language. The relevance judgments for all topics for a particular document language were made at the site responsible for that language. This is the first time that TREC has used multiple relevance assessors for a single topic.

For the complete overview of the track see "Cross-Language Information Retrieval (CLIR) Track Overview" by M. Braschler, J. Krause, C. Peters, and P. Schäuble.

6.2 The Filtering track

Each of the previous TRECs have had a second main task called the routing task. A routing task investigates the performance of systems that use standing queries to search new streams of documents, as news clipping services and library profiling systems do, for example. As the routing task has been defined in TREC, participants use old topics with existing relevance judgments to form routing queries, and use those queries to rank a previously unseen document collection. Real routing applications generally require a system to make a binary decision whether or not to retrieve the current document, however, not simply form a ranking of a document set. The filtering track was started in TREC-4 to address this more difficult version of the routing task.

The TREC-7 filtering track contained three tasks of increasing difficulty (and realism). For each task, topics 1–50 and the AP newswire collection on Disks 1–3 were used (with different splits into training and test sets, depending on the task). The first task was the traditional routing task. The second task was a *batch* filtering task in which systems are given topics and relevance judgments as in the routing task, and must then decide whether or not to retrieve each document in the test portion of the collection. This task

is what previous filtering tracks performed. The third task, and the focus of the track, was an *adaptive* filtering task. In this task, a filtering system starts with just the query derived from the topic statement, and processes documents one at a time in date order. If the system decides to retrieve a document, it obtains the relevance judgment for it, and can modify its query as desired.

Because filtering tasks return an unordered set of documents, not a ranking, different evaluation measures from those used for ad hoc or routing are required. Developing appropriate measures for filtering systems continues to be an important part of the track. The main approach used in TREC is to use utility functions as measures of the quality of the retrieved set—the quality is computed as a function of the benefit of retrieving a relevant document and the cost of retrieving an irrelevant document [5]. In TREC-7 two different utility functions were used:

$$\begin{aligned} \text{F1} &= 3\text{R}^+ - 2\text{N}^+ \\ \text{F3} &= 4\text{R}^+ - \text{N}^+ \end{aligned}$$

where R^+ and N^+ are the number of relevant and non-relevant documents retrieved, respectively. A problem with utilities as measures is that different topics have widely varying possible utility values, making it difficult to meaningfully compare scores across topics or compute an average score. An approach to scaling and normalizing utilities was introduced in this year's track [3].

Twelve groups submitted at least one TREC-7 filtering run. A total of 46 runs was submitted, consisting of 10 routing runs, 12 batch filtering runs, and 24 adaptive filtering runs. The track results demonstrated that adaptive filtering is a challenging problem for current systems. Indeed, when using the F1 utility measure to evaluate performance, the "baseline" system which retrieves no documents was the most effective system overall. Comparison with batch filtering results show that setting an appropriate threshold for when to retrieve a document is a critical, and difficult, task in adaptive filtering.

For the complete overview of the track see "The TREC-7 Filtering Track: Description and Analysis" by D.A. Hull.

6.3 The High Precision track

The task in the high precision track was to retrieve fifteen relevant documents for a topic within five minutes (wall clock time). Users could not collaborate on a single topic, nor could the system (or user) have previous knowledge of the topic. Otherwise, the user was free to use any available resources as long as the five-minute time limit was observed. The task is an abstraction of a common retrieval problem: quickly find a few good documents to get a feel for the subject area.

Since the track guidelines put no limits on who the user could be, an implicit assumption of the track is that the runs are performed by system experts. As such, the track provides an upper-bound on the effectiveness obtainable by the systems. The 5-minute time limit was selected so that the intrinsic effectiveness of the system, the system efficiency, and the user interface would all be tested by the task. The same 50 topics and document set as used in the TREC-7 ad hoc task were used for the HP track.

Four groups participated in the TREC-7 track, submitting a total of seven runs. One finding of the track was that retrieving 15 good documents is a simple enough task for current retrieval systems that disagreements between the searcher and the assessor regarding what constitutes a relevant document bounds performance. However, new time-based evaluation measures introduced in the track offer a possible solution.

For the complete overview of the track see "The TREC-7 High Precision Track" by C. Buckley.

6.4 The Interactive track

The interactive track is another track that was started in TREC-4. The high-level goal of the track is the investigation of searching as an interactive task by examining the process as well as the outcome. One of the main problems with studying interactive behavior of retrieval systems is that both searchers and topics generally have a much larger effect on search results than does the retrieval system used. The TREC-7 track used an experimental framework designed to provide an estimate of the difference between an experimental and a control system that is uncontaminated by the differences between searchers and topics.

The experimental framework both defined a common task for participants to perform and prescribed an experimental matrix. The search task used the title and description sections plus a special "Instances" section of eight ad hoc topics; the documents searched were the *Financial Times* collection from Disk 4. The topics each described a need for information of a particular type such that multiple distinct examples or instances of that information were contained in the document collection. The searchers job was to save documents covering as many distinct answers to the question as possible in a 15-minute time limit. The NIST assessor for the topic made a comprehensive list of instances from the documents submitted by the track. The effectiveness of the search was evaluated by the fraction of total instances for that topic covered by the search (instance recall) and the fraction of the documents retrieved in the search that contained an instance (instance precision). Participants were also required to collect demographic and psychometric data from the searchers, and to report extensive data on each searcher's interactions with the search systems.

The experimental matrix defined how searchers and topics were to be divided among the experimental and control systems. (Participants were free to choose whatever systems they wanted to serve as experimental and control. That is, the track did not attempt to coordinate cross-site comparisons or test particular hypotheses.) The matrix was based on a latin square design, which provides the desired uncontaminated estimate of the difference between the systems. The minimum experiment defined by the design required eight searchers, with each searcher performing four searches with each of the two systems. The eight-searcher minimum was imposed since the results of the TREC-6 track suggested that with eight topics at least eight searchers are required to obtain statistically significant results [4].

Eight groups participated in the interactive track, performing a total of ten experiments. Since comparison of systems across sites was not supported by the experimental design, the results of the track need to be understood in the context of the particular research goals of the individual research groups.

For the complete overview of the tracks see "TREC-7 Interactive Track Report" by P. Over.

6.5 The Query track

The query track was a new track whose goal was to create a large query collection. The variability in topic performance (e.g., see the discussion of the effect of topic length on performance in Section 5.1) makes it impossible to reach meaningful conclusions regarding query-dependent processing strategies unless there is a very large query set—much larger than the sets of 50 topics used in the TREC collections. The query track was designed as a means for creating a large set of different queries for an existing TREC topic set, topics 1–50.

Participants in the track created different types of queries from the topic statements and/or relevance judgments. A query of a given type was created for each of the 50 topics, forming one query set. Five different query types were used:

Very short: two or three words extracted from the topic statement.

Sentence: an English sentence based on the topic statement and the relevant documents.

Manual feedback: an English sentence based on reading 5–10 relevant documents only (by someone who doesn't know the topic statement).

Manual structured query: a manually constructed query based on the topic statement and relevant documents. The use of operators supported by the participant's system was encouraged. The TIPSTER DN2 format was used to represent the query structure.

Automatic structured query: a query constructed automatically from the topic statement and relevance judgments. TIPSTER DN2 format used to represent the query structure.

Participants exchanged the query sets they created with all other participants in the track, and all participants ran all query sets their system could support. The document set used for the runs was the documents on Disk 2 plus the AP collection on Disk 3. The retrieval results were submitted to NIST where all runs were judged and evaluated.

Since the track design included all groups running all query sets, a number of direct comparisons are possible. First, participants can see how effective their system is using their own queries. Second, they can

see how effective their search component is when using other queries. Finally, participants can evaluate how effective their query construction strategies are by seeing how other groups fared with their queries.

Unfortunately, only two groups participated in the query track, too few to make any meaningful comparisons. The track will run again in TREC-8, with the hope that heightened awareness of the problems the query track is addressing will generate participation.

For the complete overview of the track see "The TREC-7 Query Track" by C. Buckley.

6.6 The Spoken Document Retrieval (SDR) track

The SDR track fosters research on retrieval methodologies for spoken documents (i.e., recordings of speech). The track, which began in TREC-6, is a successor to the "confusion tracks" of earlier TREC conferences, which investigated methods for retrieving document surrogates whose true content has been confused or corrupted in some way. In the SDR track, the document surrogates are produced by speech recognition systems.

As the earlier confusion tracks had used, the TREC-6 SDR track used a known-item search task. The TREC-7 track implemented a full ranked retrieval task. The document collection consisted of transcripts of approximately 100 hours of broadcast news programs, representing about 3000 news stories. Participants worked with four different versions of the transcripts: the *reference* transcripts, which were hand-produced and assumed to be perfect; the first *baseline* transcripts, which were produced by a baseline speech recognition system running at about 35% word error rate; a second set of baseline transcripts, produced by the baseline recognizer running at about 50% word error rate; and the *recognizer* transcripts, which were produced by the participant's own recognizer system. Document boundaries were given in the hand-produced transcripts, and the same boundaries were used in the other versions.

NIST created a set of 23 topics, which were used to search each of the versions of the transcripts. The different versions of the transcripts allowed participants to observe the effect of recognizer errors on their retrieval strategy. The different recognizer runs provide a comparison of how different recognition strategies affect retrieval. To make this comparison as complete as possible, participants were encouraged to retrieve using other groups' recognizer transcripts as well. These runs are called *cross-recognizer* runs.

Eleven groups participated in the TREC-8 SDR track. The results of this year's track display a linear correlation between the error rate of the recognition and a decrease in retrieval effectiveness, a correlation that was not present in last year's track that used a known-item search task. Not surprisingly, the correlation is stronger when recognizer error rate is computed over content-based words (e.g., named entities) rather than *all* words.

For the complete overview of the track see "1998 TREC-7 Spoken Document Retrieval Track Overview and Results" by J. Garofolo, E.M. Voorhees, C.G.P. Auzanne, V.M. Stanford, and B.A. Lund.

6.7 The Very Large Corpus (VLC) track

The VLC track explores how well retrieval algorithms scale to larger document collections. In contrast to the ad hoc task that uses a 2 GB document collection, the first running of the VLC track in TREC-6 used a 20 GB collection, while the TREC-7 track used a 100 GB document collection. The TREC-7 collection consisted of World Wide Web data that was collected by the Internet Archive (`http://www.archive.org`). The track used the TREC-7 ad hoc topics, and a set of relevance judgments produced by assessors at the Australian National University. Because of the difficulty of getting sufficient relevance judgments to accurately measure recall, the main effectiveness measure used for VLC runs was precision after 20 documents were retrieved.

To more accurately measure the effect size has on the retrieval systems used by the participants, the track provided 3 collections: the original 100 GB collections plus 1% and 10% subsamples. Participants indexed each of the three collections and ran the entire topic set on each. They then reported timing figures for each phase as well as the top 20 retrieved. The main evaluation measures were precision after 20 documents retrieved (the effectiveness measure); query response time (elapsed time as seen by the user); data structure (e.g., inverted index) building time (elapsed time as seen by the user); plus a combination timing measure that factored in the expense of the hardware used.

Seven groups participated in the VLC track, with six groups processing the entire 100 GB corpus. The track demonstrated that processing a 100 GB corpus is well within the capabilities of today's retrieval systems. Of particular note was the Multitext group that achieved sub-second query processing time while maintaining good retrieval effectiveness using hardware that cost under US$100,000.

For the complete overview of the track see "Overview of TREC-7 Very Large Collection Track" by D. Hawking, N. Craswell, and P. Thistlewaite.

7 The Future

The final session of each TREC workshop is a planning session for future TRECs—especially to decide on the set of tracks for the next TREC. Two new tracks are planned for TREC-8, the question answering track and the Web track. The question answering track is designed to encourage research on methods for *information* retrieval as opposed to document retrieval. The goal in the track will be for systems to produce short text extracts that contain the answer for each of a set of 200 questions. The goal in the Web track will be to investigate whether links can be used to enhance retrieval. The track will use a 2 GB subset of the data collected for the VLC track and a typical TREC ad hoc task. Also, participation in the query track is encouraged, since the benefits of that track increase with increased participation.

Acknowledgments

The authors gratefully acknowledge the continued support of the TREC conferences by the Intelligent Systems Office of the Defense Advanced Research Projects Agency. Thanks also go to the TREC program committee and the staff at NIST. The TREC tracks could not happen without the efforts of the track coordinators; our special thanks to them.

References

[1] D. K. Harman, editor. *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, October 1996. NIST Special Publication 500-236.

[2] Donna Harman. Analysis of data from the second Text REtrieval Conference (TREC-2). In *Proceedings of RIAO94*, pages 699–709, 1994.

[3] David A. Hull. The TREC-7 filtering track: Description and analysis. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, August 1999. NIST Special Publication 500-242.

[4] Eric Lagergren and Paul Over. Comparing interactive information retrieval systems across sites: The TREC-6 interactive track matrix experiment. 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 164–172, Melbourne, Australia, August 1998. ACM Press, New York.

[5] David D. Lewis. The TREC-4 filtering track. In Harman [1], pages 165–180. NIST Special Publication 500-236.

[6] Peter Schäuble and Páraic Sheridan. Cross-lanugage information retrieval (CLIR) track overview. In Voorhees and Harman [12], pages 31–43. NIST Special Publication 500-240.

[7] K. Sparck Jones. Reflections on TREC. *Information Processing and Management*, 31(3):291–314, 1995.

[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] Jean Tague-Sutcliffe and James Blustein. A statistical analysis of the TREC-3 data. In D. K. Harman, editor, *Overview of the Third Text REtrieval Conference (TREC-3) [Proceedings of TREC-3.]*, pages 385–398, April 1995. NIST Special Publication 500-225.

[10] Ellen M. Voorhees and Donna Harman. Overview of the fifth Text REtrieval Conference (TREC-5). In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, pages 1–28, November 1997. NIST Special Publication 500-238.

[11] Ellen M. Voorhees and Donna Harman. Overview of the sixth Text REtrieval Conference (TREC-6). In Voorhees and Harman [12], pages 1–24. NIST Special Publication 500-240.

[12] E.M. Voorhees and D.K. Harman, editors. *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*, August 1998. NIST Special Publication 500-240.

Cross-Language Information Retrieval (CLIR) Track Overview

Martin Braschler[1], Jürgen Krause[2], Carol Peters[3], Peter Schäuble[4]

[1] Eurospider Information Tech. AG, Schaffhauserstr. 18, CH-8006 Zürich, Switzerland
[2] Informationszentrum Sozialwissenschaften, Lennéstr. 30, D-53113 Bonn, Germany
[3] Istituto di Elaborazione della Informazione (IEI-CNR), Via S. Maria 46, Pisa, Italy
[4] Swiss Federal Institute of Technology (ETH), CH-8092 Zürich, Switzerland

1 Introduction

This year, the TREC cross-language retrieval track took place for the second time. In TREC-7, we extended the task presented to the participants. The goal was for groups to use queries written in a single language in order to retrieve documents from a multilingual pool of documents written in many different languages. This is also a more comprehensive task than the usual definition of cross-language information retrieval, where systems work with a language pair, retrieving documents in a language L1 using queries in language L2.

The document languages used this year were English, German, French, and, newly introduced for TREC-7, Italian. The queries were available in all of these languages. Because it seemed unlikely that all interested parties can work with all four languages, it was agreed that there would be a secondary evaluation involving a smaller task. Consequently, groups were allowed to send in runs using the English queries to retrieve documents from a subset of the pool containing just the English and French documents. Coordination of the track took place at ETH in Zurich, as for last year.

The continued interest in the cross-language track showed the importance of this emerging area. There are many applications where information should be accessible to users regardless of its language. With the ever growing amount of information available to us all, situations when a user of an information retrieval system is faced with the task of querying a multilingual document collection are becoming increasingly common. Such collections can be made up of documents from multinational companies, from multilingual countries or from large international organizations such as the United Nations or the European Commission. Of course, the world wide web is also an example for such a document collection.

A lot of users of such multilingual data sources have some foreign language knowledge, but their proficiency may not be good enough to formulate queries to appropriately express their information need. Such users will benefit greatly if they can enter queries in their native language, because they will be able to inspect the documents even if they are untranslated. Monolingual users, on the other hand, can use translation aids, manual or automatic, to help them access the search results.

2 Overview of CLIR

Approaches to cross-language information retrieval can be categorized according to how they solve the problem of matching the query and documents across different languages – how they "*cross the language barrier*". This may be achieved by using query translation, document

translation, or by using both query and document translation (see also Figure 1). One possibility would be to translate queries and documents into a controlled, language independent indexing vocabulary. TextWise (Diekema et al., 1999) uses WordNet synsets as such a vocabulary. More common in the framework of TREC are free text approaches. These methods can be further classified according to what resources are used to cross the language boundary: machine translation, machine-readable dictionaries, or corpus-based resources.

Machine translation (MT) seems an obvious choice for cross-language information retrieval systems. Groups that experimented with machine translation for CLIR include NEC in Japan (Yamabana et al., 1998). They used MT to translate users' queries in an interactive process involving both dictionaries and statistical information derived from bilingual corpora. Systran is also reporting work addressing cross-language retrieval (Gachot et al., 1998). In the context of this year's track, the Berkeley group carried out experiments with three different off-the-shelf MT systems (Gey et al., 1999). Note that CLIR is a difficult problem to solve based on MT alone: queries that users typically enter into a retrieval system are rarely complete sentences and provide little context for sense disambiguation.

Corpus-based approaches to CLIR include, among others, the use of Latent Semantic Indexing (LSI) by researchers at Bellcore and elsewhere (Littman et al., 1998), the Generalized Vector Space model proposed by CMU (Carbonell et al., 1997) and work from ETH/Eurospider using Similarity Thesauri and Pseudo-relevance feedback (Braschler and Schäuble, 1998). All these approaches use corpus resources as training data to adapt the CLIR mechanism or build information structures used for subsequent retrieval.

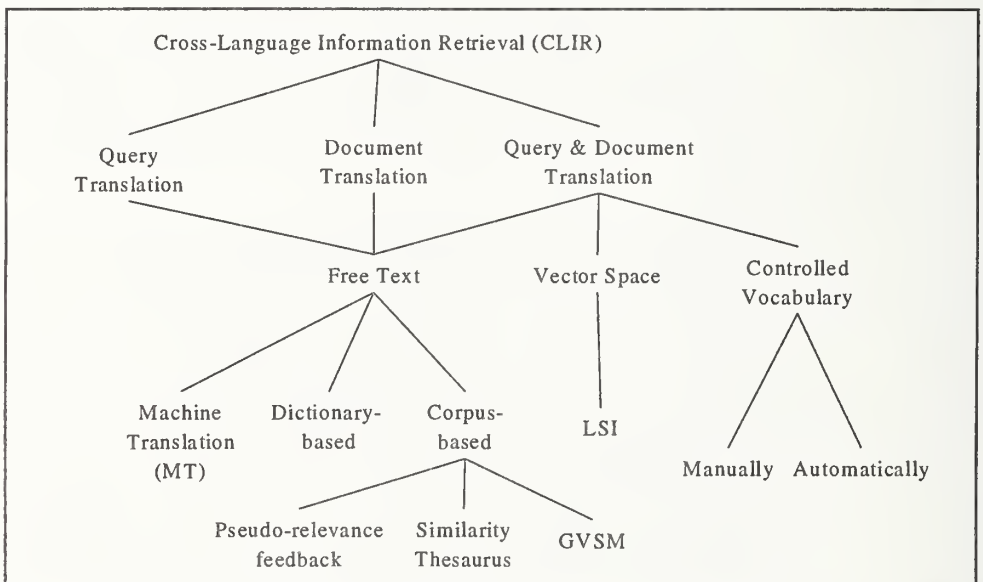


Figure 1: Approaches to cross-language information retrieval

Another natural approach to cross-language retrieval is the use of existing linguistic resources, mainly machine-readable bilingual dictionaries. Among groups that looked into the use of such dictionaries are researchers from Xerox Research Centre Europe (Hull and Grefenstette, 1996), the University of Massachusetts (Ballesteros and Croft, 1996) and the Computing Research Laboratory at the New Mexico State University (NMSU) (Davis and

Ogden, 1997). Various ideas have been proposed to address some of the problems associated with dictionary-based translations, such as ambiguities and vocabulary coverage.

Approaches from all three main classes have been used by the participants in the TREC-7 CLIR track.

3 CLIR-Track Task Description

This year, the participants were asked to retrieve documents from a multilingual pool containing documents in four different languages. They were able to chose the topic language, and then had to find relevant documents in the pool regardless of the languages the texts were formulated in. As a side effect, this meant that most groups had to solve the additional task of merging results from various bilingual runs.

The languages present in the pool were English, German, French and Italian, with Italian being a new language introduced for TREC-7. The 28 topics were distributed in each of these four languages. To allow for participation of groups that do not have the resources to work in all four languages, a secondary evaluation was provided that permitted such groups to send in runs using English queries to retrieve documents from a subset of the pool just containing texts in English and French. This year, we did not have monolingual runs as part of the cross-language track.

The TREC-7 task description also defined a subtask, working with a second data collection, containing documents from a structured data base from the field of social science. Unfortunately, the introduction of this data was probably premature, since no groups were working with this data this year. The data will however be used again for next year's CLIR track.

The document collection for the main task contained the same documents as used for TREC-6. The English texts were taken from the Associated Press, covering three years (1988 to 1990) worth of news stories. For German and French, news stories were taken from SDA, the "Schweizerische Depeschenagentur" (Swiss News Agency). They were chosen to cover the same time period. While the texts for German and French were produced by the same company, this does not mean that they contain translations. However, there is a sizeable topic overlap between the texts in these two languages. For German, additionally texts from the Swiss news paper "Neue Zürcher Zeitung" (NZZ) were used. We had one year of articles (from 1994) available to participants. As an extension to this document collection, Italian texts from SDA were introduced in TREC-7. Again, while produced by the same company as the French and some of the German texts, they are not direct translations from either of these languages. We had texts from 1989 and 1990 available in Italian. Table 1 gives more details of the document collections.

There have been significant changes in the way the topics were created for this year. The experience of the first CLIR track showed that it is difficult to produce topics in all languages in a single place. Therefore, a distributed approach to topic creation was chosen. We had four different sites, each located in an area where one of the topic languages is natively spoken.

Document collections			
Doc. Language	**Source**	**No. Documents**	**Size**
English	AP news, 1988-90	242,918	750 MB
German	SDA news, 1988-90	185,099	330 MB
	NZZ articles, 1994	66,741	200 MB
French	SDA news, 1988-90	141,656	250 MB
Italian	SDA news, 1989-90	62,359	90 MB

Table 1: details for the document collections.

The topic creation sites were:

- English: NIST, Gaithersburg, MD, USA (Ellen Voorhees)
- French: EPFL Lausanne, Switzerland (Afzal Ballim)
- German: IZ Sozialwissenschaften, Germany (Jürgen Krause, Michael Kluck)
- Italian: CNR, Pisa, Italy (Carol Peters).

From each site seven topics were chosen to be included in the topic set. The other 21 queries were then translated. This ultimately led to a pool of 28 topics, each available in all four languages.

Participants were free to experiment with different topic fields, and with both automatic and manual runs, similar to the definitions of the TREC adhoc task.

4 Results

A total of nine groups from five different countries submitted results for the TREC-7 CLIR track (see Table 2). Because of the different task this year, the number of runs per group could be reduced, since the number of language combinations is much smaller with the document pool being fixed. The participants submitted 27 runs, 17 for the main task, and 10 for the secondary evaluation. Five groups (Berkeley, Eurospider, IBM, Twenty-One and Maryland) tackled the main task. English was, not surprisingly, the most popular topic language, with German coming in a strong second. Every language was used by at least one group.

Participant	**Country**
CEA (Comissariat à l'Energie Atomique)	France
Eurospider Information Technology AG	Switzerland
IBM T.J. Watson Research Center	USA
Los Alamos National Laboratory	USA
TextWise LLC	USA
Twenty-One (University of Twente/TNO-TPD)	Netherlands
University of California at Berkeley	USA
University of Maryland	USA
Université de Montréal	Canada

Table 2: Table of participants.

The relevance assessments used for evaluation of these runs were produced at the same four sites that were used for topic creation.

Remarkably, average precision numbers are generally much higher than last year. While it is hard to compare absolute levels across topic sets, this would be an indication that the level of the results has improved this year. Unfortunately, there is little mention by participants about experiments on last year's topics with the current systems that could substantiate such an assumption. Also, the Twenty-One group (Hiemstra et al., 1999) makes a case as to why absolute levels may be too high ("flattering") this year.

Figure 2 shows a comparison of runs for the main task. Shown are the best automatic runs against the full document pool for each of the five groups that solved the full task. As can be seen, most participants performed in a fairly narrow band. This is interesting given the very different approaches of the individual participants: IBM used translation models automatically trained on parallel and comparable corpora (McCarley, 1999). Twenty-One used sophisticated dictionary lookup and a "boolean-flavoured" weighting scheme (Hiemstra et al., 1999). Eurospider employed corpus-based techniques, using similarity thesauri and pseudo-relevance feedback on aligned documents (Braschler et al., 1999). The Berkeley (Gey et al., 1999) and Maryland groups used off-the-shelf machine translation systems.

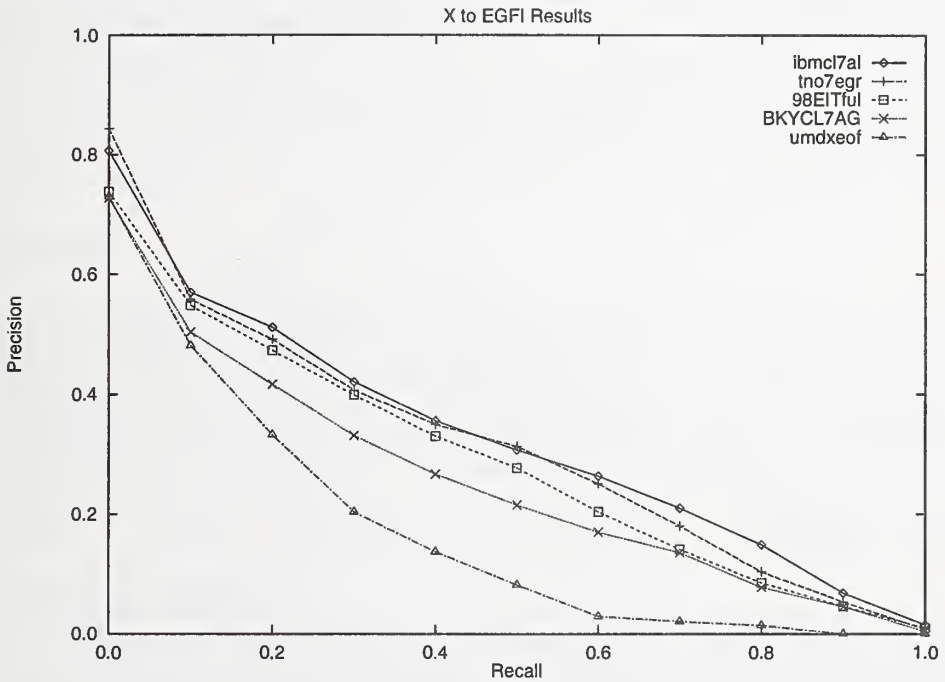


Figure 2: Results of the main evaluation X→EGFI.

Figure 3 shows results from the secondary evaluation. Again, the graph shows the best automatic run submitted by each participant. The run of the Los Alamos group is an exception, as it is classified as a manual run. Here too, the top three runs are quite close. IBM was again using their statistical translation models, Twenty-One was using dictionary-based translation with fuzzy query expansion terms and Berkeley was again using their MT approach. Following are four groups only participating in the secondary evaluation: Université de Montréal, CEA, TextWise and Los Alamos.

A particularly interesting aspect of this year's track was how participants approached the merging problem. Again, many interesting methods were used: Among the solutions proposed were: Twenty-One compared averages of similarity values of individual runs, Eurospider used

document alignments to map runs to comparable score ranges through linear regression and IBM used modeling of system-wide probabilities of relevance. But it was also possible to avoid the merging problem: the Berkeley group expanded the topics to all languages and then ran them against an index containing documents from all languages, therefore directly retrieving a multilingual result list.

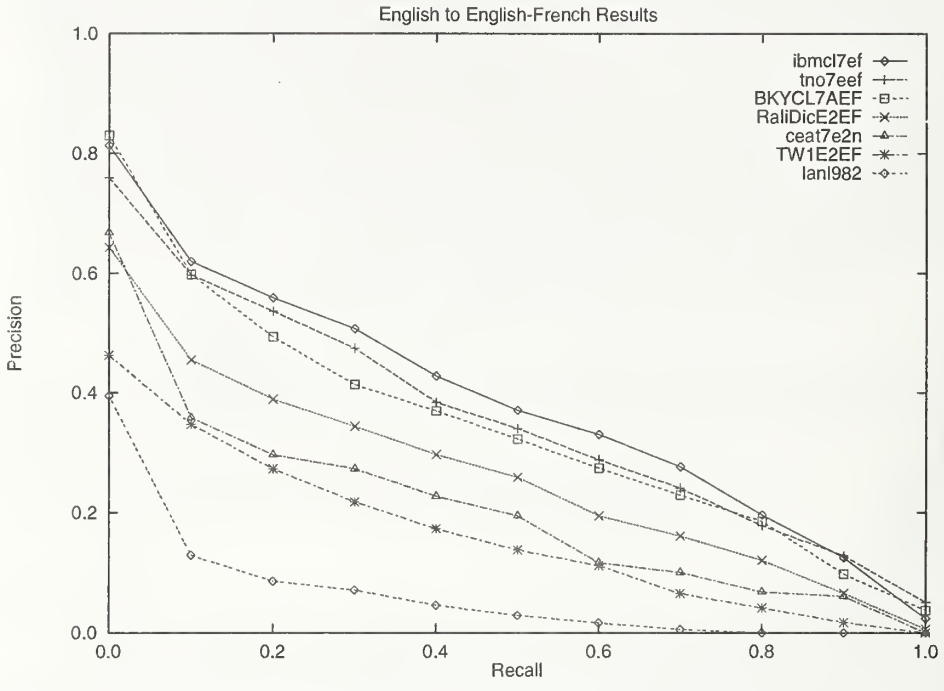


Figure 3: Results of the secondary evaluation E→EF.

5 Evaluation issues

As mentioned, one of the distinguishing features of the CLIR track is that topic development is distributed. Topic development is clearly subjective, and tends to depend on the creator's own particular background. However, for CLIR it is presumed that both the language and cultural background also impact on the choice and phrasing of topics. A close examination of this year's topics would probably permit an astute observer to group them fairly accurately, according to source language and creation site. This should not be considered negative in the participants' viewpoint nor should it affect the validity of the exercise. However, it causes some problems both with reference to translation of the topics and their assessment.

Since it is unrealistic to find topic creators that have total competence in all four languages, each topic is developed in one language and then translated at the other sites. Topic translation thus raises the typical problems involved in any translation: a total understanding of the source in order to achieve a perfect rendering of the target. The conflict is as to how far the target version can deviate from the source in terms of style, vocabulary, and authenticity. It is necessary to find an acceptable balance between precision with respect to the source and naturalness with respect to the target language. While preserving the topic meaning, terms must be used in the target topic that are actually found in the documents of that language. Thus, a

high level of performance is required of the topic translators to avoid an imbalance in topic authenticity.

The relevance assessments were also produced in the same distributed setting. Of course, an accurate assessment of relevance for retrieved documents for a given topic implies a good understanding of the topic. In the distributed scenario of the CLIR track, understanding is also influenced by the multilingual/multicultural characteristics of the task. Although the topic creators initially worked on the basis of their knowledge of possible events for the years covered by the document collections, the final decision and refinement with regard to the topics was made based on the contents of the document collection. The way a particular argument is presented in a collection therefore will influence its formulation. However, this presentation is not necessarily reproduced in the documents in other languages. Thus a topic which did not appear to raise problems of interpretation in the language used for its preparation, may be much more difficult to assess against documents in another language. Some of the topics were found by the assessors to be too vague or difficult to interpret, while others required very specific knowledge. The decision for each local topic developer to include one or two topics of high local significance also caused some difficulties. Some more political arguments, that were well known and much discussed in the local document collection but still had wider implications were difficult to understand and recognize in other collections.

Obviously, it tends to be easier to assess the results for a collection in the original language used for a topic. This fact needs further investigation to assess its real effect (if any) on the overall results.

6 Outlook

The CLIR track will return next year. It was agreed to keep the main task; retrieving documents in many different languages. There also will be a secondary evaluation, retrieving documents from a pool of English documents and one additional chosen language. The GIRT subtask will be offered again next year, and will also allow to send in monolingual runs, something that is not planned for the other evaluations.

The evaluation issues mentioned above mean that there will be emphasis on clear rules for translation, and topics will be circulated to check for problems of interpretation. "Difficult" topics will be possibly accompanied by interpretation aids or training of assessors.

Acknowledgements

We thank the Neue Zürcher Zeitung (NZZ), the Schweizerische Depeschenagentur (SDA) and the Asssociated Press (AP) for making their data available to the TREC community. We would also like to express our gratitude to everyone involved in topic creation and relevance assessment at NIST, the IZ Sozialwissenschaften, CNR and the EPFL.

References

Ballesteros, L. and Croft, W. B. (1996). Dictionary-based Methods for Crosslingual Information Retrieval. In *Proceedings of the 7th International DEXA Conference on Database and Expert Systems Applications.*

Braschler, M. and Schäuble, P. (1998). Multilingual Information Retrieval Based on Document Alignment Techniques. In *Second European Conference on Research and Advanced Technology for Digital Libraries, Heraklion, Greece.*

Braschler, M., Mateev, B., Mittendorf, E., Schäuble, P., and Wechsler, M. (1999). SPIDER Retrieval System at TREC7. In *Proceedings of the Seventh Text Retrieval Conference (TREC7), National Institute of Standards and Technology (NIST), Gaithersburg, MD.*

Carbonell, J., Yang, Y., Frederking, R., Brown, R. D., Geng, Y., and Lee D. (1997). Translingual information retrieval: A comparative evaluation. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence.*

Davis, M. and Ogden, W. (1997). QUILT: Implementing a Large-Scale Cross-Language Text Retrieval System. In *Proceedings of the 20th ACM SIGIR Conference on Research and Development in Information Retrieval, Philadelphia, PA.*

Diekema, A., Oroumchian, F., Sheridan, P., and Liddy, E. D. (1999). TREC-7 Evaluation of Conceptual Interlingua Document Retrieval (CINDOR) in English and French. In *Proceedings of the Seventh Text Retrieval Conference (TREC7), National Institute of Standards and Technology (NIST), Gaithersburg, MD.*

Gachot, D. A., Lange, E., and Yang, J. (1998). The SYSTRAN NLP browser: An application of machine translation technology in multilingual information retrieval. In Grefenstette, G., editor, *Cross-Language Information Retrieval*, chapter 9. Kluwer Academic Publishers, Boston.

Gey, F. C., Jiang, H., and Chen, A. (1999). Manual queries and Machine Translation in Cross-Language Retrieval at TREC-7. In *Proceedings of the Seventh Text Retrieval Conference (TREC7), National Institute of Standards and Technology (NIST), Gaithersburg, MD.*

Hiemstra, D. and Kraaij, W. (1999). TREC-7 working notes: Twenty-One in ad-hoc and CLIR. In *Proceedings of the Seventh Text Retrieval Conference (TREC7), National Institute of Standards and Technology (NIST), Gaithersburg, MD.*

Hull, D. and Grefenstette, G. (1996). Querying Across Languages: A Dictionary-based Approach to Multilingual Information Retrieval. In *Proceedings of the 19th ACM SIGIR Conference on Research and Development in Information Retrieval, Zurich, Switzerland.*

Littman, M. L., Dumais, S., and Landauer, T. K. (1998). Automatic cross-language information retrieval using latent semantic indexing. In Grefenstette, G., editor, *Cross-Language Information Retrieval*, chapter 5. Kluwer Academic Publishers, Boston.

McCarley, J. S. (1999). Multilingual Information Retrieval at IBM. In *Proceedings of the Seventh Text Retrieval Conference (TREC7), National Institute of Standards and Technology (NIST), Gaithersburg, MD.*

Schäuble, P. and Sheridan, P. (1998). Cross-Language Information Retrieval (CLIR) Track Overview. In *Proceedings of the Sixth Text Retrieval Conference (TREC6), National Institute of Standards and Technology (NIST), Gaithersburg, MD.*

Yamabana, K., Muraki, K., Doi, S., and Kamei, S. (1998). A language conversion front-end for cross-language information retrieval. In Grefenstette, G., editor, *Cross-Language Information Retrieval*, chapter 8. Kluwer Academic Publishers, Boston.

The TREC-7 Filtering Track: Description and Analysis

David A. Hull

Xerox Research Centre Europe
6 chemin de Maupertuis, 38240 Meylan France
hull@xrce.xerox.com

Abstract

This article describes the experiments conducted in the TREC-7 filtering track, which consisted of three subtasks: adaptive filtering, batch filtering, and routing. The focus this year is on adaptive filtering, where the system begins with only the topic statement and must interactively adjust a filtering profile constructed from that topic in response to on-line feedback. In addition to motivating the task and describing the practical details of participating in the track, this document includes a detailed graphical presentation of the experimental results and provides a brief overall analysis of the performance data.

1 Introduction

Text filtering is the process of sorting through large stores of textual data for information which fits a user's profile. This task is crucial for information processing in the modern age. Filtering is often described as the inverse of the traditional search problem. Instead of providing a query which is processed once on a static document collection, searchers define a persistent profile which is compared to a stream of arriving documents. Text filtering or routing has been a core task in TREC since the first year, and it continues to evolve to reflect changes in the way information is processed in the real world. The current focus of the filtering track is on the realistic simulation of on-line time-critical text filtering applications. There are many important aspects to the text filtering problem, but the TREC experiments are designed primarily to analyze the performance of document ranking and selection algorithms. Speed and scalability are important only to the extent that they enable systems to process 50 user profiles over a stream of a few hundred thousand documents in reasonable time. Evaluation is based only on the quality of the retrieved document set.

The core filtering task can be described briefly as follows:

> Given a topic description, build a filtering profile which will select the most relevant examples from an incoming stream of documents. As the document stream is processed, the system may be provided with a binary judgement of relevance for some of the retrieved documents. This information can be used to adaptively update the filtering profile.

Filtering differs from search in that documents arrive sequentially over time. The filtering track is divided into three subtasks: adaptive filtering, batch filtering, and routing. In adaptive filtering, systems start with only the original user profile which they must use to build a text classification rule. In batch filtering and routing, the systems can also take advantage of a large set of training

documents, a sample of which have already been evaluated as relevant or not relevant with respect to that profile. The difference is that in batch filtering, the system must decide either to accept of reject each document, while in routing, the system can return a ranked list of documents. The most important change between the TREC-6 and the TREC-7 experiments is the new emphasis this year on the adaptive filtering task.

The search task has traditionally been evaluated by average precision curves and other rank-based measures which use an extensive ranked list based on the scores of all incoming documents to determine performance. This list cannot be created until all documents have been scored and ranked, which means that system performance is not measured as a function of time. However, the underlying model upon which filtering is based is one where a stream of documents are compared to a long-lasting set of query profiles. This means that adhoc retrieval and routing simulate a non-interactive process where users look at documents only once at the end. For filtering, it is more realistic to assume that users examine documents periodically over time. The actual frequency of user interaction is unknown and task-dependent. Rather than attempt to simulate a particular task which might allow for batching and partial ranking of the document set, we choose to operate at the opposite end of the spectrum, assuming that the user wants to be notified about each potentially interesting document immediately after it arrives. This means that the decision to accept or reject a document must be independent of subsequently arriving documents. Therefore, traditional rank-based evaluation measure are not appropriate. Adaptive and batch filtering results will consist of unordered sets of documents which will be analyzed using set-based evaluation measures. More background on the filtering track, including some specific scenarios, can be found in the TREC-5 filtering track description [2].

2 TREC-7 Task Description

The primary goal for TREC-7 is to move towards a more realistic simulation of the filtering task by concentrating on adaptive filtering. Due to the large number of training documents evaluated for relevance, routing and batch filtering can be viewed as static machine learning or text classification tasks. This is an unrealistic simulation of most filtering applications for the following reasons. First, the training set is far too large. Second, the training set has been built through the TREC pooling process, meaning that the documents were retrieved from many different systems. Most filtering systems must rely on user feedback from documents they have found themselves or from the results of a single independent search. Adaptive filtering addresses this issue by eliminating the training documents. However, batch filtering and routing are still included to maintain continuity with previous years and to encourage as broad a range of systems to participate as possible. It is very important to continue these tasks because they substantially improve the quality of the document pool used for evaluation. Adaptive filtering is a much harder task, meaning that the percentage of relevant documents retrieved in these runs will be much lower. The other tasks will help us to get a broader coverage of the relevant document space. This section is devoted to a detailed description of the three subtasks and the topics and documents used in each task.

The primary corpus for the TREC-7 filtering experiments consists of 3 years of Associated Press newswire covering the period 1988-1990. There are approximately 80,000 documents from each year for a total of 240,000 documents. The AP collection is partitioned by year over the first three TREC disks. The documents are ordered roughly by date on the disks, and all systems were required to process the documents in this order. The AP newswire covers a broad variety of domains and the documents average roughly 450 words in length. This year's experiments used topics 1-50 which were constructed for the first TREC conference. These topics are quite long detailed descriptions

of the information need, ranging from 50-250 words in length with an average length of roughly 100 words. They are divided into title, description, narrative and concept fields. For an example of the style of these long topics, see the introductory paper in the TREC-6 proceedings [4]. Since topics 1-50 have been used at several previous TREC conferences, there are relevance judgements available for the 1988 and 1989 AP collection. However, the 1988 judgements do not have the quality and coverage typically associated with TREC, since they come from the TREC-1 experiments. The small number of first-year systems had to struggle just to complete the task and the documents were sampled in less depth than in recent years. Access to these relevance judgements was restricted in different ways depending on the subtask. The 1990 AP collection represents a clean test set, since it has no relevance judgements for these topics.

2.1 Adaptive Filtering

Previous versions of the TREC filtering task have been viewed as unrealistic because they provide too much training data, and because this training data comes from the previous search results of many different systems which use many different search algorithms. In practice, most systems can only expect relevance judgements from documents which they have been responsible for retrieving. The adaptive filtering task is designed to model this situation. In this task, each system starts only with the topic description and no evaluated documents. Documents arrive sequentially and the system can update the query profile in response to previously viewed documents. In addition, each document retrieved will be immediately evaluated for relevance, and that information will be passed on to the system.

It is not possible within the TREC framework to actually provide new relevance judgements to filtering systems as they see documents. Instead, the interactive component must be simulated using training data and previously obtained relevance judgements. In this model, the relevance judgement is available but hidden from the system until it decides whether or not it will retrieve a document. Relevance judgements from unretrieved documents are never revealed to the system. Note that test documents are evaluated in the same fashion, but relevance judgements are not immediately available (unless the system has a user providing manual feedback). This means that systems may wish to select a document not only according to its likelihood of relevance, but also according to its value as a training observation to improve future filtering performance. This makes adaptive filtering a more interesting and more complex task. The final retrieved set consists of only the documents selected during the first pass over the collection. Systems are not supposed to backtrack and use an improved user profile to retrieve previously ignored documents. Unfortunately, this point was not entirely clear in the initial track description and several systems did not completely adhere to this restriction.

Systems had the option to choose whether to use the rest of the TREC document collection (excluding AP) to generate collection frequency statistics (such as IDF) or auxilliary data structures (such as automatically-generated thesauri) or begin with no prior information. Systems were given access to relevance judgement for retrieved documents during processing, but only if the document was selected for that topic. Systems were specifically prohibited from using relevance judgements from unretrieved documents or relevance judgements for topics 1-50 from other parts of the TREC collection. The text from each processed document could be used to update term frequency statistics or auxilliary data structures whether or not the document was retrieved for any topic. There was also the option to treat unevaluated training documents which were retrieved as not relevant. Evaluation was based on two utility functions, as described in section 2.3.

2.2 Batch Filtering and Routing

In batch filtering, the documents and relevance judgements from the 1988 AP collection are available to all systems as a training set. Systems are prohibited from using relevance judgements for topics 1-50 from any other part of the TREC collection. As in adaptive filtering, systems can also have instant access to the relevance judgement (if it exists) for any document retrieved while processing the rest of the AP collection. Evaluation is based on utility. In routing, like batch filtering, the documents and relevance judgements from the 1988 AP collection are provided in advance as a training set. In addition, routing systems are free to use relevance judgements from other parts of the TREC collection (except 1989-1990 AP) for constructing the initial filtering profiles. The other difference is that routing systems return a ranked list of the top 1000 retrieved documents, and are evaluated according to average uninterpolated precision, much like adhoc search. Batch filtering and routing are included in the TREC-7 task to open participation to as many groups as possible and to improve the quality of the document pool used for evaluation.

2.3 Evaluation

Filtering systems are expected to accept or reject each document as it arrives and it is assumed that the user may well look at accepted documents immediately. Therefore, the output of the filtering system is treated as an unordered set of documents. This means that evaluation measures based on a ranked set of documents, such as precision-recall curves, are not appropriate. It is also important to handle topics with no relevant documents in a reasonable fashion, since in this case a system must keep the retrieved document set as small as possible. Classical set-based evaluation measures from information retrieval such as raw precision and recall do not behave gracefully for topics with few or no relevant documents. For example, the precision of a system which returns one non-relevant document is zero. The precision of a system which returns one thousand non-relevant documents is also zero. However, the former system is doing a far better job of filtering than the latter. As an alternative, we rely on a measure from information theory known as utility.

2.3.1 Utility

Utility assigns a value or cost to each document, based on whether it is retrieved or not retrieved and whether it is relevant or not relevant, as shown in the contingency table below:

	Relevant	Not Relevant
Retrieved	R+ / A	N+ / B
Not Retrieved	R- / C	N- / D

$$\text{Utility} = A*R+ + B*N+ + C*R- + D*N-$$

The variables R+/R-/N+/N- refer to the number of documents in each category. The utility parameters (A,B,C,D) determine the relative value of each possible category. A positive utility parameter can be thought of as the value of each document in that category, while a negative utility parameter is the cost of classifying a document in that category. Therefore, the larger the utility score, the better the filtering system is performing for a given query profile. For TREC-7, we test two different settings of the utility parameters:

```
F1 = 3*R+ - 2*N+        --> retrieve if P(rel) > .4
F3 = 4*R+ - N+          --> retrieve if P(rel) > .2
```

Note that the second utility function is labelled F3 to distinguish it from the slightly different F2 utility function which was used last year. The F2 function had a negative C parameter. This function was changed because the true number of unretrieved relevant documents is difficult to estimate and because most systems returned a negative F2 utility. This does not necessarily mean that they were performing badly, rather many topics had a large number of relevant documents. Filtering according to a utility function is equivalent to filtering by estimated probability of relevance. Therefore, the utility functions above are listed with the appropriate probability thresholds. Readers can find the general formula for converting a utility function into a probability threshold in Lewis [3], and a derivation of the formula can be found in any general book on decision theory.

Utility is not an ideal measure for judging the performance of filtering systems for a number of reasons. Utility scores will vary widely from topic to topic based on the number of relevant documents, and it is difficult to average or compare them across topics. Since we use linear utility functions, all relevant documents are equally important, no matter how many exist or have been retrieved for a given topic. There is no sense of diminishing value for the thousandth relevant document. In practice, users might well have different utility functions for each topic, based on the nature of the topic and their information need. For administrative and practical reasons, it is too difficult to define a separate utility function for each topic. However, the utility measure handles topics without relevant documents easily, which is an important advantage. We will start to look at techniques for scaling and normalizing utility in this year's evaluation. The most appropriate evaluation strategy for real filtering systems depends on the task and the user model.

2.3.2 Pooling vs. Sampling

In general, the size of the retrieved set is unbounded, making accurate evaluation of performance difficult for some topics, regardless of measure. The traditional pooling approach to document assessment has been augmented with a random sampling strategy. In pooling, the top N retrieved documents from each run and each topic are merged to create a single document pool which is assessed. All documents which do not appear in that pool are assumed to be not relevant. This strategy is reasonably fair for all participants, but results in performance estimates that are biased downwards. For filtering, the retrieved set is unranked, so one cannot simply select the top N documents. The approach for filtering is therefore to select a random sample of size N from the retrieved set for each system. If the retrieved set is smaller than N, all documents are selected.

Pooling is less than ideal for filtering topics where the retrieved sets are much larger than N. First of all, the filtering pool is of lower quality because the documents are randomly sampled from a large retrieved set rather than obtained by selecting the top ranked documents. Fortunately, this effect is mitigated somewhat in the TREC-7 experiments because some routing runs, which are based on ranked retrieval, also contribute to the pool. Second, the topics with large retrieved sets are the ones which will tend to have the most relevant documents, and thus will suffer from the most bias due to incomplete assessment. Fortunately, we know from sample theory that the proportion of relevant documents in a simple random sample is an unbiased estimate of the proportion of relevant documents in the population. For the utility function F1, we can convert an estimate of the proportion of relevant documents directly into an estimate of utility via the following formula ([2], p.81, eq 1):

$$\text{F1} = ((A - B) * (r/n) + B) * N$$

where n is the size of the sample, r is the number of relevant documents in the sample and the other terms are the same as defined above. Another nice property of the sampling approach to evaluation is that we can also calculate the standard error of the utility estimate, which is given by

Utility	R1	R2	R3	R4
T1	0	0	-18	4
T2	150	276	160	75
T3	-6	-44	-43	-11

Rank	R1	R2	R3	R4
T1	2.5	2.5	1	4
T2	2	4	3	1
T3	4	1	2	3

Table 1: Pseudo-example of ranking runs R1-R4 for topics T1-T3.

([2], p.87, eq 30):

$$\text{SE(F1)} = ((A-B)^2 N \frac{(N-n)r(n-r)}{n^2(n-1)})^{1/2}$$

For more details on the mathematics of sampling, see Lewis [2]. Sampling provides unbiased estimates of utility, but there is a price. Since a lot of information is thrown away (all the relevance judgements for documents sampled by other systems but not the one being evaluated), the sampled estimates tend to have much more variation.

2.3.3 Averaging and Comparing Systems with Utility

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). Therefore, we use two different techniques to average performance, ranking within topics and scaling of the utility scores. Average rank statistics have been used for the last two filtering tracks while scaling of the utility function is new this year. Each method has its advantages and disadvantages, as will be described below.

The average rank measure is computed in two steps: (1) for each topic, systems are ranked in order of their performance, (2) the ranks are averaged by system over all topics. Systems are ranked from lowest utility to highest utility, meaning that the larger the average rank score, the better the system is performing with respect to its competitors. Table 1 presents a pseudo-example of the ranking process. The advantage of the average rank measure is that all topics are equally important in determining a system's performance. The measure is insensitive to topics with very large retrieved sets and wide variation in the utility scores. Average rank scores generated by the same set of systems are directly comparable, even if they are based on different evaluation measures or different retrieval tasks. This makes it possible to compare systems in situations where it would otherwise be difficult (as is the case with utility). On the negative side, all topics are equally important, meaning that topics with large variation which reflect real differences between systems do not receive higher weight. There is no absolute standard of performance and scores are only meaningful relative to the set of systems being analyzed. The results depend on the systems being compared, so adding or removing a system will change the scores of other systems.

We can use non-parametric statistical tests to determine the significance of average rank differences between systems. In order to keep the results simple and readable given the large number of experiments and competing systems, we will only present the pairwise comparisons with respect to the best performing system. For each experiment, we apply two different tests: a conservative test (a non-parametric variant of the Newman-Keuls multiple comparisons test) and a more powerful test (a non-parametric variant of the Least Significant Difference (LSD) test). For more information on the statistical tests and their performance characteristics in TREC-style IR experiments, please consult our NIST technical report [1]. The alpha level is set at 0.05, which corresponds to the error rate for each pairwise comparison in the LSD. The true pairwise error rate for the Newman-Keuls test is an order of magnitude smaller.

While direct averaging of utility scores is undesirable, due to the wide variation in the number of relevant documents per topic, it is possible to scale the utility scores prior to averaging.[1] The most obvious scaling strategy is to divide by the maximum possible utility score for each topic. However, this approach is seriously flawed for negative utility scores. A system which returns one hundred non-relevant documents will receive a score of -100 for a topic with one relevant document and -1 for a topic with one-hundred relevant documents. Since the maximum possible positive score on a topic is 1, topics with negative utilities will dominate the average. Therefore, a more complex utility scaling function is required.

For the TREC-7 experiments, we test the following utility scaling function:

$$u_s^*(S,T) = \frac{\max(u(S,T), U(s)) - U(s)}{MaxU(T) - U(s)}$$

where $u(S,T)$ and $u_s^*(S,T)$ are the original and scaled utility of system S for topic T, $U(s)$ is the utility of retrieving s non-relevant documents, and $MaxU(T)$ is the maximum possible utility score for topic T. This scaling function assigns a lower bound to the utility function which can be set with the parameter s. There is a reasonable justification for this approach. Assume that a filtering profile is performing really poorly. At some point, the user will get fed up with reading non-relevant documents and delete the profile entirely. The parameter s sets the number of non-relevant documents at which the user's tolerance is exhausted. All utility scores less than $U(s)$ are set to $U(s)$. Therefore, utility scores can range between $U(s)$ and $MaxU(T)$ and the scores are renormalized to range between 0 and 1 and then averaged.

The parameter s is set once for all topics, and unlike average rank, a system's scaled utility score is absolute, remaining the same regardless of the other systems are being evaluated. The scaled score can be interpreted relative to the perfect system (utility of 1.0) and the worst possible system (0.0). Unfortunately, the average scaled utility score is highly dependent on the definition of the worst possible system, as determined by the parameter s, and so is only meaningful in relation to a particular lower bound. Since the decision on where to put the lower bound is fairly arbitrary, we will plot the average scaled utility scores over a range of values for s (25-800). A low value for s differentiates more between systems that do well on topics with few relevant documents. On the other hand, it reduces the penalty for systems which do very poorly on these or other topics. One could define a two-parameter scaling model which sets the zero point for utility scaling and the lower bound for acceptable performance to different values, thus allowing for negative scaled utility. This would make it possible to distinguish to the two types of behavior described above. However, for simplicity, we choose to use the same value for the zero point and the lower bound in these experiments.

2.4 Submission Requirements

Each participating group could submit up to four adaptive filtering runs and up to two runs each for batch filtering and routing. There were no required runs this year. Runs were classified into one of three categories:

(A) Automatic - Any run which uses fully automatic methods for profile construction and updating. This can include automatic learning from test documents as they are filtered.

(B) Manual - Any run which uses manual techniques for profile construction, up to and including making additional relevance judgments on training documents. No manual intervention

[1] Paul Kantor first proposed that scaling is possible, while Stephen Robertson suggested the initial variant of the scaling method described here.

based on information from the test documents is allowed, although automatic learning is still permitted.

(C) Manual Feedback - Any run which uses manual techniques for updating profiles based on previously viewed test documents. The run may or may not also use manual techniques for profile construction.

There are no training documents for adaptive filtering, so manual intervention (B) is limited to profile construction for this task. In practice, none of the groups submitted manual runs. Groups were also asked to indicate whether they used other parts of the TREC collection to build term collection statistics or other resources.

Due to assessment constraints, NIST only promised to evaluate a maximum of 100 documents per topic per participating group. In the end, NIST had enough resources to evaluate two sets of 100 documents per group. These documents were obtained as follows. Each group was asked to select one run based on the F1 utility function and one run based on the F3 utility function for sampling. If the document set for the F1 utility run on topic T was greater than 100 documents, 100 documents were sampled for assessment. Otherwise, all documents were assessed. Furthermore, a sufficient number of document from the F3 Utility set were sampled to bring the total set size up to 100 documents. If the union of the F1 Utility and F3 Utility sets was less than 100 documents and the group submitted a ranked retrieval run, then the sampling program went down the list in rank order adding new documents until a sample of size 100 was extracted. If the group submitted two sets of adaptive filtering runs, this process was repeated for the second set.

3 TREC-7 results

The TREC-7 filtering track had 12 participating groups who submitted a total of 46 runs, distributed as follows:

	# groups	# runs
Total	12	46
-----	--	--
adaptive	8	24
batch	6	12
routing	6	10

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.

- AT&T Labs Research [AT&T] (att98f)
- Canadian Imperial Bank of Commerce / Active On-Line Systems [CIBC] (sigma)
- City Univ. London / Univ. Sheffield / Microsoft [CitySM] (ok7ff)
- CLARITECH Corporation [CLARITECH] (CLARIT)
- IBM Almaden Research Center [IBM-A] (arc98cs)
- IRIT / University of Toulouse (IRIT) (Mer)
- University of Iowa [UIowa] (IAHK)

Organization	1st yr?	Adaptive	Batch	Routing	Resources
AT&T	-	-	X	X	-
CIBC	X	X	-	-	-
CitySM	-	X	-	-	TREC
CLARITECH	-	X	X	-	TREC
IBM-A	X	-	-	X	N/A
IRIT	X	X	X	X	-
UIowa	X	X	X	-	-
UMass	-	X	-	-	TREC
NTT Data	X	-	X	X	E/J dict
CUNY	-	X	X	X	TREC
Rutgers-K	-	-	-	X	N/A
TNO	X	X	-	-	TREC

Table 2: Listing of subtask participation and resources used for each group (X = participant).

- University of Massachusetts Amherst [UMass] (INQ5)
- NTT Data Corporation [NTT Data] (nttd7)
- Queens College CUNY [CUNY] (pirc8)
- Rutgers University [Rutgers-K] (AntRout)
- TNO-TPD/HFRI [TNO] (TNO)

Table 2 lists the subtasks each group participated in, the resources they used (TREC = documents from other parts of the TREC collection, E/J dict = English/Japanese phrase dictionary), and whether this is their first year participating.

3.1 Summary of approaches

In this section, we briefly describe the techniques used by each of the groups in their experiments. The information presented here is based on the material provided by the participants in their draft papers, and so should be considered preliminary and incomplete. For more information, please consult the site report papers included in this volume [5]. IBM-A and Rutgers-K did not provide notebook papers so there is no information available on these systems. Both participated only in the routing subtask.

AT&T participated in batch filtering and routing. One routing run (fr5) was based on a scaled down version of their TREC-6 routing system. The other routing run (fr4) and the batch filtering runs were generated from their new ATTICS toolkit for machine learning of text classifiers. The TREC-7 runs use a two-pass Rocchio algorithm, the first pass based on only the judged documents and the second passed based on the judged and top 5000 unjudged documents (assumed not relevant). The version of ATTICS used for TREC-7 had no feature selection, stop listing, or term weighting.

CIBC participated in adaptive filtering. Their system uses an adaptive Rocchio relevance feedback algorithm with initial feature weights based on the AT&T TREC-6 system. The initial profile is represented by its 20 most important terms and query expansion is limited to 20 additional terms. The system is built on top of a multi-agent architecture, with agents assigned to feature extraction, profile generation, and profile selection (thresholding).

CitySM participated in adaptive filtering. They concentrated on accurate threshold selection. The runs ok7ffx2 use the same initial query throughout while runs ok7ffx3 use probabilistic reweighting. No query expansion was performed. Thresholds are initially calibrated by applying logistic regression to the ratio of the observed score to the average score of the top 1% of retrieved documents (ast1). A two parameter logistic model is fit to an independent topic and document set. Both ast1 and the intercept term are adaptively adjusted on a weekly basis as training data becomes available. CitySM assumed that there is an advantage to selecting more documents at the beginning to help calibrate the thresholds more accurately. Therefore, the system uses a ladder of thresholds. Each profile starts at the bottom of the ladder and the threshold is increased as relevant documents are discovered.

CLARITECH participated in adaptive and batch filtering. They used their TREC-6 Rocchio relevance feedback algorithm for term selection and weighting and concentrated on accurate threshold selection for adaptive filtering. The initial thresholds are set to a fixed delivery ratio of documents (e.g. 1 out of 2000) tuned on an independent corpus. Thresholds are allowed to vary between a lower bound zero utility value and an upper bound optimal value. The optimal threshold gives the best utility on the training data. The zero utility threshold is the largest score threshold which generates negative utility. The actual threshold is determined by two parameters, a fixed threshold bias and a correction factor which is based on the system's confidence in its estimate of the optimal threshold. Documents are batched in gradually increasing batch sizes ranging from 3000 initially up to 24,000. The runs (afFxb) stop retrieving documents on AP90 for topics which have a negative accumulated utility over AP88 and AP89.

IRIT participated in all three subtasks. Their system is based on a 3-layer neural network model, with layers consisting of queries and documents connected by a layer of terms. For batch filtering and routing the system computes the difference between the observed output of the network and the desired output based on the judged training documents, and this error term is propagated back through the network. This process is repeated over many iterations. A similar approach is used for adaptive filtering. Unfortunately, they misunderstood the adaptive filtering task and thought that they were supposed to rerun the final adaptive filtering profile over the entire AP collection to generate the retrieved set. Clearly, this approach leads to much better performance than selecting documents only during the first pass, so their runs were deleted from the adaptive filtering comparison graphs. However, the results of their system can still be found in the system by topic performance tables.

UIowa participated in adaptive and batch filtering. Their system introduces a two-level clustering approach to filtering. The original topic statements are used to construct a *primary* cluster profile with a low fixed threshold for membership. Documents which exceed this threshold are then clustered dynamically into a set of *secondary* sub-clusters. When the similarity between the secondary cluster and the primary cluster profile exceeds a certain visibility threshold, all documents in the secondary cluster are presented to the user for evaluation. Note that this actually violates the conditions of the filtering track which state that a binary decision to accept or reject must be made immediately after it is filtered. Each secondary cluster is then labelled red (don't retrieve) or green (retrieve) according to the relative frequency and density of relevant, non-relevant, and evaluated documents. Future members of these sub-clusters are treated accordingly.

UMass used the same incremental Rocchio filtering algorithm (InRoute) presented at TREC-6. This year the system used more complex initial queries (similar to adhoc) and a *high precision* setting for the filtering threshold, resulting in a conservative document selection strategy.

NTT Data participated in batch filtering and routing. Their system uses Rocchio relevance feedback and dynamic feedback optimization based on terms and term pairs, following the model of the AT&T TREC-6 routing/filtering system. In addition, they extracted conjunctions of more

than two terms by running a classification tree algorithm (C4.5) over the training data to find larger groups of terms strongly associated with relevance. No information about threshold setting for batch filtering is provided in the notebook paper.

CUNY participated in all three subtasks, although only the adaptive filtering experiments are described in the notebook paper. They adapted the profiles every 10,000 documents filtered using a probabilistic term reweighting scheme. No query expansion was performed. Retrieval thresholds are allowed to vary between a fixed upper and lower bound. The thresholds are set initially to the upper bound, meaning that the system is starting with a conservative document selection strategy. Thesholds are adjusted every 10,000 documents as a function of three parameters: number of documents which have passed through the system, number of documents examined, and number of documents relevant.

TNO participated in adaptive filtering. They used an adaptive Rocchio relevance feedback algorithm with the profile being built from the last N relevant and the last N non-relevant documents. Each profile starts with a fixed initial threshold. The threshold is adjusted to be the mid-point between the average similarity of the last N relevant and the average similarity of the last N non-relevant documents. The difference between the two submitted runs is the choice of the initial threshold (0.35 vs. 0.40). There is no tuning of thresholds to the utility functions.

3.2 Evaluation results

Figures 1-10 summarize the evaluation results for the TREC-7 filtering track. There are two types of graphs. Figures 1, 4, and 7-10 use average rank as the performance measure, and represent 2-3 different experimental runs on the same plot. Figures 2-3 and 5-6 use average scaled utility as the performance measure, and show a range of different values for the lower bound. In all plots, the run labelled *Baseline* is computed by retrieving no documents for every topic (i.e. a utility score of zero for all topics). The average rank plots have dashed lines connecting the same system over the different experiments. The vertical arrows link all the systems which are not distinguishable from the top performing system according to the statistical significance tests described in the previous section. The longer arrow always corresponds to the more conservative test. In the average scaled utility plots, the horizontal axis represents the number of non-relevant documents used to define the lower bound utility, plotted on a log scale. The vertical axis represents the difference in average scaled utility between each system and the baseline. Therefore, the baseline is represented by the straight horizontal axis marked with index points for the lower bound.

Figure 1 compares average rank performance for F1 and F3 utility for the adaptive filtering subtask. The system rankings don't change a lot, although the baseline drops substantially. This plot demonstrates that the F1 utility function sets an extremely challenging target, since most groups fall well below the baseline. Only CLARITafF1b comes close with a strategy that shuts down the filtering profiles which have accumulated a long-term negative utility. Several systems (TNO and IAHKaf12) seem to track the baseline more closely, generally because these systems have small or empty retrieved sets for many topics.

Figure 2 presents average scaled utility for the F1 function and the adaptive filtering subtask. Systems can be divided into two categories, TNO and IAHKaf11 which perform at a relatively constant level regardless of the lower bound, and the rest of the systems, which improve substantially as the number of non-relevant documents s used to define the lower bound is increased. CLARITafF1b clearly dominates the latter group of systems, entirely on the strength of its decision to shut down the profiles which had accumulated a negative utility. Figure 3 presents average scaled utility for the F3 function and the adaptive filtering subtask. In this case, most systems track the baseline regardless of lower bound, with the notable exception of IAHKaf32. For the F3 utility

	AntRout1	AntRout2	arc98cs	nttd7rt1	att98fr4	att98fr5	MerRou	pirc8R1	pirc8R2
≤4 AP88 rel	0.314	0.422	0.166	0.512	0.010	0.352	0.218	0.087	0.048
>4 AP88 rel	0.247	0.248	0.086	0.515	0.342	0.436	0.318	0.317	0.433

Table 3: Breakdown of routing performance for topics with few relevant documents on AP88.

function, average scaled utility seems like a reasonably good evaluation measure, since there is minimal dependence on the choice of lower bound.

Figure 4 compares average average rank performance for F1 and F3 utility for the batch filtering subtask. It appears that both NTT Data and CLARITECH have more success with the F3 utility function relative to the other participants. Figures 5 and 6 present average scaled utility for the F1 and F3 function respectively and the batch filtering subtask. For F1 utility, the CLARIT system performs much better with a more generous lower bound. This is largely a function of its decision to start with a delivery ratio threshold rather than a threshold estimated from the training data. This means that they retrieved non-relevant documents for all of the hard topics where it was better off to retrieve nothing and thus their average performance suffers when the lower bound emphasizes doing well on hard topics. In both plots, the systems converge more towards the baseline as the lower bound is decreased. This is a natural progression because the majority of batch filtering systems are doing a lot better than the baseline on most topics, but this advantage is reduced as the lower bound drops and the performance of a utility score of zero is more highly rewarded.

Figure 7 compares performance on the routing task between the raw average uninterpolated precision scores on the left axis and the average ranked precision on the right axis (rescaled to facilitate comparison). The two approaches to evaluation yield more or less the same results. Note that routing systems were allowed to use any existing relevance judgement for topics 1-50 outside the test collection. It is clear that not all systems took advantage of that extra data. Some systems appear to have used only AP88 training data for routing, since this is what was required for batch filtering. If we measure performance separately on the 10 topics with 4 or few relevant documents in AP88 (29-37 and 39) versus the rest, we note that systems fall into two categories (see Table 3). The ones on the left probably used all the training data, and do at least as well on the subset with few AP88 training examples. The systems on the right see their performance drop, often drastically. There is good reason to believe that some of these systems could improve their performance by using the full set of training documents.

Figures 8 and 9 plot average rank performance by year for F1 and F3 utility respectively and the adaptive filtering subtask. For F1 utility, the relative performance of systems changes substantially between 1988 and 1989. To a large extent, this reflects the decision of some systems (like CLARIT and ok7ff) to lower the thresholds early to gather more training data in the hopes of improving performance later. For F3 utility, many systems are bunched together around the baseline in 1988, and then they spread apart as some systems manage to take much better advantage of the arriving training data than others. This may reflect the fact that some systems are retrieving more documents in 1988 than others even though their performance in terms of utility is about the same. It is interesting to note that the INQ51x had some difficulty setting their initial thresholds for both utility functions (retrieving far too many documents in 1988) but recovered in 1989-1990 to perform very respectably. This is in spite of the fact that they thought they had chosen a conservative threshold setting strategy!

Figure 10 compares the pooling and the sampling strategy for evaluation. The results presented here are limited to the 14 topics where significant sampling took place. For all other topics, the retrieved sets are almost always less than 100 documents, so the pooled and the sampled results are virtually identical. Since the score differences over these 14 topics are also small, it is fair

to conclude that pooling introduces minimal bias in favor of any individual system. However, a single run, sigmaF1, does improve when evaluated by sampling, indicating that this system may be behaving differently from the others. The most extreme example of this pattern is topic 22, where sigma received a sampled score of 2197 and a pooled score of -844! The 95% confidence interval is plus or minus 930 documents so there is a lot of room for variation. However, it is likely there are still a fair number of unevaluated relevant documents for this topic which sigma was finding but not getting credit for. There are several other topics where sigma's pooled score is below the 95% confidence interval for its sampled score, indicating that it might have received unfair treatment from the pooling process.

4 General Commentary

The first, most obvious conclusion is that adaptive filtering is a much harder task than batch filtering. This is immediately evident when one looks at where the baseline ends up in Figures 1 and 4. This should come as no surprise since batch systems were able to take advantage of training sets consisting of hundreds of judged documents. In fact, we might consider whether the F1 utility function should be changed to reflect the fact that it is an unrealistically tough standard for adaptive filtering systems. Only one system managed to come within striking distance of the baseline (retrieving no documents!) and only by shutting down poorly performing profiles before the 1990 data were evaluated.

It is also clear that adaptive filtering is a very different task than batch filtering, requiring a much more careful approach to threshold selection. In batch filtering, threshold selection is important, but it is also crucial to learn a high quality initial profile based on the training data. In previous TREC filtering experiments, the same systems appear at the top of the rankings for both batch filtering and routing, indicating that a good ranking algorithm is very important. In order to build a good ranking algorithm, the system must learn effectively from the training data, and most successful systems used powerful statistical learning techniques. This is no longer necessary (or even possible) in adaptive filtering. In fact, the best performing systems from CLARITECH and CitySM did very little profile learning, using minimal (or no) query expansion and limited term reweighting. In fact, ok7ffx2 performed no profile adjustment at all and was still highly competitive with the best performing systems! All of its effort was directed towards adaptive threshold selection.

In adaptive filtering, it is important to recognize when a filtering profile is retrieving lots of non-relevant documents and adjust the threshold rapidly in response. CLARITafF3b took the extreme step of shutting down some profiles entirely and performed much better than the corresponding run which continued to retrieve documents. At TREC-7, most systems filtered a fixed size batch of documents before updating profiles. CLARITECH experimented with increasing size batches and found that this method works better, however they still used the same batch size for all topics. There are obvious efficiency advantages to updating all topic profiles simultaneously, which is a good reason to prefer this approach. However, the initial observation suggests that it might be better to update profiles as a function of documents retrieved rather than documents filtered. This makes intuitive sense, since a profile which has retrieved only a few documents is not going to change much whether 100 or 10,000 documents have been filtered. This will also allow the system to quickly adjust a profile that is going far off course and retrieving way too many documents as a result of a change in the distribution of arriving documents. However, variable topic updating would require a more complex filtering architecture.

In TREC filtering experiments, it is very important to adapt the threshold setting mechanism to reflect the utility functions used in evaluation. Several systems do not take this step, and their

performance suffers accordingly. When system performance is evaluated using other measures, such as precision and recall, these systems are often quite competitive. Therefore, it is unlikely that these systems would perform badly in a practical setting, since the utility functions are constructed artificially for our simulation experiments. However, it is important to have a threshold tuning mechanism which can respond to the user's needs.

There are still many interesting research topics to pursue, particularly in the area of adaptive filtering. For example, no one has yet explored whether the distribution of a feature over time is related to its usefulness as a discriminator for relevance. The filtering track will continue for TREC-8, with much the same breakdown of tasks as we have seen for TREC-7. We want to continue to encourage anyone with interest in the topic to consider participating in the filtering track next year. The track is designed to be flexible to allow groups with very different research interests to participate, and the barriers to entry are low. Participating groups will play a strong role in determining the direction of the TREC-8 filtering track.

Acknowledgements Many people have contributed to the development of the TREC filtering track, in particular David Lewis, Karen Sparck Jones, Chris Buckley, Paul Kantor, Ellen Voorhees, Stephen Robertson, the TREC program committee, and the team at NIST. I am merely building upon their work.

References

[1] David A. Hull and Paul B. Kantor. Advanced Approaches to the Statistical Analysis of TREC Information Retrieval Experiments. Contact the first author for a copy: hull@xrce.xerox.com, 1997.

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

[3] David D. Lewis. Evaluating and Optimizing Autonomous Text Classification Systems. In *Proc. of the 18th ACM/SIGIR Conference*, pages 246–254, 1995.

[4] Ellen M. Voorhees and Donna Harman. Overview of the 6th Text Retrieval Conference (TREC-6). In *The Sixth Text Retrieval Conference (TREC-6), NIST SP 500-240*, pages 1–24, 1998.

[5] Ellen M. Voorhees and Donna Harman, editors. *The 7th Text Retrieval Conference (TREC-7)*, 1999. To appear.

Figure 1 - Adaptive Filtering: Ranked F1 vs. F3 Utility

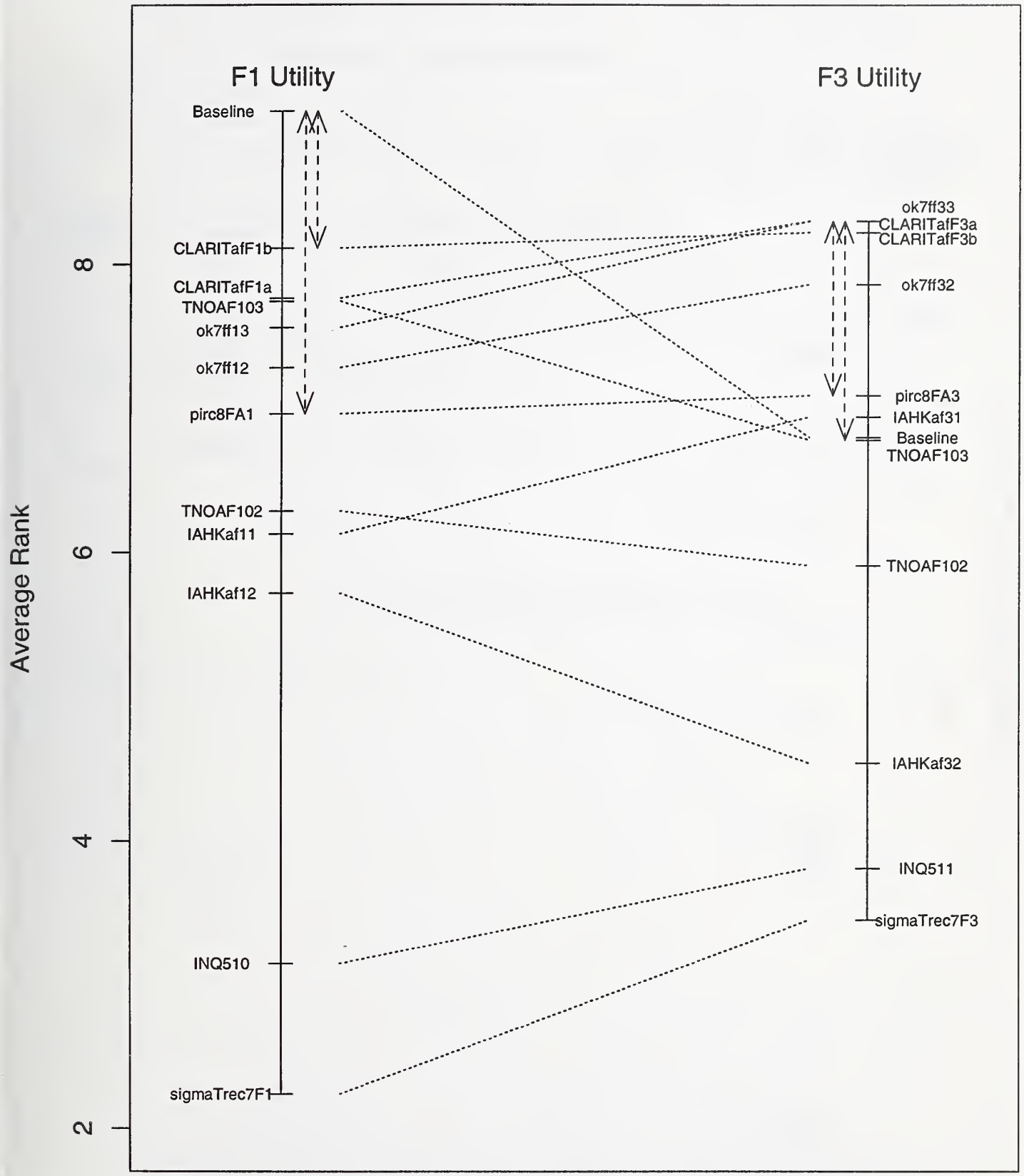


Figure 2 - Adaptive Filtering: Scaled F1 Utility

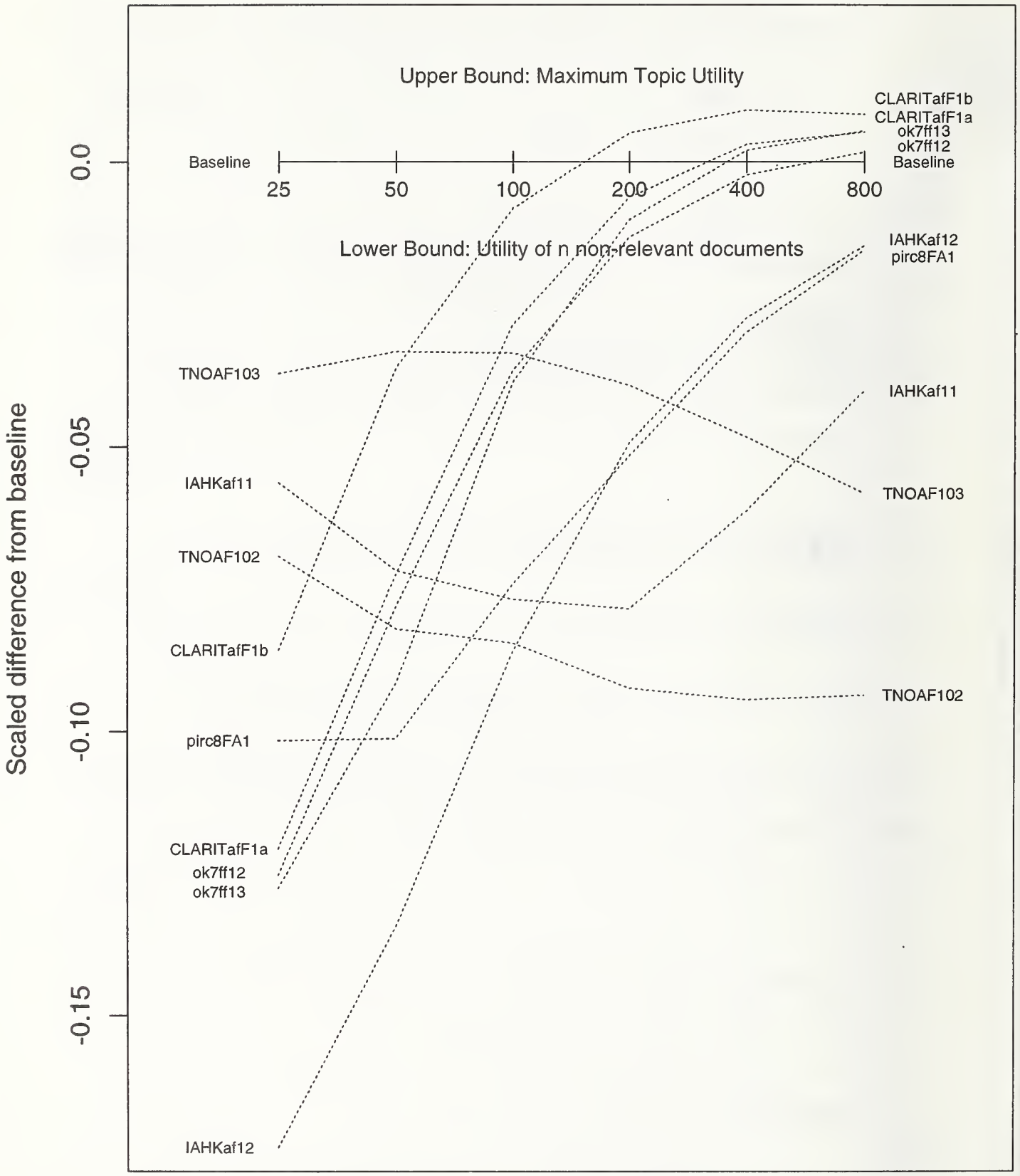


Figure 3 - Adaptive Filtering: Scaled F3 Utility

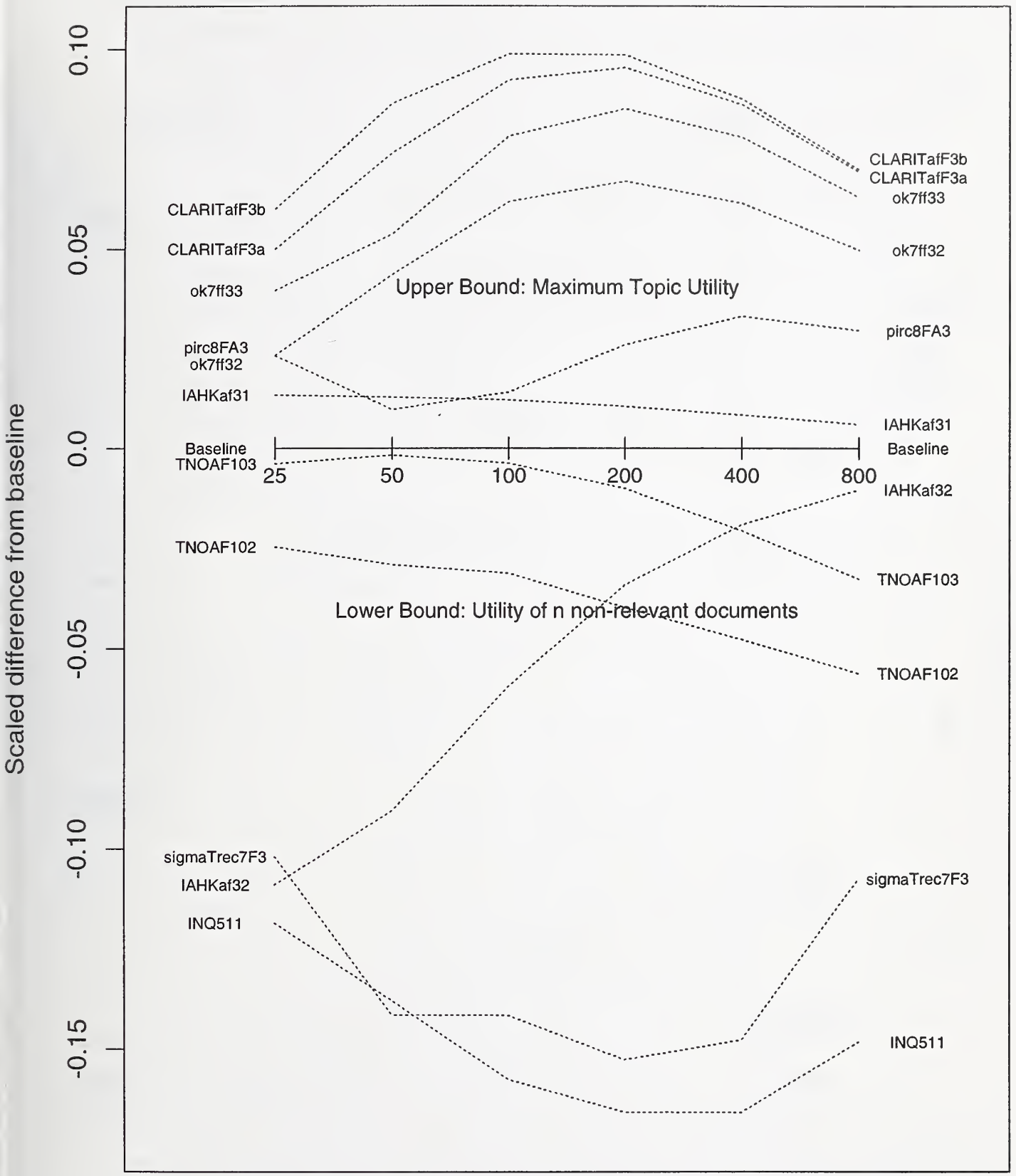


Figure 4 - Batch Filtering: Ranked F1 vs. F3 Utility

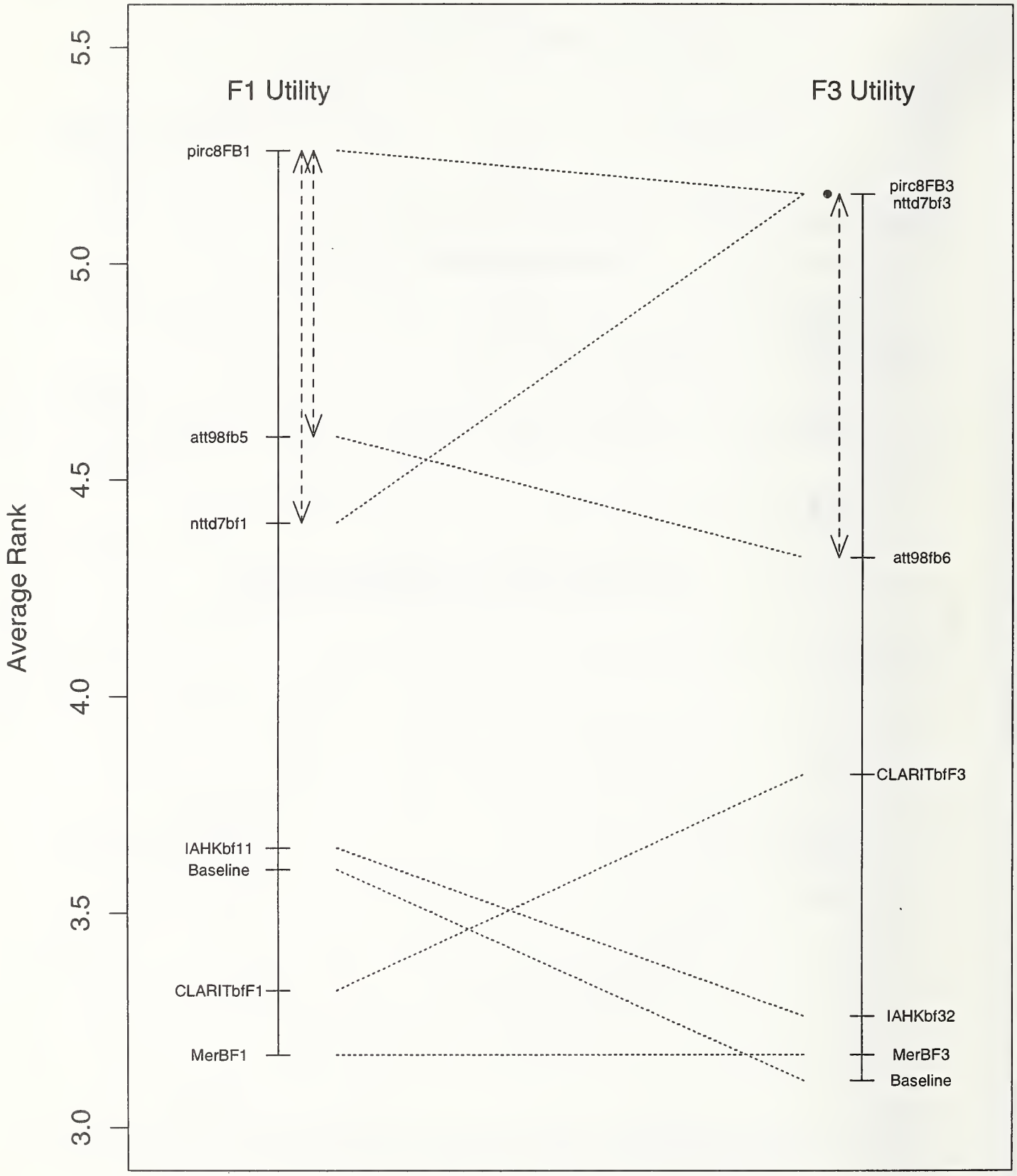


Figure 5 - Batch Filtering: Scaled F1 Utility

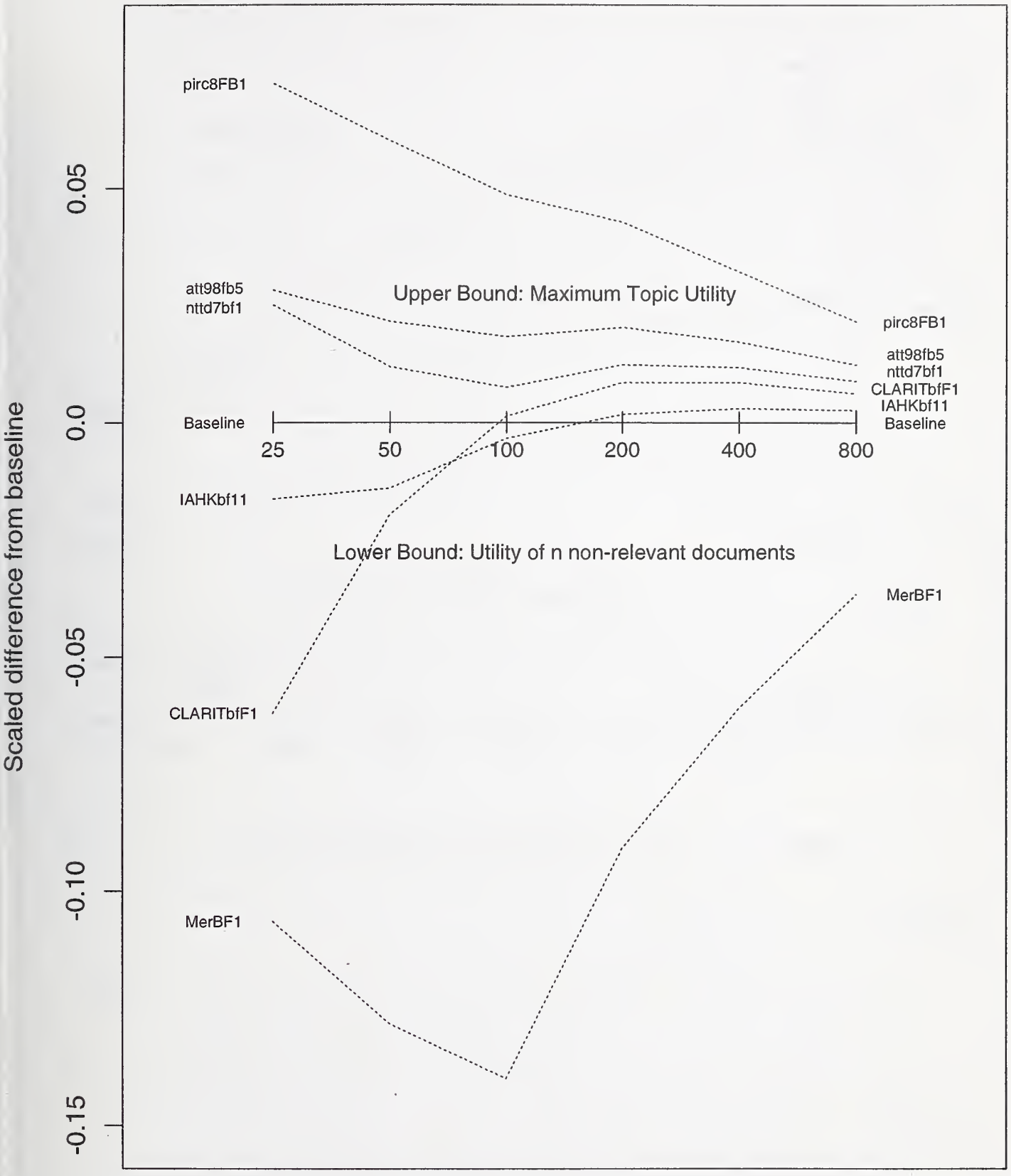


Figure 6 - Batch Filtering: Scaled F3 Utility

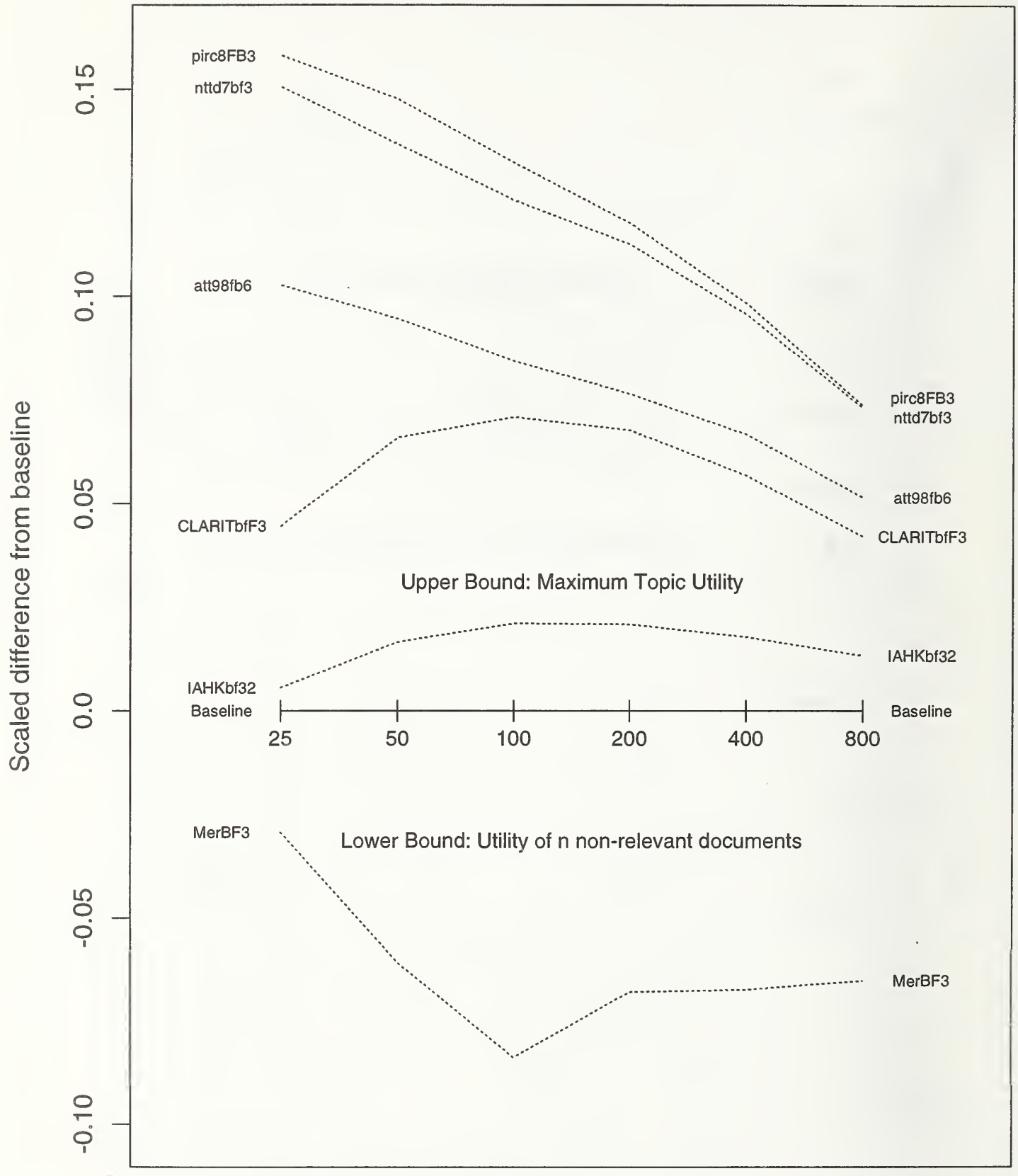


Figure 7 - Routing: Averaged vs. Ranked Precision

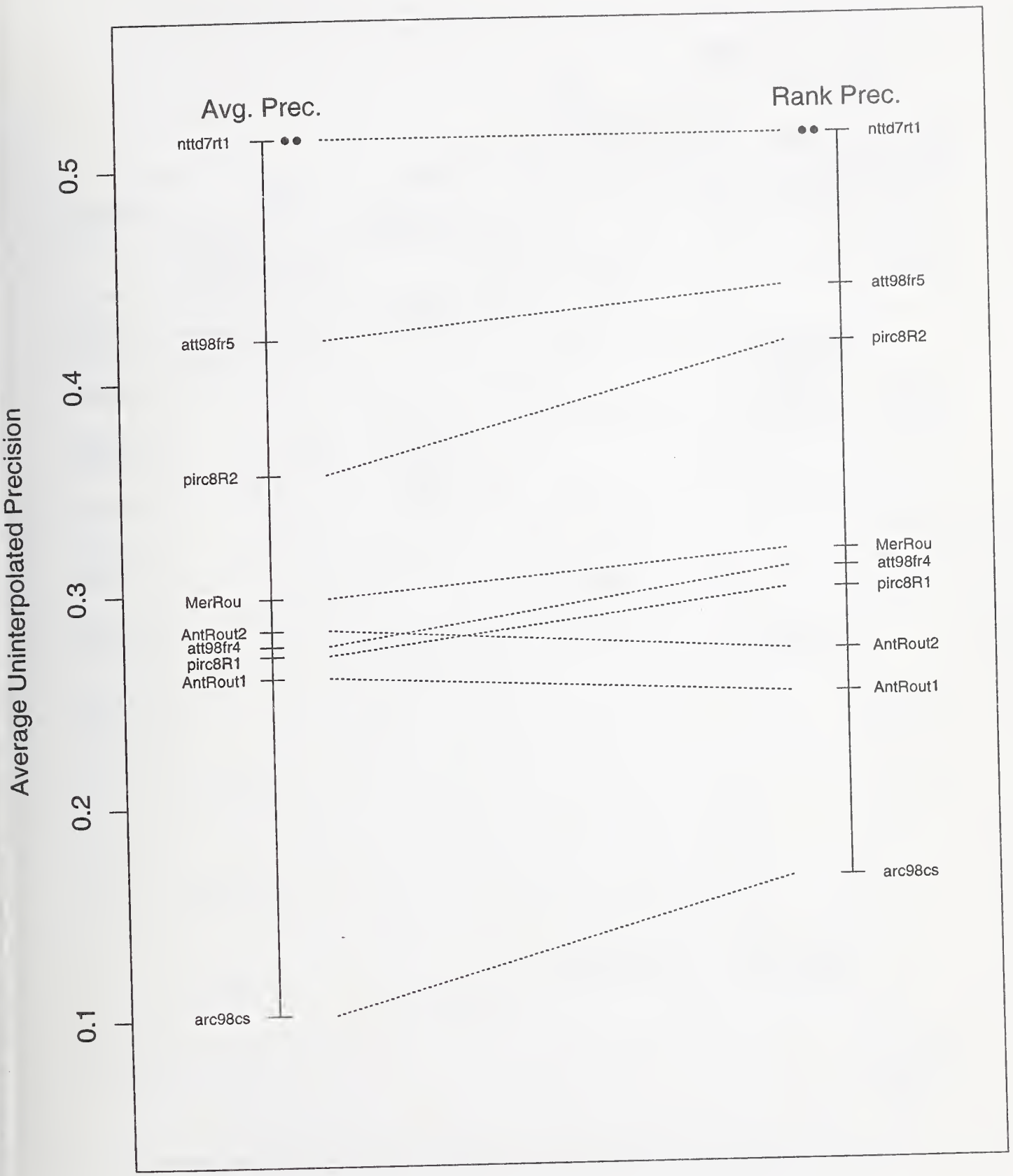


Figure 8 - Adaptive Filtering by year (Ranked F1 Utility)

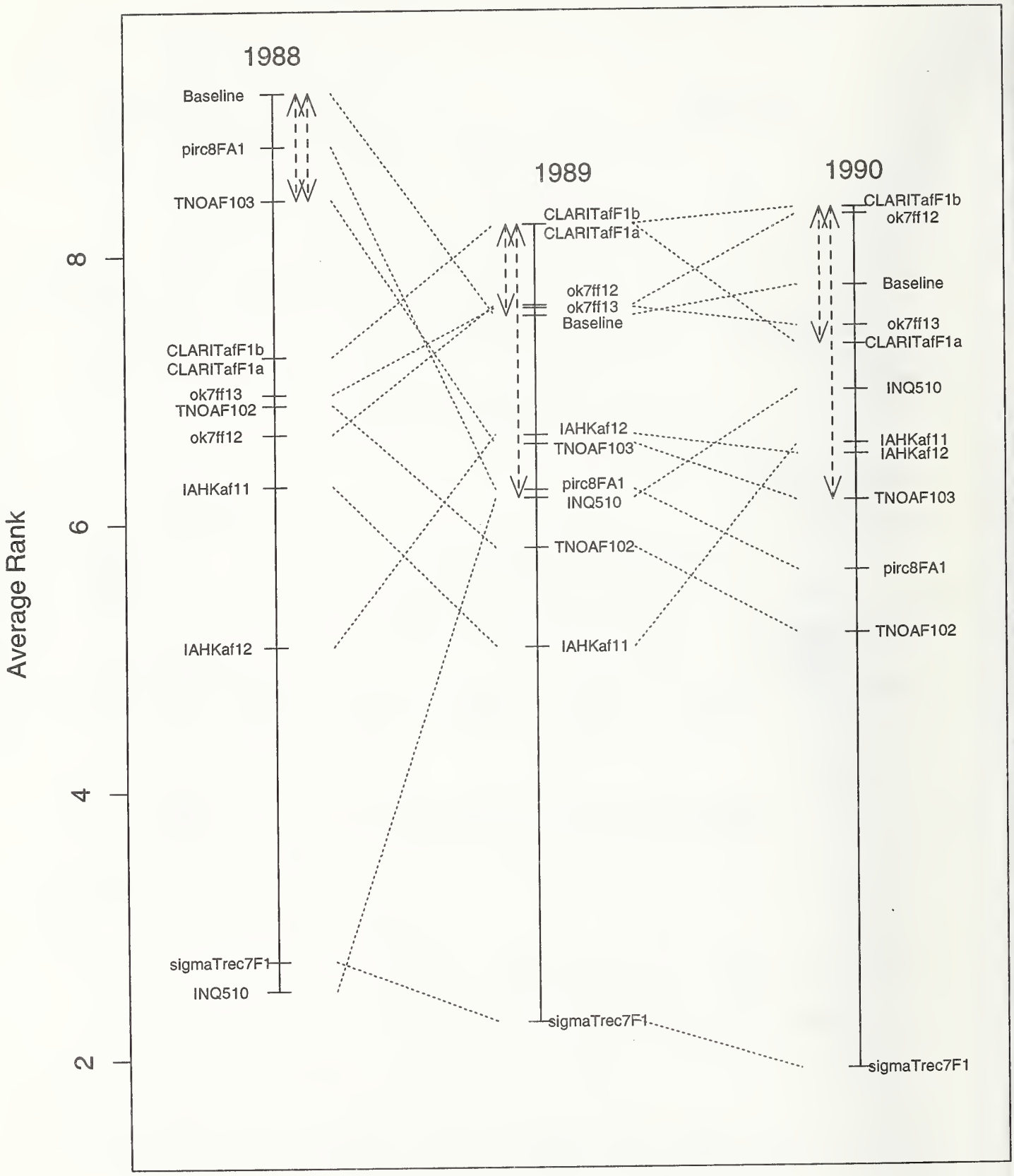


Figure 9 - Adaptive Filtering by year (Ranked F3 Utility)

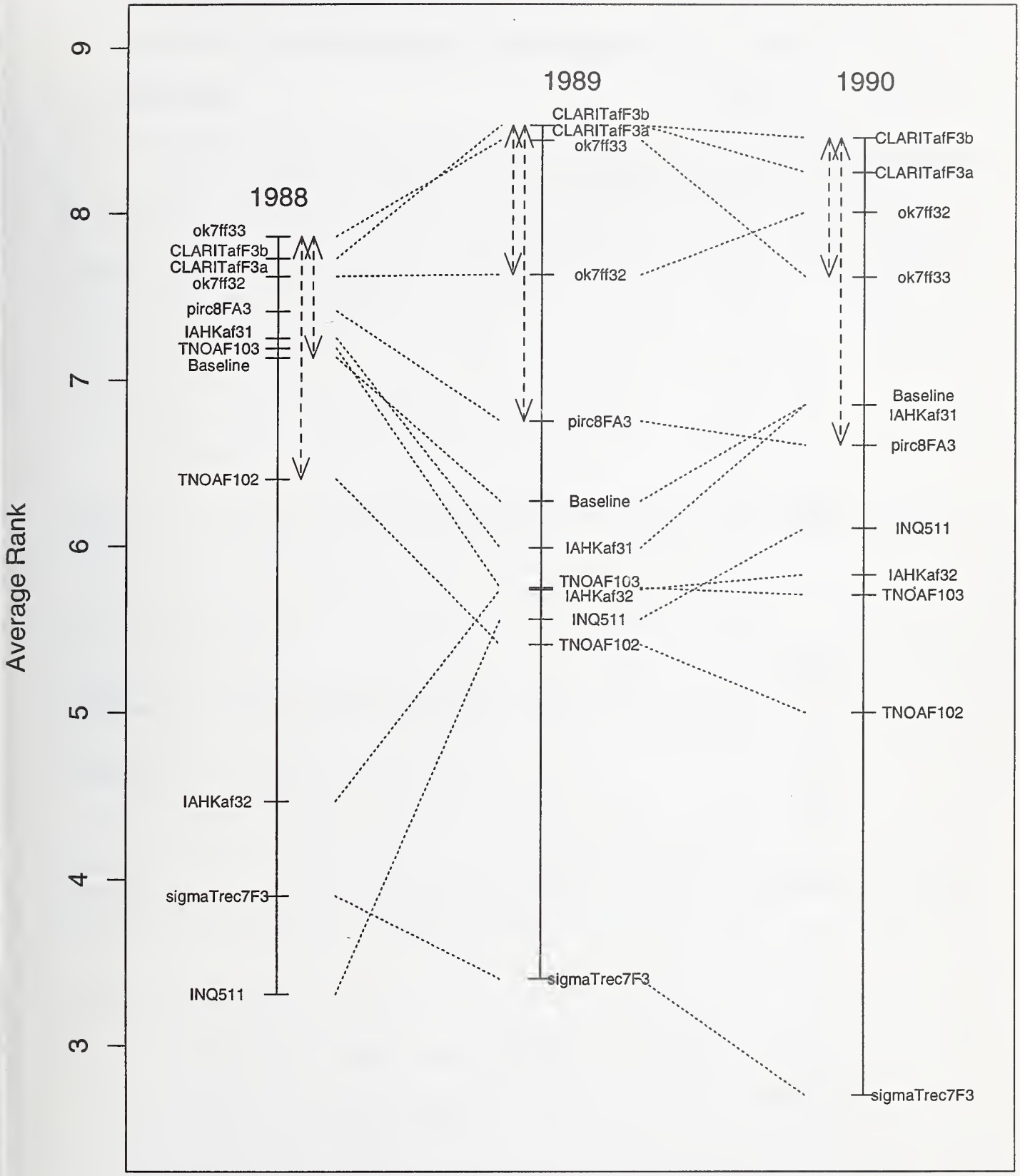
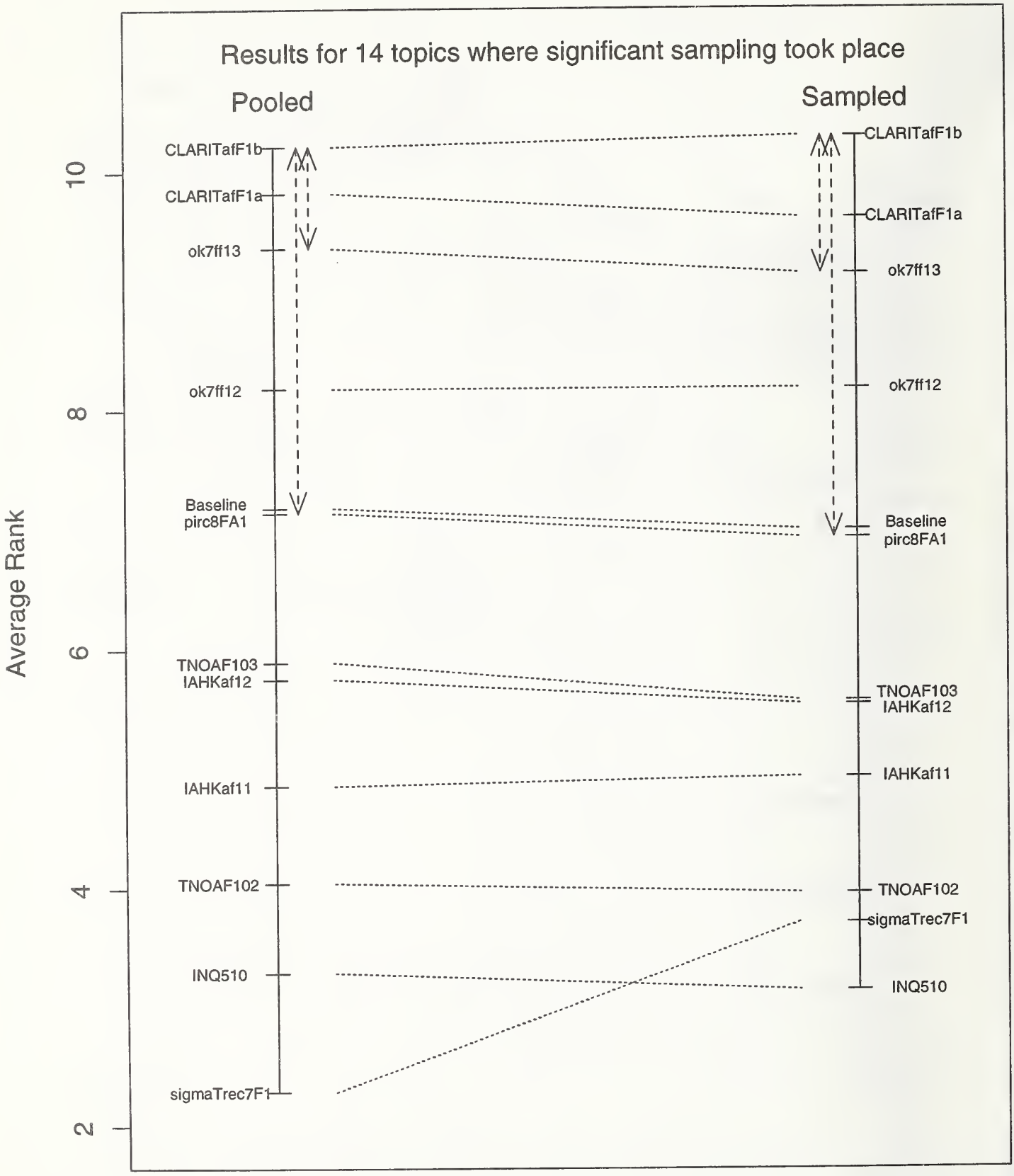


Figure 10 - Pooled vs. Sampled F1 Utility



The TREC 7 High Precision Track

Chris Buckley - SabIR Research, Inc.

Track Overview

TREC 7 is the second year the High-Precision (HP) track has been run. It is an attempt to perform a task that is much more closely related to real-world user interactions than the ad-hoc or routing task. The goal is simple: a user is asked to find 15 relevant documents in 5 minutes. No other restrictions are put on the user (other than no prior knowledge of the query, and no asking other users for help). Official evaluation is simply how many actual relevant documents were found among the 15 documents supplied by the user, modified slightly for those queries with fewer than 15 relevant documents in the collection (Relative Precision at 15 documents).

There are no restrictions on the type of resources the user may use during this task other than

- Only one user per query per run (no human collaboration).
- The user and system can have no previous information about the query (eg, the system cannot have previously built a query dependent data structure.)

In particular, the users are allowed to make multiple retrieval runs, allowed to look at documents, allowed to use whatever visualization tools the system has, and allowed to use system or collection-dependent thesauruses, as long as they stay within the 5 minute clock time.

This track tests (at least) the effectiveness, efficiency, and user interface of the systems. The task provides a forum for testing many of the neat ideas in user interface and visualization that have been suggested over the years.

Unlike other interactive evaluations (for example, the TREC 6 Interactive task), no attempt is made to factor out user differences when comparing across systems. All users are assumed to be experts and equally proficient in use of their own system. This allows for fair comparison of systems, but implies that the absolute level of performance within the track will be better than the level obtainable from casual users. These are upper-bound interactive experiments.

The only changes in the rules from the TREC 6 track are to raise the number of relevant documents required to 15 instead of 10, and to forbid cutting and pasting of the original query. This latter change requires the participants to type in the query, and makes the task fairer for those groups for whom cutting and pasting would not give a query in the proper form. It also has the side effect of making the task more difficult since reading and typing parts of the query might take 30 seconds (10% of the available time).

High-Precision Results

Four groups participated in the High-Precision track with a total of 7 runs.

- Cornell University/Sabir Research (3 runs)
- University of Waterloo (2 runs)
- Australian National University (1 run)
- CUNY-Queens: PIRCS system (1 run)

All four groups used basically the same retrieval approach as they used for their ad-hoc runs. In all cases, the majority of the user time was spent just judging documents that were sequentially given to the user by the system, with occasional query reformulation. All groups allowed the user to reformulate the query by hand; Cornell also used automatic relevance feedback to expand the query based on terms from seen relevant documents.

All four groups did quite well on the task. The CUNY group had a slight misunderstanding of the task that adversely affected their score.

Run	Precision	Relative Precision	Num relevant
Cor7HP3	.5853	.5967	439
Cor7HP2	.5813	.5920	436
Cor7HP1	.5787	.5909	434
uwmt7h1	.5693	.5772	427
uwmt7h2	.5373	.5467	403
acsys7hp	.5120	.5295	384
pircs8Ha	.4773	.4839	358

Agreements with TREC Assessors.

One important question is how the users agree with the official TREC relevance judgements. If the HP track is to have meaning, the disagreement between user interpretation of relevance to a query, and the official assessor interpretation can not dominate the results. Both Cornell and Waterloo studied how their judgements agreed with the official NIST judgements. The Waterloo high-precision runs were a subset of their manual ad-hoc runs and they didn't separate out their high-precision figures, but overall they agreed with the Cornell figures.

For the three Cornell runs, Table 1 gives the total number judged relevant, possibly relevant, and non-relevant for each user, for both the TREC-assessor judged relevant documents and the TREC-assessor judged non-relevant documents. For example, Cornell User 3 judged 290 documents (159+131) relevant or iffy that the official assessors had judged non-relevant.

Run	TREC judged Rel			TREC judged NonRel			Overlap
	UserRel	Iffy	NonRel	UserRel	Iffy	NonRel	(Iffy=rel)
Cor7HP1	315	170	51	79	181	448	61%
Cor7HP2	396	73	36	115	128	444	63%
Cor7HP3	374	100	84	159	131	674	56%

Table 1: Cornell High-Precision User-assessor consistency (50 queries)

The last column gives the overlap on judgements of relevant documents. Overlap is defined as the intersection of the relevant judgements divided by the union of the relevant judgements, and is the desired measure for how well judgements agree. (Early TREC studies used other measures that are dependent on the number of total number of documents judged. These measures might have been reasonable for those particular studies since the total number of documents was fixed, but the measures are *not* valid for general use, despite the fact that others have adopted them.)

The great majority of the disagreements are the users considering documents relevant that the assessor considered non-relevant. In fact, consider the 15 queries with lowest overlap for each of the three users; for all 45 queries the user has looser criteria than the assessor. This is to be expected, since the assessor as the originator of the query can easily have in mind a stricter query than made it to the topic description. For example, in query 375 "hydrogen energy", the assessor obviously did not want hydrogen fuel for car engines, though that wasn't clear from the topic. The three users marked

a total 50 documents as relevant or iffy that were not relevant. Query 363 "tunnel disasters" was another with major disagreements (36 documents).

The disagreements in the other direction are rarer and a bit less obvious. For example, query 377, "cigar smoking", had the most disagreements, with 15 total assessor relevant documents being marked non-relevant by the three users.

The overall level of disagreement between assessor and users is unfortunately high. The overall level of performance is being strongly affected by agreement with assessor, rather than intrinsic performance.

Difficulty of Task.

One of the ways of telling how easy or difficult the TREC 7 HP task is, is to look at the queries for which the users did not find 15 documents that they thought were relevant. Table 2 gives the number of documents that are included in the final submitted retrieval without being judged for the three Cornell runs. There will be unjudged documents only if the user did not find 15 relevant or iffy documents after 5 minutes.

Run	num docs unjudged	num queries with unjudged	num unjudged rel docs
Cor7HP1	122	24	12
Cor7HP2	139	25	20
Cor7HP3	84	17	13

Table 2: Cornell Unjudged Retrieved Documents

Half or less of the 50 queries have any unjudged documents at all for all three Cornell users. This includes queries for which there were fewer than 15 relevant documents in the collection. This implies for the majority of the queries, the only evaluation differences are due to disagreement with assessors rather than effectiveness of system. Combined with the high disagreement between users and assessors, the conclusion must be reached that the task is too easy.

Put another way, on average for Cornell User 2, of 15 documents returned per query

- 8.8 were agreed relevant
- 2.4 were agreed non-relevant
- 3.5 had relevance disagreements

The disagreements are more important than non-found documents. Again, this suggests the task was too easy and didn't stress the users and system enough.

Timing Evaluation.

Cornell, Waterloo, and ANU kept track of not only what each user document judgement was, but when it occurred (though ANU only has figures for 39 out of the 50 queries). Thus we can analyze the time performance of each user, and hopefully develop time-based evaluation measures that reflect the power and efficiency of systems.

The most obvious fact to look at is when the relevant documents were retrieved. Figure 1 gives the number of relevant documents retrieved during each 5 second timeslice for Cornell User 1, on average for 50 queries. The number of retrieved relevant starts off at 0 for the first 20 to 50 seconds as the user reads and types in the query. Then it steadily increases for the next minute or so and then starts slowly decreasing up until the 5 minute point is reached. There's a big hump at 300 seconds as the 15

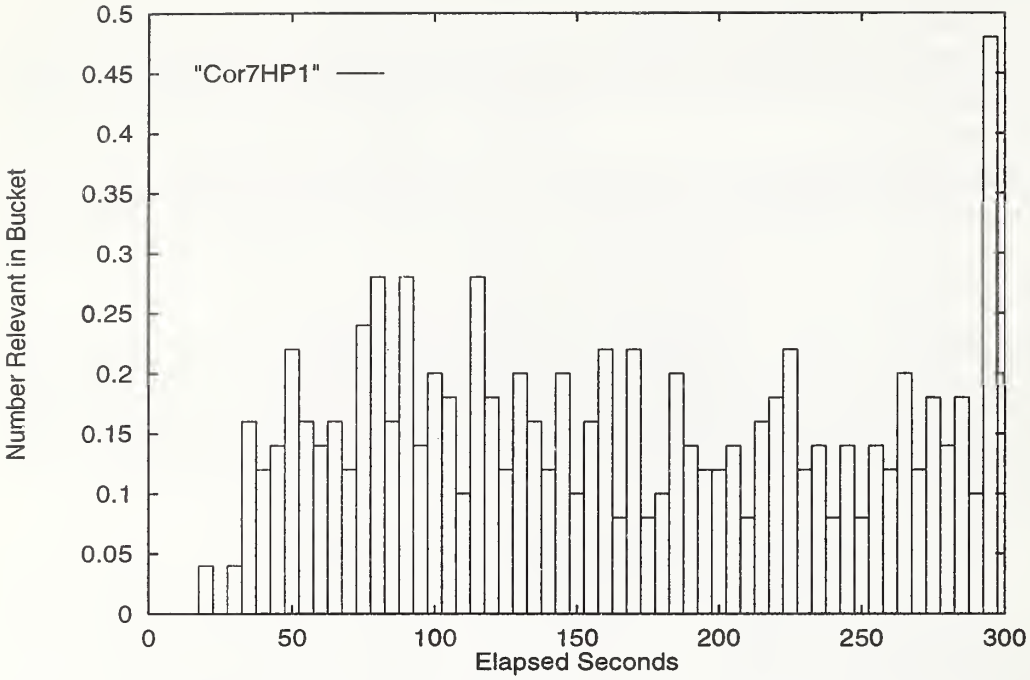


Figure 1: Average Relevant Retrieved per 5 second Timeslice over 50 Queries

documents to be returned get filled in with unjudged documents. In the normal course of retrieval , these documents would be judged over the next few buckets.

This graph is actually evidence against the conclusion reached earlier that the task was too easy. The rate at which relevant documents are being added close to 300 seconds is still substantial. The previous evidence indicates it can't go on for much longer, and that less than half of the queries are still active. However, there is no sudden drop-off as there would be if this particular run finds too many relevant documents.

Figure 2 compares the three Cornell users on a typical single query, Query 366. The measure being plotted is precision at 15 documents. More generally, the time-precision measure is defined as
Time-precision = num-relevant-retrieved-so-far / total-session-retrieved

User 2 typed in a shorter query so started judging documents earlier than the others. User 2 maintains a lead up until 180 seconds, when User 1 takes over. Then at 240 seconds, User 3 takes the lead for the last minute.

For this particular query, it is clear that User 3 has the best end result (precision after 5 minutes). But it is also clear that User 2 and possibly User 1 have better sessions: they find relevant documents sooner during the first 4 minutes.

Figure 3 gives the same comparison except on the average of all 50 queries. Once again, User 2 has the lead for most of the session up until the very end when User 3 takes over. For most of the session, User 2 is about 10 seconds ahead of User 3 and 20 seconds ahead of User 1. Again, User 3 has the best end result, but User 2 had the best session.

Other evaluation measures give the same overall results. For example, Unranked Average Precision at 15 documents is given in Figure 4. The curve is almost identical.

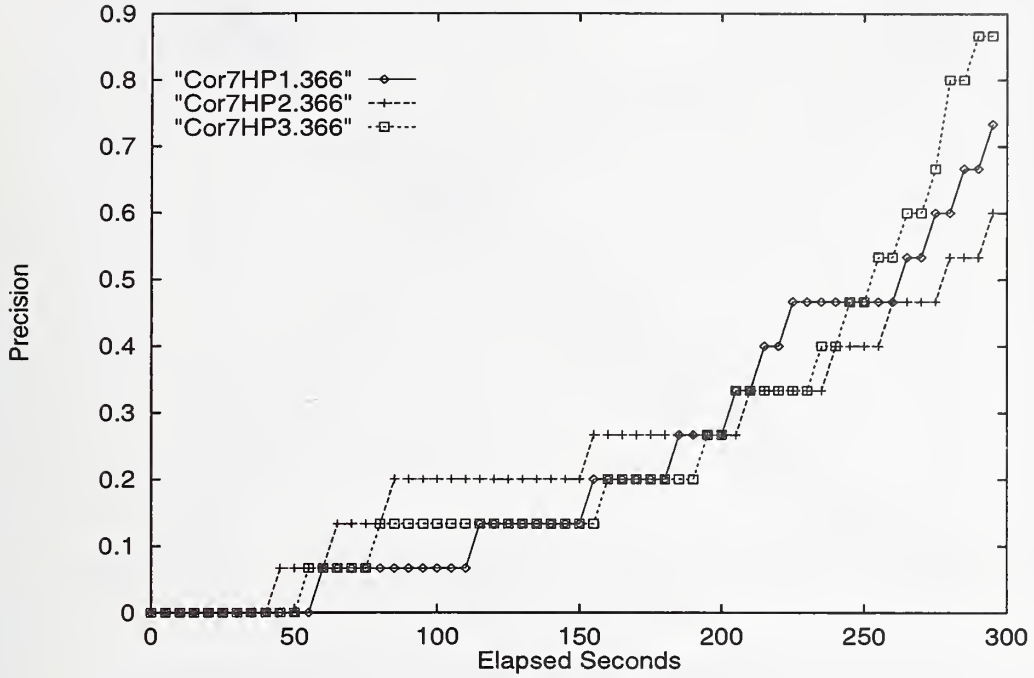


Figure 2: Time-Precision (at 15 Documents) vs. Time for Query 366

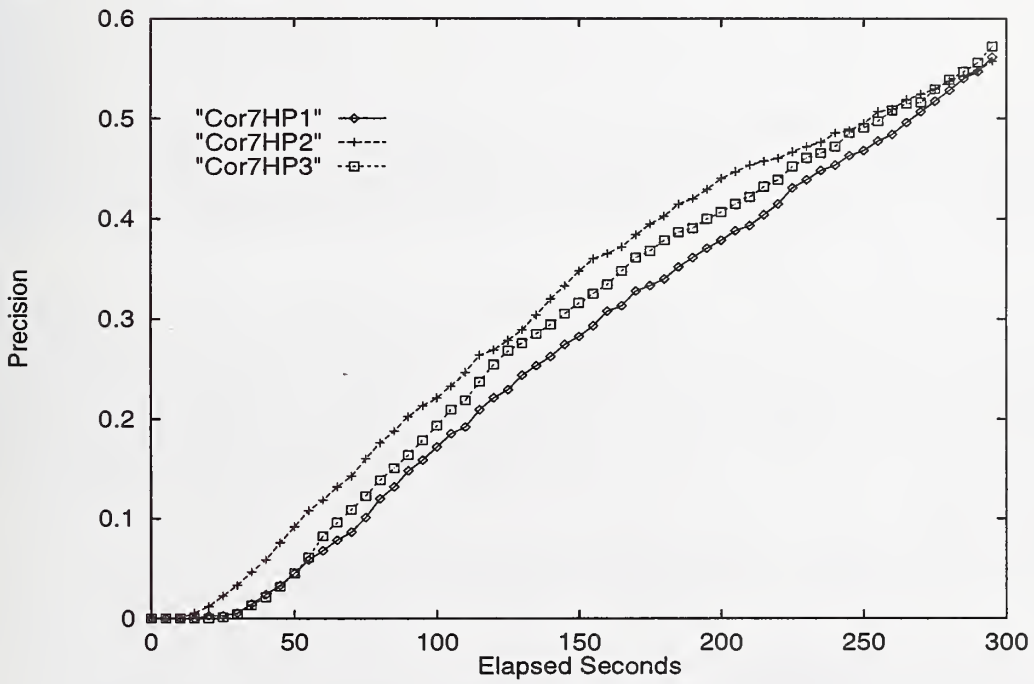


Figure 3: Time-Precision (at 15 Documents) vs. Time over 50 Queries

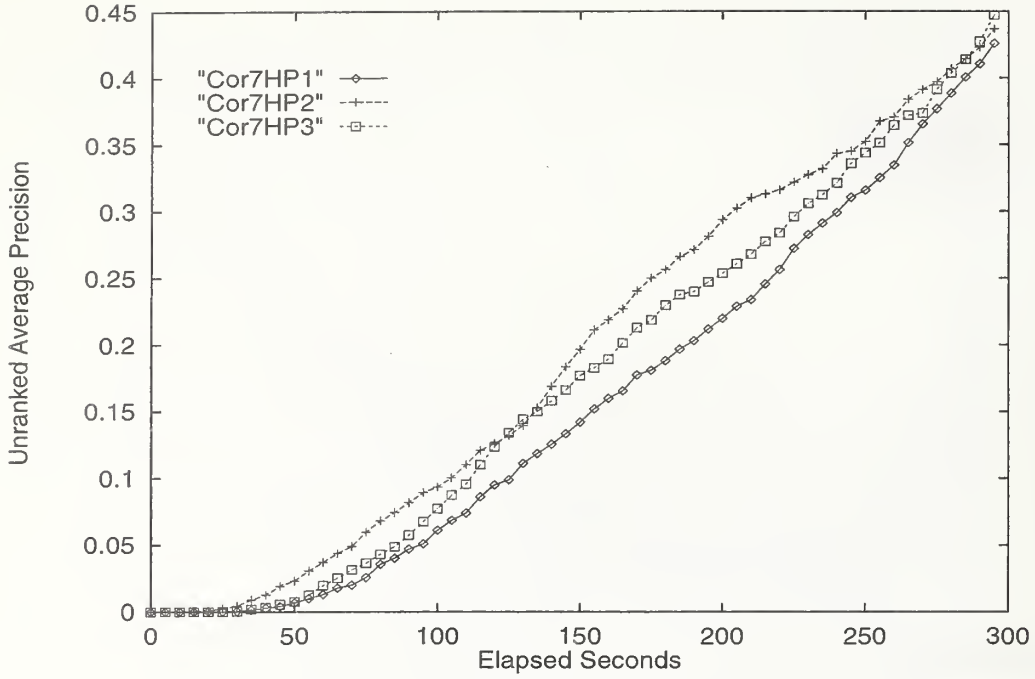


Figure 4: Unranked Average Precision vs. Time over 50 queries

One different evaluation measure is Utility(1,-1,0,0) in Figure 5. This measures increases by 1 when a relevant document is retrieved and decreases by 1 when a non-relevant document is retrieved. It is a poor evaluation measure for the HP task. It is dominated by the retrieved non-relevant documents; i.e., those documents for which user and assessor disagree on relevance. None-the-less, the results are informative.

User 2's lead is even more substantial (remember User 2 has the most accurate judgements as measured by agreement with assessors). But what is very interesting is how the plots for User 2 and User 3 flatten out over the last 2 minutes. For every relevant document being added, a non-relevant document is being added. This may indicate more disagreements occur late, or maybe there is a natural stopping spot late. Further study is needed, especially since the handling of "iffy" documents may be partly responsible for the effect.

All of these time-based measures and graphs suggest that a reasonable evaluation measure for an entire session is the area under each plot, much in the same way as the area under the recall-precision curve is a good single measure (this is "average precision"). Table 3 gives three such measures, corresponding to the three different plots seen above. As expected, for all 3 session measures, User 2 has a substantial (6% – 8%) lead over User 3 and even more over User 1. It can certainly be argued that this evaluation approach is a better approach than the official measure used for the track (precision after 5 minutes.)

Run	Average Precis	Average UAP	Average Utility(1,-1)
Cor7HP1	.2726	.1590	3.997
Cor7HP2	.3104	.1934	4.606
Cor7HP3	.2901	.1780	4.287

Table 3: Timing Evaluation

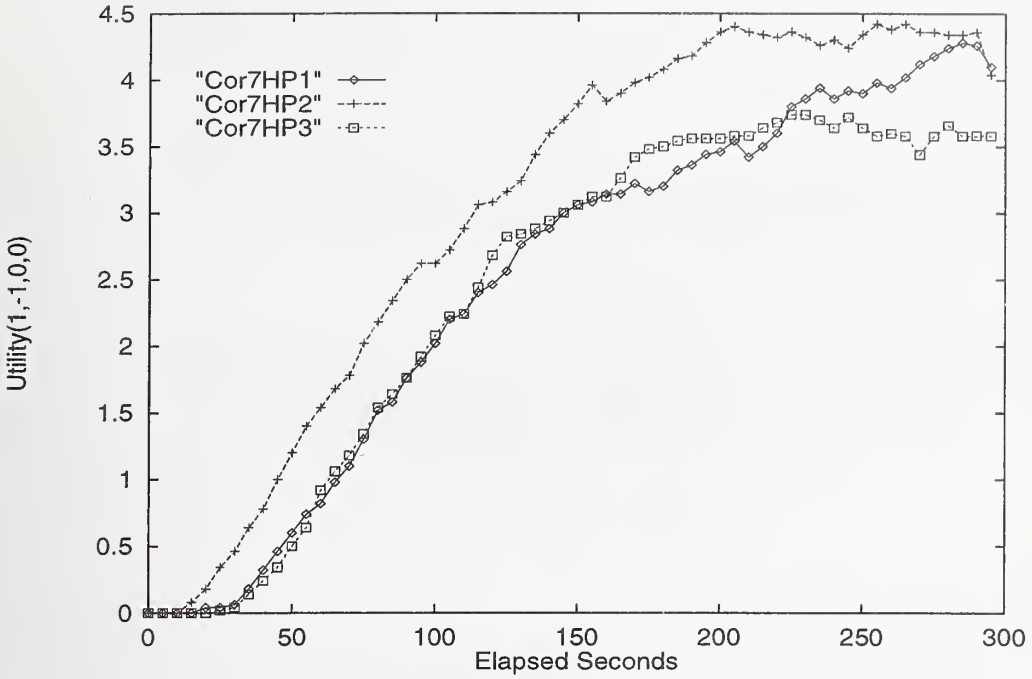


Figure 5: Utility(1,-1,0,0) vs. Time over 50 queries

These session evaluation measures can be extended to work on any time-based retrieval. It would be very interesting to apply these measures to the standard Manual portion of the ad-hoc task. Perhaps in the future, we can request that timing figures be optionally supplied, perhaps as the iteration field, to Manual submissions. There are still open questions regarding these measures. A couple that immediately spring to mind is how sensitive they are to starting time, and to size of time-slice. However, they still seem to offer a hope at bringing efficiency into evaluation of manual systems and sessions.

Note that the latest copy of trec_eval is in pub/smart/trec_eval.7.0beta.tar.gz on ftp.cs.cornell.edu and includes all the measures discussed here plus many others, though perhaps not in their final form (for instance, the timing information is assumed to be in the "sim" field but will probably be moved.)

Conclusion

The High-Precision track of TREC 8 was an attempt to evaluate a realistic user-oriented task, namely finding relevant documents quickly. All the groups did well, coming very close to the limits of performance that relevance judgement disagreements allow. This suggests that the task was probably too easy and didn't stress the user and systems enough.

It was unfortunate that no group used any really innovative user interface, such as looking at clustered retrieved documents. Instead, all groups took the approach of trying to judge as many good individual documents as quickly as possible. The experimental setup allows for much more interesting comparisons than were done this year.

After the experiment, we looked in-depth at methods of analyzing and evaluating time-dependent retrieval sessions. We came up with several new evaluation measures that seem to capture the essentials of what a session evaluation of manual retrieval should capture. These approaches may be quite useful outside of the High-Precision track, perhaps to evaluate timed Manual retrieval.

TREC-7 Interactive Track Report

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

July 12, 1999

Abstract

This report is an introduction to the work of the TREC-7 Interactive Track with its goal of investigating interactive information retrieval by examining the process as well as the results.

Eight research groups ran a total of 15 interactive information retrieval (IR) systems on a shared problem: a question-answering task, eight statements of information need, and a collection of 210,158 articles from the Financial Times of London 1991-1994.

This report summarizes the shared experimental framework, which for TREC-7 was designed to support analysis and comparison of system performance only within sites. The report refers the reader to separate discussions of the experiments performed by each participating group - their hypotheses, experimental systems, and results. The papers from each of the participating groups and the raw and evaluated results are available via the TREC home page (trec.nist.gov).

1 Introduction

For TREC-7 the high-level goal of the Interactive Track remained the investigation of searching as an interactive task by examining the process as well as the outcome. To this end a common experimental framework was designed with the following features:

- an interactive search task
- 8 topics - brief statements of information need
- a document collection to be searched
- a required set of searcher questionnaires
- a required psychometric test for all searchers
- 6 classes of data to be collected at each site and submitted to NIST
- 3 summary measures to be calculated by NIST for use by participants

The framework allowed groups to estimate the effect of their experimental manipulation free and clear of the main (additive) effects of participant and topic and it was designed to reduce the effect of interactions.

In TREC-7 the emphasis was on each group's exploration of different approaches to supporting the common searcher task and understanding the reasons for the results they get. No formal coordination of hypotheses or comparison of systems across sites was planned, but groups were encouraged to seek out and exploit synergies. Some groups designed/tailored their systems to optimize performance on the task; others simply used the task to exercise their system(s). Figure 1 lists the research groups that took part, their systems (control and experimental), and the number of searches performed on each. Here are the high-level issues addressed by each team:

- The researchers at New Mexico State University at Las Cruces investigated the benefit of a thumbnail-document view over a more conventional interface. (Ogden, Davis, & Rice, 1999)
- In London and Sheffield the Okapi Group made two pairwise comparisons: Okapi with relevance feedback versus Okapi without and Okapi without versus ZPRISE without. (Robertson, Walker, & Beaulieu, 1999)
- The team at Oregon Health Sciences University carried out a large-scale comparison of Boolean and natural language searching involving 28 searchers. (Hersh et al., 1999)
- The Royal Melbourne Institute of Technology group examined differences in retrieval coverage and efficiency resulting from different organizations of query results: a list of cluster descriptors versus a list of document titles. (Fuller et al., 1999)
- Researchers at Rutgers conducted a study to investigate the effectiveness and usability of a particular implementation of negative relevance feedback and of relevance feedback as a term-suggestion device. (Belkin et al., 1999)
- At the University of California at Berkeley they replicated their experiments from TREC-6 but with larger numbers of searchers/searches and more information about searchers and their search experiences gathered from the track questionnaires. (Gey, Jiang, Chen, & Larson, 1999)
- The University of North Carolina at Chapel Hill team tried to determine whether the ability to see and modify term weights improves retrieval effectiveness and whether the passage is a better unit of relevance feedback than the document. (Yang, Maglaughlin, Meho, & Sumner, 1999)
- The University of Toronto researchers compared an experimental system which blended querying and browsing with a system approximating a common web search engine system where querying is distinct from browsing the documents found. (Bodner & Chignell, 1999)

Groups	Systems	Searches
New Mexico State University at Las Cruces	J24	32
	ZP	32
Okapi Group	ok_noRF	32
	zp_noRF	32
	ok_noRF	32
	ok_withRF	32
Oregon Health Sciences University	MB	112
	MR	112
Royal Melbourne Institute of Technology	clus	64
	list	64
Rutgers University	RUINQ-G	67
	RUINQ-R	68
University of California at Berkeley	C	32
	Z	32
University of North Carolina at Chapel Hill	irisp	32
	irlss	32
	irlsa	32
	irlss	32
University of Toronto	a	32
	b	32

Figure 1: Patricipating research groups, their systems and the number of searches performed on each.

2 Method

2.1 Participants

Each research group selected its own experimental participants, known in what follows as "searchers." There was only one restriction: no searcher could have previously used either the control system or the experimental system. Additional restrictions were judged impractical given the difficulty of finding searchers. A minimum of eight searchers was required, but the experimental design allowed for the addition of more in groups of four and additions were encouraged. Standard demographic data about each searcher were collected by each site and some sites administered additional tests.

2.2 Apparatus

IR systems

In addition to running its experimental system(s), each participating site chose a control system appro-

priate to the local research goals.

Computing resources

Each participating group was responsible for its own computing resources adequate to run both the control and experimental systems and collect the data required for their own experiments and for submission to NIST. The control and the experimental systems were to be provided with equal computing resources within a site but not necessarily the same as those provided at other sites.

Topics

Eight of the 50 topics created by NIST for the TREC-7 adhoc task were selected and modified for use in the interactive track by adding a section called "Instances" and removing the "Narrative." The eight topics were entitled as follows:

- 352i British Chunnel impacts
- 353i antarctic exploration
- 357i territorial waters dispute
- 362i human smuggling
- 365i El Nino
- 366i commerical cyanide uses
- 387i radioactive waste
- 392i robotics

Each of the eight topics described a need for information of a particular type. Contained within the documents of the collection to be searched were multiple distinct examples or instances of the needed information. Here is an example interactive topic.

```
Number: 352i

Title:  British Chunnel impacts

Description:
Impacts of the Chunnel - anticipated
or actual - on the British economy
and/or the life style of the British

Instances:
In the time alloted, please find as
many DIFFERENT impacts of the sort
described above as you can. Please
save at least one document for EACH
such  DIFFERENT impact. If one
document discusses several such
impacts, then you need not save
other documents that repeat those,
since your goal is to identify as
many DIFFERENT impacts of the sort
described above as possible.
```

The results of test searches performed at NIST were used to:

- choose the eight topics from a larger set
- attempt to balance the blocks for difficulty
- attempt to define the sequence of use within each block so that the difficulty increased

Searcher task

The task of the interactive searcher was to save documents, which, taken together, contained as many different instances as possible of the type of information the topic expressed a need for - within a 15 minute time limit.

Searchers were encouraged to avoid saving documents which contribute no instances beyond those in documents already saved, but there was no scoring penalty for saving such documents and searchers were to be told that.

Document collection

The collection of documents to be searched was the Financial Times of London 1991-1994 collection (part of the TREC-7 adhoc collection). This collection contains 210,158 documents (articles) totaling 564 megabytes. The median number of terms per document is 316 and the mean is 412.7.

Searchers	System,Topic combinations (in Latin squares for evaluation)							
S1	E,T1	C,T5	E,T2	C,T6	E,T3	C,T7	E,T4	C,T8
S2	C,T5	E,T1	C,T6	E,T2	C,T7	E,T3	C,T8	E,T4
S3	E,T5	C,T1	E,T6	C,T2	E,T7	C,T3	E,T8	C,T4
S4	C,T1	E,T5	C,T2	E,T6	C,T3	E,T7	C,T4	E,T8

Figure 2: Half the minimal 8-searcher-by-8-topic matrix as evaluated. E = experimental system, C = control.

Searchers	System,Topic combinations (in the order seen by searchers)							
S1	E,T1	E,T2	E,T3	E,T4	C,T5	C,T6	C,T7	C,T8
S2	C,T5	C,T6	C,T7	C,T8	E,T1	E,T2	E,T3	E,T4
S3	E,T5	E,T6	E,T7	E,T8	C,T1	C,T2	C,T3	C,T4
S4	C,T1	C,T2	C,T3	C,T4	E,T5	E,T6	E,T7	E,T8

Figure 3: Half the minimal 8-searcher-by-8-topic matrix as run.

2.3 Procedure

Each searcher performed eight searches on the document collection using the eight interactive track topics. Each searcher performed half of the total number of searches on the site's experimental system and the other half on its control system. Instructions on the task preceded all searching and a system tutorial preceded the first use of each system. In addition, each searcher was asked to complete a questionnaire, prior to all searching, after each search, after the last search on a given system, and after all searching was complete. The detailed experimental design determined the order in which each searcher used the systems (experimental and control) and topics.

The minimal 8-searcher-by-8-topic matrix was constructed of 16 2-searcher-by-2-topic Latin squares. Figure 2 shows half of the required matrix; the other half is identical except it includes four additional searchers. Each 2-by-2 square has the property that the "treatment effect," here $E - C$, the control-adjusted response, can be estimated free and clear of the main (additive) effects of searcher and topic. Participant and topic are treated statistically as blocking factors. This means that even in the presence of the anticipated differences between searchers and topics, the designs provided estimates of $E - C$ that were not contaminated by these differences.

However, the estimate of $E - C$ would be contaminated by the presence of an interaction between topic and searcher. Therefore, we replicated the 2x2 Latin square 4x4 times to get the minimal 8x8 design for each site. The contaminating effect of the topic by searcher interaction was reduced by averaging the sixteen estimates of $E - C$ that are available, one for each 2x2 Latin square. This is analogous to averaging replicate measurements of a single quantity in order to reduce the measurement uncertainty. Each 2-by-2 square yields 1 within-searcher estimate of the $E - C$ difference for a total of 16 such estimates for each 8-searcher-by-8-topic matrix.

To reduce the searcher's cognitive load and possible confusion due to switching search systems with each search, the columns were permuted as indicated in Figure 3 for the running of the experiment.

In resolving experimental design questions not covered here (e.g., scheduling of tutorials and searches, etc.), participating sites were asked to minimize the differences between the conditions under which a given searcher used the control and those under which he or she used the experimental system.

2.4 Data submitted to NIST

Six sorts of data were collected for evaluation/analysis (for all searches unless other-

wise specified) and are available from the TREC-7 Interactive Track web page (www-nlpir.nist.gov/projects/t7i/t7i.html).

- sparse-format data - list of documents saved and the elapsed clock time for each search
- rich-format data - searcher input and significant events in the course of the interaction and their timing
- searcher questionnaires on background, user satifaction, etc.
- the results of the Educational Testing Service's FA-1 test (controlled associations)
- a full narrative description of one interactive session for topic 365i
- any further guidance or refinement of the task specification given to the searchers

Only the sparse-format data were evaluated at NIST to produce a triple for each search: instance precision and recall (these as defined in the next section) and elapsed clock time.

2.5 Evaluation of the sparse-format data submitted to NIST

Evaluation by NIST of the sparse-format data proceeded as follows. For each topic, a pool was formed containing the unique documents saved by at least one searcher for that topic regardless of site.

For each topic, the NIST assessor, normally the topic author, was asked to:

1. Read the topic carefully.
2. Read each of the documents from the pool for that topic and gradually:
   (a) Create a list of the instances found somewhere in the documents
   (b) Select and record a short phrase describing each instance found
   (c) Determine which documents contain which instances
   (d) Bracket each instance in the text of the document in which it was found

Then for each search (by a given searcher for a given topic at a given site), NIST used the submitted list of selected documents and the assessor's instance-document mapping for the topic to calculate:

- the fraction of total instances (as determined by the assessor) for the topic that are covered by the submitted documents (i.e., instance recall)
- the fraction of the submitted documents which contain one or more instances (i.e., instance precision)

The third measure, elapsed clock time, was taken directly from the submitted results for each search.

3 Results and Discussion

Since comparison of systems across sites is not supported by the experimental design, the reader is directed to the site reports in these proceedings or on the TREC web site (trec.nist.gov) for presentation and discussion of results in context of the local research goals.

The mean results by topic are presented here in Figure 4. Since thh order of topics was the same in all experiments, the effect of order is indistinguishable from that of topic. While the first topic in each block seems to have been easier than those that followed, it is not clear that the blocks are overall of equal difficulty, as was intended.

4 Author's note

The design of the TREC-7 Interactive Track matrix experiment grew out of the efforts of the many people who contributed to the discussion of ends and means on the track discussion list and through other channels. The author would like to acknowledge the special contributions of the track coordinators, Steve Robertson and Nick Belkin, of Bill Hersh, who coordinated the use of the FA-1 test.

Block	Order in block	Topic	Mean instance recall across all searcher-systems	Mean instance precision across all searcher-systems	Number of searches	Number of instances identified by NIST
1	1	365i	0.750	0.893	117	24
	2	357i	0.257	0.437	117	13
	3	362i	0.259	0.632	117	12
	4	352i	0.248	0.673	117	28
2	1	366i	0.375	0.835	117	7
	2	392i	0.324	0.692	117	36
	3	387i	0.375	0.778	117	9
	4	353i	0.187	0.409	116	11

Figure 4: Results by topic.

References

Belkin, N. J., Perez Carballo, J., Cool, C., Kelly, D., Lin, S., Park, S. Y., Rieh, S. Y., Svage-Knepshield, P., & Sokora, C. (1999). Rutgers' TREC-7 Interactive Track Experience. In E. M. Voorhees & D. K. Harman (Eds.), *The Seventh Text REtrieval Conference (TREC-7).* Gaithersburg, MD, USA.

Bodner, R. C., & Chignell, M. H. (1999). ClickIR: Text Retrieval using a Dynamic Hypertext Interface. In E. M. Voorhees & D. K. Harman (Eds.), *The Seventh Text REtrieval Conference (TREC-7).* Gaithersburg, MD, USA.

Fuller, M., Kaszkiel, M., Kim, D., Ng, C., Robertson, J., Wilkinson, R., Wu, M., & Zobel, J. (1999). TREC 7 Ad Hoc, Speech, and Interactive Tracks. In E. M. Voorhees & D. K. Harman (Eds.), *The Seventh Text REtrieval Conference (TREC-7).* Gaithersburg, MD, USA.

Gey, F., Jiang, H., Chen, A., & Larson, R. R. (1999). Manual Queries and Machine Translation in Cross-language Retrieval and Interactive Retrieval with Cheshire II at TREC-7. In E. M. Voorhees & D. K. Harman (Eds.), *The Seventh Text REtrieval Conference (TREC-7).* Gaithersburg, MD, USA.

Hersh, W., Price, S., Kraemer, D., Chan, B., Sacherek, L., & Olson, D. (1999). A large-scale comparison of boolean vs. natural-language searching for the TREC-7 interactive track. In E. M. Voorhees & D. K. Harman (Eds.), *The Seventh Text REtrieval Conference (TREC-7).* Gaithersburg, MD, USA.

Ogden, W., Davis, M., & Rice, S. (1999). Document thumbnail visualizations for rapid relevance judgements: When do they pay off? In E. M. Voorhees & D. K. Harman (Eds.), *The Seventh Text REtrieval Conference (TREC-7).* Gaithersburg, MD, USA.

Robertson, S. E., Walker, S., & Beaulieu, M. (1999). Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive track. In E. M. Voorhees & D. K. Harman (Eds.), *The Seventh Text REtrieval Conference (TREC-7).* Gaithersburg, MD, USA.

Yang, K., Maglaughlin, K., Meho, L., & Sumner, R. G., Jr. (1999). IRIS at TREC-7. In E. M. Voorhees & D. K. Harman (Eds.), *The Seventh Text REtrieval Conference (TREC-7).* Gaithersburg, MD, USA.

5 Appendix: Instructions to be given to each searcher

The following introductory instructions are to be given once to each searcher before the first search:

> Imagine that you have just returned from a visit to your doctor during which it was discovered that you are suffering from high blood pressure. The doctor suggests that you take a new experimental drug, but you wonder what alternative treatments are currently available. You decide to investigate the literature on your own to satisfy your need for information about what different alternatives are available to you for high blood pressure treatment. You really need only one document for each of the different treatments for high blood pressure.
>
> You find and save a single document that lists four treatment drugs. Then you find and save another two documents that each discusses a separate alternative treatment: one that discusses the use of calcium and one that talks about regular exercise. You've run out of time and stop your search. In all, you have identified six different instances of alternative treatments in three documents.
>
> In this experiment, you will face a similar task. You will be presented with several descriptions of needed information on a number of topics. In each case there can be multiple examples or instances of the type of information that's needed.
>
> We would like you to identify as many different instances as you can of the needed information for each topic that will be presented to you - as many as you can in the 15 minutes you will be given to search. Please save one document for EACH DIFFERENT instance of the needed information that you identify. If you save one document that contains several instances, try not to save additional documents that contain ONLY those instances. However, you will not be penalized if you save documents unnecessarily.
>
> As you identify an instance of the needed information, please keep track of which instances you have found: write down a word or short phrase to identify the instance, or–if the system provides a facility to keep track of instances–use it.
>
> Carefully read each topic to understand the type of information needed. This will vary from topic to topic. On one topic you may be looking for instances of a certain kind of event. On another you may be searching for examples of certain sorts of people, places, or things.
>
> Do you have any questions about
>
> - what we mean by instances of needed information,
> - the way in which you are to save nonredundant documents for each instance?

The TREC 7 Query Track

Chris Buckley - SabIR Research, Inc.

Introduction

General IR research is being held up because we don't have enough queries of various types to investigate advanced retrieval techniques that are query dependent. There's no way we can get enough relevance judgements on new queries to form a good query pool. The Query track looks at multiple query variations of past TREC topics to get a large number of query formulations.

The track guideline states four goals:

1. Start investigating the split between query formation/analysis and back-end engines. Evaluating what makes a good general query formation approach.
2. Get many variations of the same topic so we can start analyzing (including with strong NLP approaches) queries, and determining what sorts of things we want to pull out of queries.
3. Get a collection of mixed fact/content queries. For decades we've had systems (eg Pnorm) that can handle these, but haven't been able to evaluate and compare due to lack of a query collection.
4. Get a collection of reasonable very short queries, more typical of real-life ad-hoc queries.

Query Track Task

Each group forms variations of each of the 50 topics in some subsets of the following categories (as defined in the guidelines):

1. Very short: (2-3 words) based on topic.
2. Sentence: NL (natural Language), based on topic and judgements
3. Manual Feedback: Manual NL sentence based on reading 5 or so relevant documents without reference to the topic (done by someone who doesn't have the topics memorized and who might use different vocabulary than the topic). An attempt to get a sentence which might use different vocabulary than the topic.
4. Manual structured query: based on topics and judgements. Perhaps mixed fact and content queries. Perhaps result of manual NL analysis.
5. Automatic structured query: based on topics and judgements (Note that "structure" could be just a list of words, or could be very complicated based on semantics.) Perhaps the result of automatic NL analysis.

Then all groups run everybody's queries for some subset of the categories above (whatever categories their system can be made to support). The names of the submitted runs consist of 7-8 letters/digits. The first 3 letters identify the group running the query. The last 4-5 letters are the queryset id, including category. Thus, "CorAPL5a" would be Cornell running the first Category 5 query set that was constructed by APL.

Query Track Runs

This was the first year for the query track. As it ended up, only two groups participated in the track. Thus it is impossible to come up with as many conclusions as we had wanted.

The two groups are Cornell/SabIR and the APL Labs at Johns Hopkins. Cornell constructed one set of queries in each of the 5 categories; pretty much directly using the definitions of the categories. APL constructed 4 query sets, skipping category 3 and 4, but having two versions of category 5. For the first two categories, APL deliberately tried to construct different queries than the obvious choice of words. This increased query variability, though at a cost of overall effectiveness as we will see later.

All 5 sets of queries were reasonably easy to construct. Cornell's category 4 queries do not have much detailed structure; they are basically a weighted sum of a vector query and a pnorm query. Cornell's category 5 queries are straight weighted relevance feedback vectors.

The queries were all constructed in DN2 format. DN2 is a quite complicated query language, but luckily very few features needed to be known for the queries the two groups constructed. We did not run directly on the DN2 queries but translated them back and forth from normal TREC queries. In the future, it is clear we should use standard TREC form as much as possible. The DN2 format scared several groups away who might have participated.

Query Examples

As an example of variability of the queries, here are all the different forms of topic 4, expressed in DN2 format.

```
<DN2 ID=4 QUERYSET=APL1a>
  "Foreign debt reorganization"
</DN2>

<DN2 ID=4 QUERYSET=Cor1>
"debt rescheduling agreements"
</DN2>

<DN2 ID=4 QUERYSET=APL2a>
  "What countries have received assistance in the form of a
reduction in the rate at which they must repay their loans?"
</DN2>

<DN2 ID=4 QUERYSET=Cor2>
"debt rescheduling agreements and loan restructuring
accords between debtor countries and the EC, Paris Club
and creditor banks"
</DN2>

<DN2 ID=4 QUERYSET=Cor3>
"What restructuring of debt repayment by third-world
countries have creditor nations accepted?"
</DN2>

<DN2 ID=004 QUERYSET=APL5a>
  <INDEPENDENT>
```

```
    <FULL_TERM WEIGHT=1.000000> "creditor"  </FULL_TERM>
    <FULL_TERM WEIGHT=0.839627> "debtor"    </FULL_TERM>
    <FULL_TERM WEIGHT=0.695880> "rescheduling"</FULL_TERM>
    <FULL_TERM WEIGHT=0.692837> "debt"      </FULL_TERM>
    ...
  </INDEPENDENT>
</DN2>

<DN2 ID=004 QUERYSET=APL5b>
  <INDEPENDENT>
    <FULL_TERM WEIGHT=1.000000> "creditor"  </FULL_TERM>
    <FULL_TERM WEIGHT=0.639725> "debt"      </FULL_TERM>
    <FULL_TERM WEIGHT=0.596154> "billion"   </FULL_TERM>
    <FULL_TERM WEIGHT=0.556568> "nobrega"   </FULL_TERM>
    <FULL_TERM WEIGHT=0.556568> "mailson"   </FULL_TERM>
    ...
  </INDEPENDENT>
</DN2>

<DN2 ID=4 QUERYSET=Cor5>
  <INDEPENDENT>
    <FULL_TERM weight=0.1142> "repayers"  </FULL_TERM>
    <FULL_TERM weight=0.1311> "brazil"    </FULL_TERM>
    <FULL_TERM weight=0.2155> "paris"     </FULL_TERM>
    <FULL_TERM weight=0.2056> "accordance"</FULL_TERM>
    ...
  </INDEPENDENT>
</DN2>

<DN2 ID=4 QUERYSET=Cor4>
  <INDEPENDENT>
    <INDEPENDENT weight=0.7>
      <FULL_TERM weight=0.6756> "rescheduled"  </FULL_TERM>
      <FULL_TERM weight=0.2056> "accordance"   </FULL_TERM>
      <FULL_TERM weight=0.1844> "pact"         </FULL_TERM>
      <FULL_TERM weight=0.1764> "agreement"    </FULL_TERM>
      <FULL_TERM weight=0.1592> "debt"         </FULL_TERM>
      <FULL_TERM weight=0.1359> "restructuring"</FULL_TERM>
        ...
    </INDEPENDENT>
    <AND weight=0.3>
      <OR> "debt" "interest" "loan" "repayment" </OR>
      <OR> "rescheduling" "restructuring" </OR>
      <OR> "agreement" "accord" "settlement" "pact" "talks"
           "propose" "negotiate" "request" "grant" </OR>
    </AND>
  </INDEPENDENT>
</DN2>
```

Query Track Results

Table 1 gives results on running the 9 query set variations (5 variations from Cornell and 4 from APL) on the test document collection (TREC Disk 1 plus the AP subcollection from Disk 3). The runs all strongly differ from each other in results. In general, the Cornell queries performed better for Cornell than the APL queries. Part of that is that goals of the APL queries were explicitly to use different, possibly non-optimal, vocabulary. But part of it could be that Cornell constructed queries to suit Cornell's system. In particular, the query set Cor5 was constructed using relevance feedback based on Cornell document weights. How well these weights suit other systems remains to be seen. We didn't have enough participating systems to be able to conclude anything.

	APL		Cornell/SabIR	
Query Set	P(20)	Ave Prec	P(20)	Ave Prec
APL1a	.1460	.0559	.2350	.1051
APL2a	.1230	.0477	.2710	.1142
APL5a	.3010	.1627	.4010	.1971
APL5b	.5480	.2577	.6450	.3219
Cor1	.2730	.1055	.5030	.2457
Cor2	.4290	.1846	.6040	.3367
Cor3	.2330	.0917	.4560	.2020
Cor4	—	—	.6500	.3282
Cor5	.4540	.2296	.7760	.4586

Table 1: Results of Cornell and APL on Different Query Sets

As normal, even with the very strong overall differences in results between query sets, large numbers of individual queries of the weaker query set do better than the corresponding query in the stronger set. Table 2 gives the number of queries (out of 50) for which one query set beats another, keeping the system constant (Cornell's system was used). For instance, APL5b beat Cor2 on 28 out of 50 queries, despite having weaker overall evaluation averages.

>	Cor1	Cor2	Cor3	Cor4	Cor5	APL1a	APL2a	APL5a	APL5b
Cor1	0	7	32	11	2	43	39	30	18
Cor2	43	0	46	23	4	48	47	43	22
Cor3	18	4	0	5	1	38	36	26	12
Cor4	39	27	45	0	8	48	47	41	23
Cor5	48	46	49	42	0	50	49	48	46
APL1a	6	2	11	2	0	0	27	9	2
APL2a	11	3	14	3	1	22	0	16	3
APL5a	20	7	24	9	2	40	32	0	15
APL5b	32	28	38	27	4	48	47	35	0

Table 2: Comparative Query (row better than column for X queries)

There is a tremendous amount of query variability hidden in the comparative averages. We need to understand this variability. It is not clear that 9 query variations is enough to get a handle on variability; but at least it is a start.

Query Track Conclusions

It is impossible to conclude much from this initial track attempt since there were only two participants. We can verify what we already knew about queries:

- Different formulations of the same query can behave tremendously differently. In general, the more information included in the query, the better the results.
- Different queries behave very differently. There are significant numbers of queries where more information hurts.

We simply do not have enough information to look at how different systems interact with the various forms of the queries. Many interesting questions remain to be tackled in next year's track!

1998 TREC-7 Spoken Document Retrieval Track Overview and Results

John S. Garofolo, Ellen M. Voorhees, Cedric G. P. Auzanne, Vincent M. Stanford, Bruce A. Lund

National Institute of Standards and Technology (NIST)
Information Technology Laboratory
Building 225, Room A-216
Gaithersburg, MD 20899

ABSTRACT

This paper describes the 1998 TREC-7 Spoken Document Retrieval (SDR) Track which implemented an evaluation of retrieval of broadcast news excerpts using a combination of automatic speech recognition and information retrieval technologies. The motivations behind the SDR Track and background regarding its development and implementation are discussed. The SDR evaluation collection and topics are described and summaries and analyses of the results of the track are presented. Alternative metrics for automatic speech recognition as applicable to retrieval applications are also explored. Finally, plans for future SDR tracks are described.

1. BACKGROUND

Spoken Document Retrieval (SDR) involves the search and retrieval of excerpts from recordings of speech using a combination of automatic speech recognition and information retrieval techniques. In performing SDR, a speech recognition engine is applied to an audio input stream and generates a time-marked textual representation (transcription) of the speech. The transcription is then indexed and may be searched using an information retrieval engine. In traditional information retrieval, a topic (or query) results in a rank-ordered list of documents. In SDR, a topic results in a rank-ordered list of temporal pointers to potentially relevant excerpts. In an operational SDR system, these excerpts could be topical sections of a recording of a conference or radio or television broadcasts.

SDR was chosen as a TREC domain because of its potential use in navigating large multi-media collections of the near future and because it was believed that the component Automatic Speech Recognition and Information Retrieval technologies might work well enough now for usable SDR in some domains. SDR also provides a rich research domain in that it supports both development of large-scale near-real-time continuous speech recognition technologies and technologies for retrieval of spoken language. Further, SDR provides a venue for synergy between the speech recognition and information retrieval communities to improve both technologies and create hybrids.

The first community-wide evaluation SDR technology was implemented in 1997 for TREC-6. This pilot evaluation implemented a "known-item" task in which a particular relevant document was to be retrieved for each of a set of queries over a 50-hour collection of radio and television news broadcasts. Three retrieval conditions were implemented to examine the effect of recognition performance on retrieval performance:

1. *Reference* - retrieval using human-generated reference transcripts which for the purposes of this evaluation were considered to have "perfect" recognition.

2. *Baseline* - retrieval using IBM-contributed recognizer-generated transcripts with a 50% Word Error Rate. This provided both a common recognition error condition and an entrée for sites which did not have access to a recognition system of their own.

3. *Speech* - retrieval using the recordings of the broadcasts themselves requiring both recognition and retrieval technologies.

Thirteen sites participated in the pilot evaluation, eight of which implemented the Speech retrieval condition using their own or a team site's speech recognition system. The pilot evaluation proved that an evaluation of SDR technology could be implemented and that existing technologies worked quite well for a known-item task on a small collection. The results were so good that NIST chose to highlight the percent of target stories which were top-ranked (retrieved at rank one) by the systems.

Using the Percent Retrieved at Rank 1 metric, the University of Massachusetts retrieval system yielded the best performance for all three conditions. The UMass system achieved a retrieval rate of 78.7% for the Reference Retrieval condition and 63.8% for the Baseline Retrieval condition. For the Full SDR condition, UMass using a Dragon-Systems-produced 1-best recognizer transcript

with a 35% Word Error Rate, achieved a 76.6% retrieval rate.[1] The 2.1% difference in performance between retrieval using the reference transcripts and retrieval using the Dragon recognizer transcripts represented only one unretrieved story out of the 49 test topics.

2. MOTIVATION

The 1998 SDR Track was designed to address the known inadequacies in the 1997 SDR Track (small corpus, known-item task) to provide a more realistically challenging retrieval task. For 1998, an approximately 100-hour broadcast news test set collected by the Linguistic Data Consortium (LDC)[2], used previously as "the second 100 hours of BN training for Hub-4 recognition systems", was selected and a traditional TREC ad-hoc-style relevance task was chosen with topics and relevance assessments generated by human assessors. Two recognizer-produced transcript sets with different word error rates were provided by NIST as well as LDC human-generated reference transcripts. Also, for the first time, sites were encouraged to contribute their one-best recognizer-produced transcripts so that other sites could run retrieval on them. The improved test paradigm and alternative transcription sets with a spectrum of recognition error rates permitted us to further examine the relationship between recognition errors and retrieval accuracy. The new cross-recognizer task also permitted us to explore the development of alternative metrics for automatic speech recognition technology which would address particular inadequacies of the technology with regard to its use in information retrieval applications.

3. SDR EVALUATION PLAN

The complete evaluation plan for the 1998 TREC-7 Spoken Document Retrieval Track can be found at:

http://www.nist.gov/speech/sdr98/sdr98.htm

3.1 Evaluation Modes

The SDR Track included four retrieval conditions which provided component control experiments:

Reference (R1) (required) – Retrieval using the "perfect" human-transcribed reference transcripts of the Broadcast News recordings. This condition provided a control for retrieval.

Baseline (B1/B2) (required) – Retrieval using two sets of speech-recognition-generated 1-best transcripts produced by NIST using the CMU SPHINX-III recognition system. The Baseline-1 (B1) transcripts contained a moderate (33.8%) word error rate (relative to the current state-of-the-art) and the Baseline-2 (B2) transcripts contained a substantially higher (46.6%) word error rate. This condition provided two controls for recognition and permitted sites without access to recognition technology to participate.

Speech (S1/S2) (optional) – Retrieval using the Broadcast News recordings. This condition required both speech recognition and retrieval (which could be implemented by different sites). Two recognition/retrieval runs were permitted.

Cross Recognizer (CR) (optional) - Retrieval using 1-best speech-recognizer-generated transcripts contributed by other sites. This condition provided a control for recognition as well as allowing us to evaluate retrieval using a variety of recognition systems with a range of error rates.

One of the goals of the SDR Track is to encourage broad participation from both the Speech Recognition and Information Retrieval Communities. Therefore, the evaluation plan was designed to allow relatively easy entry for members of both communities. Speech recognition and retrieval experts were encouraged to team up to create pipelined or hybrid SDR systems. In addition, two participation levels were created to allow involvement by retrieval sites which did not have access to a speech recognition system:

Quasi-SDR - Sites without access to speech recognition technology were permitted to run retrieval on only the baseline recognizer transcripts and reference transcripts. (Retrieval conditions R1, B1, B2 minimally)

Full-SDR - Sites with access to speech recognition systems implemented both recognition and retrieval on the recorded news broadcasts as well as retrieval alone on the baseline recognizer transcripts and reference transcripts. (Retrieval conditions R1, B1, B2, and S1 minimally)

Participants in Full SDR with 1-best word-based recognizers were encouraged to submit their recognized transcripts to NIST. This provided the material to be used by other sites in implementing the Cross-Recognizer retrieval condition and permitted NIST to evaluate the effect of recognition error rates on retrieval performance.

For purposes of simplifying the implementation and evaluation process, the hand-annotated temporal story boundaries were given in all conditions. Figure 1. shows the general process for the TREC SDR task.

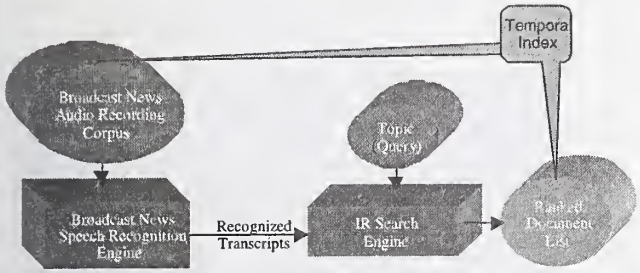


Figure 1. TREC SDR Process

3.2 Test Corpora

The LDC Broadcast News corpus was chosen for the SDR task since it contained news data from several radio and television sources and was fully transcribed and pre-segmented by story.[2] To adapt the BN corpus to the SDR task, story ID tags were added to uniquely identify each annotated story for retrieval and scoring.

A subset of 100 hours of the Broadcast News Corpus collected between June 1997 and January 1998 (which was originally collected by the LDC to provide training material for DARPA Hub-4 speech recognition systems) was chosen as the test corpus. The corpus was filtered to exclude commercials, sports summaries, weather reports, and untranscribed stories. In all, 87 hours of the 100-hour subset were selected as the test collection for the SDR evaluation. Because the story boundaries were to be known, an index giving the story IDs and time of each story boundary was provided to test participants.

The final filtered test set contained 2,866 stories with about 772,000 words. Roughly 1/3 of the stories in the test set were labeled as "filler" – non-topical sections of the broadcasts. Because of the small size of the collection for retrieval testing, these were not removed from the test set. The mean length in words for the stories in the test set was 269 words. The histogram in Figure 2 shows the distribution of the length of the stories in the test set. Note that about half of the stories contain less than 100 words and a few stories contain 2000 or more words.

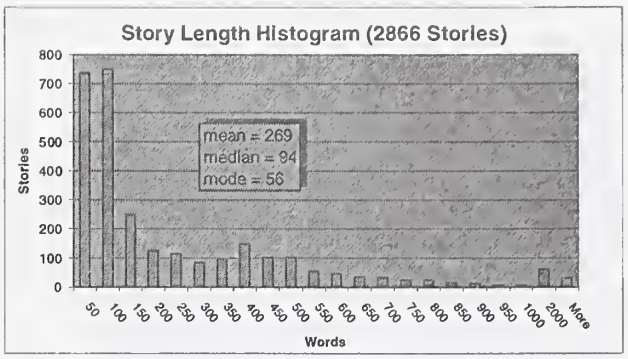


Figure 2. Test Collection Story Length Histogram

The recorded waveform material for the Speech retrieval condition was made available to the participants in April, 1998. The human-created reference transcripts for the test collection and indices which specified the 2,866 usable stories were released in June. The test topics and baseline recognizer transcripts were released in the beginning of July and results were due at NIST at the end of August. The results of the SDR track were reported at TREC-7 in November 1998 and at the DARPA Broadcast News Workshop in March 1999.

3.3 Baseline Recognizer Transcripts

CMU permitted NIST to use its SPHINX-III broadcast news recognition system to create a set of recognition-generated transcripts for the baseline retrieval (B1) condition. Since SPHINX-III ran in nearly 200 times real time on NIST's UNIX-based workstations, NIST realized that it would take almost two years of computation to complete one recognition pass over the 87 hours of recordings in the SDR test collection. NIST learned of inexpensive clusters of PC-based LINUX systems being used by NASA in its BEOWULF [3] project and set out to create such a system for recognition so that it could parallelize the recognition task.

An architecture was created in which a single scheduling server was used to control 8 computation nodes each containing 200-MHz Pentium Pro processors with 256 Mb. of memory and a 1 Gb. disk drive for swapping. The nodes were set up to boot from the server and both the server and nodes used the LINUX 2.0.32 operating system.

To implement distributed recognition, a CMU-contributed segmenter was first run on the recordings to break them up into tractable chunks of about 45 seconds each for recognition. A network scheduler using a FIFO-with-priorities algorithm (GNQS) was used to queue and track the chunks for processing over the available nodes.[4] The scheduling and network overhead was relatively low, so

the cluster performed roughly 8 times faster than a single processor machine.

The NIST High Performance Systems and Services (HPSS) Division was also investigating the use of such clusters as an alternative to supercomputers in servicing the computational needs of the NIST measurement laboratories and allowed us to enlist their nodes. This gave us access to 32 additional nodes and permitted HPSS to measure the performance of the technology. In all, 40 nodes were employed to create the B1 transcripts.

With 40 nodes, NIST was able to implement 2 baseline recognition runs. The first (B1) run was implemented with SPHINX running at moderate accuracy, using only the forward Viterbi search. This system benchmarked with the NIST SCLITE scoring software at 27.1% word error rate on the Hub-4 '97 test set and at 33.8% word error rate on the SDR test collection. NIST decided to create a second, less optimal run to examine the effect of recognition degradation on retrieval performance. The second (B2) run implemented the same SPHINX system, but with its pruning thresholds lowered. This system benchmarked at 46.6% word error rate on the SDR '98 test collection (comparable to the 50% word error rate for the IBM recognition system used to create the baseline recognizer transcripts in the 1997 SDR evaluation. [1])

3.4 SDR Topics

A team of 3 NIST TREC assessors met in April 1998 to select 25 topics for the test collection using similar procedures to those used in other TREC ad-hoc tasks. The assessors were instructed to find topics with 7 or more relevant news stories each in the collection using the NIST PRISE search engine. Unlike 1997, the assessors were not instructed to artificially construct the topics to exercise a particular component of the SDR systems. Because of content limitations in the collection, however, the assessors were able to develop only 23 usable topics including:

> *Find reports of fatal air crashes* (Topic 62)
>
> *What economic developments have occurred in Hong Kong since its incorporation into the Chinese People's Republic* (Topic 63)

As in other TREC evaluations, once the retrieval results were submitted to NIST, the output of the participating systems was used to create pools of stories to be evaluated for relevancy by the assessors. The pools were created by taking the union of the top 100 stories for each topic output by each of the systems for each of the R1, B1, B1, B2, S1, and S2 retrieval conditions. The assessors met again in September and exhaustively examined the pools to create a reference set of relevant documents for each topic which was then used to score the results of the evaluation. Figure 3 shows the relevancy profile for the test collection with regard to the test topics.

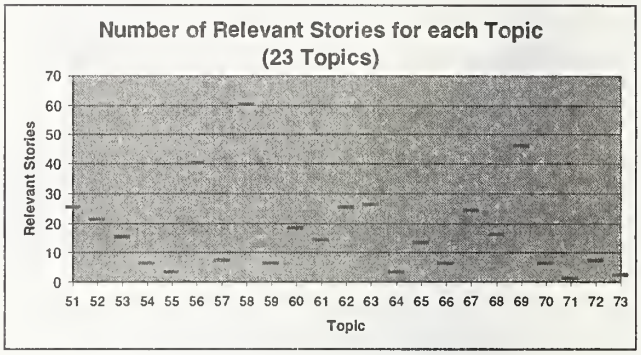


Figure 3. Relevant Stories Per Topic

On average, there were 17 relevant stories per topic. However, as Figure 3 shows, there was a great deal of variability in the number of relevant stories for particular topics. For instance, at the extremes, Topic 71 had only 1 relevant story and Topic 58 had 60 relevant stories.

4. EVALUATION RESULTS

In all, 11 sites (or recognition/retrieval teams) participated in the SDR Track. Eight of these sites performed the Full SDR task by implementing both the recognition and retrieval components of the task (S1). These sites were also required to implement the R1, B1, and B2 control conditions.

Full SDR (recognition and retrieval - R1, B1, B2, S1):

- AT&T (ATT)
- CMU Group 1 (CMU1)
- Cambridge University, UK (CUHTK)
- DERA, UK (DERA)
- Royal Melbourne Institute of Technology, Australia (MDS)
- Sheffield University, UK (SHEF)
- TNO-TPD TU-Delft, Netherlands (TNO)
- University of MA - Dragon Systems (UMass)

AT&T, CMU Group 1, DERA, RMIT, and UMass implemented secondary (S2) recognition/retrieval systems, although only DERA submitted their secondary recognition system output for scoring and redistribution.

Four of the Full SDR sites (Cambridge, DERA, RMIT, and Sheffield also implemented the Cross-Recognizer (CR) condition.

The remaining 3 sites performed only the Quasi-SDR portion of the task.

Quasi-SDR (retrieval only - R1, B1, B2):

- CMU Group 2 (CMU2)
- NSA (NSA)
- University of MD (UMD)

4.1 Speech Recognition Component Performance

The primary purpose of the SDR Track was to evaluate the retrieval of spoken documents. To this end, there was not a formal evaluation of the speech recognition component of the Full SDR systems. However, if sites used 1-best word recognition to produce transcripts as input to their retrieval systems, they were encouraged to submit these for sharing in the Cross-Recognizer condition and for NIST evaluation of the effect of recognition performance on retrieval.

It should be noted that the SDR recognition error rates are not directly comparable to error rates obtained in the NIST Hub-4 Broadcast News Transcription tasks, since the intensive verification and orthographic normalization performed for Hub-4 transcripts are not performed for SDR transcripts. Because of this, the word error rates obtained for SDR will be somewhat higher than for the identical system run on a Hub-4 test set. As a case in point, the CMU SPHINX-III-based recognition system implemented at NIST to create the transcripts for the Baseline 1 (B1) retrieval condition was scored with a 33.8% word error rate against the SDR reference transcripts. However, the identical system was benchmarked at 27.7% word error rate using the 1997 Hub-4 test set. [5]

Of the 8 participating Full SDR sites, 5 submitted recognition output to NIST for scoring. Other Full SDR sites either used an alternative recognition technique such as phone-based recognition or word lattices or chose not to share their recognition results. Figure 4 shows a frequency plot in profile of the story word error rates for each of the submitted 1-best systems. This plot gives a graphical profile of recognizer performance over the 2866 stories in the collection.

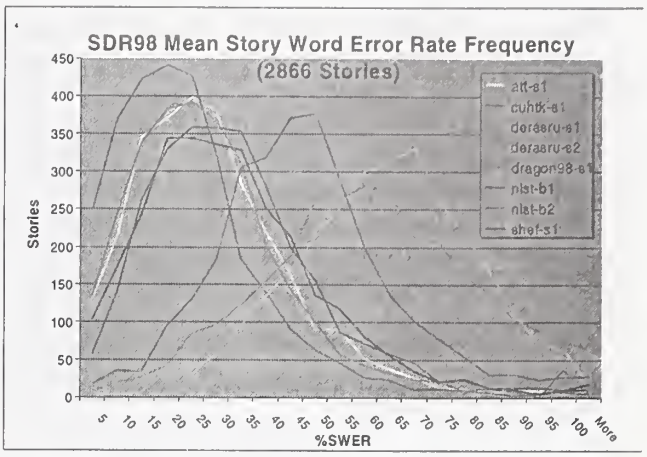


Figure 4. Story Word Error Rate Frequency Plot in Profile for Submitted Recognized Transcripts

Figure 5 shows the mean story word error rate (SWER) and mean test set word error rate (WER) for each of the submitted recognition systems. The ovals indicate no significant difference between systems in mean story word error rate error rate at 95% confidence. The best recognition results were from the Cambridge University HTK recognition system with a 24.6% test set word error rate and a 22.2% mean story word error rate.[6] A complete table of recognition scores for the submitted systems is given in Appendix A. Note that the results are slightly improved over what was reported at the TREC-7 meeting. After the meeting, it was found that there were severe story boundary annotation errors in 5 of the stories which yielded extremely high word error rates for those stories (on the order of 2,000%). These annotations were corrected and all of the results rescored.

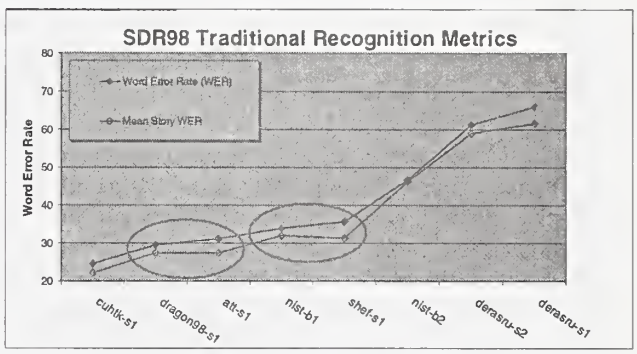


Figure 5 . Test Set and Mean Story Word Error Rate for Submitted Recognized Transcripts with Cross-System Significance at 95% for SWER

The submitted recognition systems exhibited a wide range of error rates and provided a spectrum of material for the Cross-Recognizer retrieval experiment.

4.2 Retrieval Results

Test participants were required to submit a relevance-rank-ordered list of the ID's of the top 1000 stories they retrieved for each topic. These results were then scored against the reference assessments created by the NIST assessors using the TREC_EVAL scoring software. As in other TREC tasks, the primary retrieval metric for the SDR evaluation was mean average precision over all topics. Mean average precision (MAP)1 is the metric employed in TREC retrieval tracks to provide a single figure of merit.[7] Figure 6 shows the MAP results for participating retrieval systems for the R1, B1, B2, S1, and S2 retrieval conditions.

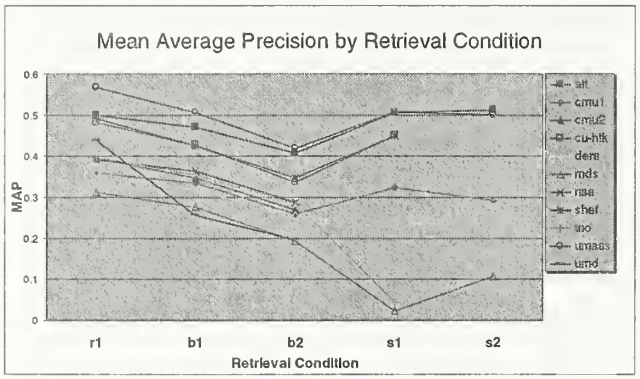


Figure 6 . Mean Average Precision for Required Retrieval Conditions

The graph shows that for all retrieval conditions except S2, the University of Massachusetts system achieved the best mean average precision. The UMass system achieved a MAP for retrieval using the human reference transcripts (R1) of .5668. The same system's retrieval for the moderate error baseline recognizer (B1) transcripts achieved a MAP of .5063, and for the high error baseline recognizer (B2) transcripts, a MAP of .4191. For the Speech input condition (S1) using their own team site's (Dragon Systems) recognizer at 29.5% word error rate, the UMass system achieved a MAP of .5075.[8] The AT&T system performed similarly for the S1 condition with a MAP of .5065. AT&T implemented a second recognition/retrieval system (S2) which achieved a MAP of .5120 - the highest results for input from a recognizer in this evaluation. It is interesting to note that the AT&T S1 and S2 results exceeded the results AT&T obtained (.4992 MAP) for the human reference transcripts (R1). AT&T attributes this to a new approach they implemented for document expansion using contemporaneous newswire texts. They applied the new document expansion approach only to their S1 and S2 runs and not to their other runs.[9] Appendix A gives a complete tabulation of the mean average precision scores for all of the systems and conditions.

In general, the results for this evaluation were quite good, with a near-linear decline in mean average precision for recognition transcripts with higher word error rates. The Cross-Recognizer retrieval results were used to further explore this apparent relationship.

4.3 Cross-Recognizer Retrieval Results and Alternative Recognition Metrics

This year, sites were encouraged to share their recognizer transcripts with other retrieval sites to implement a cross-recognizer retrieval condition. This cross testing permitted the examination of retrieval performance over a wider variety of recognized transcripts and we could begin to truly examine the relationship between recognition performance and retrieval performance. It also provided us with data to evaluate our recognition metrics for their suitability for retrieval and to experiment with new ones as well.

Four of the Full SDR sites: Cambridge University, DERA, RMIT/MDS, and Sheffield University implemented the cross-recognizer (CR) retrieval condition. The CR condition provided 9 recognition/retrieval points: the reference transcripts with "perfect" recognition, the baseline B1 and B2 transcripts, and 6 other recognizer-generated transcripts contributed by 5 sites: AT&T, Cambridge-HTK, DERA (2 sets), Dragon Systems, and Sheffield University. These recognized transcripts covered a wide range of word error rates.

When we plot mean average precision against mean story word error rate for each of the 4 retrieval systems (Figure 7), we see a linear trend in mean average precision as average recognition word error increases. The correlation averaged over these 4 retrieval systems for the 9 recognition points is ≈ .87. This high correlation indicates that there is indeed a significant relationship between word error rate and retrieval accuracy. The plot also shows a consistent pattern in performance profiles across the retrieval systems with respect to recognizers. However, the retrieval results for all systems for the B2 recognizer are low with respect to the word error rate metric (much lower than the results for the two DERA recognizers with significantly higher word error rates.) This tells us that word error rate alone is insufficient to fully predict retrieval performance.

1 Mean Average Precision (MAP) is a composite measure of retrieval performance and is equivalent to the mean across topics of the area under the uninterpolated precision/recall graph for each topic.

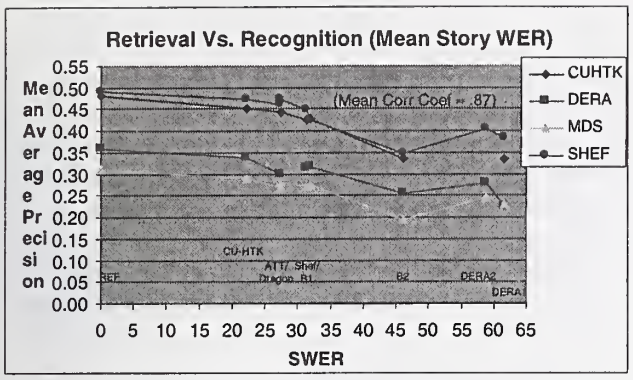


Figure 7. Cross-Recognizer Results: Mean Average Precision vs. Mean Story Word Error Rate

We believed that we might achieve an even higher correlation if a metric for speech recognition were employed which emphasizes the information-carrying words which are key for retrieval. Such a metric would be more predictive of retrieval performance and could be used to determine the suitability of a recognizer for use in a retrieval task.

We considered 3 types of metrics:

Named-Entity-based: This metric evaluates the error rate for named-entity words (people, locations, and organizations) as defined in the 1998 Hub-4 Information Extraction - Named Entity (IE-NE) Evaluation.[10] The disadvantage of this metric is that it requires named-entity annotations in the reference transcripts. Fortunately, GTE/BBN had annotated the SDR reference transcriptions for use as named-entity training data for the IE-NE evaluation.[11] We developed the following metric:

> *named entity word error rate* (ne-wer): score only the named entities in the recognizer transcripts. To implement ne-wer, we used IE-Eval/REEP Named Entity scoring software [12] to align the annotated named entity words in the reference transcript with words in the recognizer transcripts. The alignments (with embedded named-entity tags) were then scored using the NIST SCLITE speech recognition scoring software. The embedded tags permitted us to score only named-entity words. So as not to introduce entity tagger error into our metric, we ignored named entity words which might be inserted by the recognizer and evaluated only named entity words as annotated in the reference transcripts.

General IR-based: These metrics use IR approaches themselves to process, filter, and weight the words in the recognizer transcripts to be scored. Such metrics are potentially useful in predicting retrieval performance based on recognition performance and might, therefore, be used to tune a recognizer for a retrieval task. We considered 3 such metrics:

> *stop-word-filtered word error rate* (swf-wer): apply a stop-word list to the words in the reference and recognizer transcripts to remove stop (non-information-carrying) words. To implement this metric, we removed all occurrences of words in a 396-word stop word list from both the reference and recognizer transcripts. We then performed SCLITE word error rate scoring on the filtered transcripts.
>
> *Stemmed stop-word-filtered word error rate* (sswf-wer): apply a stemmer to the results of the swf-wer filtering process above to remove word differences which are irrelevant to retrieval algorithms. To implement this metric, we applied an implementation of the Porter stemmer [13] to the stop-word-filtered reference and recognizer transcripts. We then performed SCLITE word error rate scoring on the filtered transcripts.
>
> *IR-weighted stemmed stop-word-filtered word error rate* (IRW-WER). Apply an IR indexing algorithm to weight words prior to SCLITE word error rate scoring. We are currently examining IR algorithms for this application and have not yet implemented this metric.

Query-Set-Specific: These metrics evaluate the word error rate only on words given in the test topics. Such metrics are useful in analyzing the results of a given test, but are not predictive. We considered the following metric:

> *query-word word error rate* (QW-WER). To implement this, we identify the words in the test topics, remove stop words, and stem the remainder. The reference transcripts are also stemmed. The processed query word list is then used to score only the occurrences of these words in the processed reference transcripts against the corresponding aligned words in the processed recognizer transcripts. This metric has not yet been implemented.

The results of these alternative metrics as applied to the SDR recognizer are shown in Figure 8. Note that only the Named-Entity-based metrics clearly change the relative

ranking of the recognizer transcript sets. These two metrics show that the B2 recognition system was a poorer performer with regard to named entities than is evidenced by its word error rate. Our hypothesis is that the adjustment we made to the SPHINX pruning thresholds artificially reduced the likelihood of longer words being recognized - words which are more likely to be content-carrying named entities. Surprisingly, the other recognition metrics don't seem to be significantly different than word error rate in measuring recognition performance - an indication that recognition systems perform just as well (or poorly) on content words than on non-content words. This contradicts popular folklore that speech recognition systems perform more poorly on non-content-bearing "function" words. The scores for each of the metrics as applied to each of the recognized transcripts are given in Appendix A.

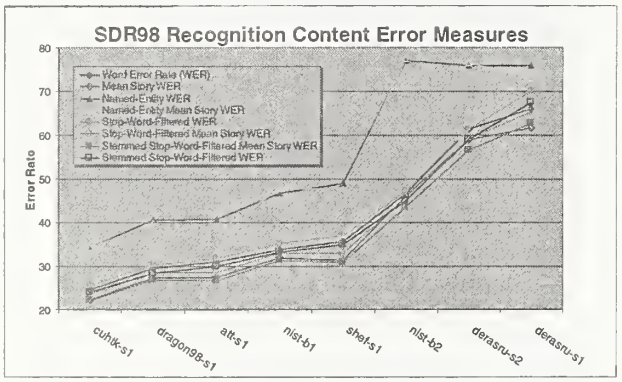


Figure 8 . Alternative Recognition Metrics

To quantify the efficacy of these metrics as predictive tools, we display a correlation analysis of the scores for the 4 retrieval systems versus the recognition metrics for each of the 9 transcript sets in Table 1.

	CUHTK	DERA	MDS	SHEF	Mean Corr	Mean Rank
WER	-0.901	-0.905	-0.785	-0.797	-0.847	6.00
SWER	-0.927	-0.912	-0.812	-0.827	-0.869	3.25
NE-WER	-0.937	-0.900	-0.897	-0.890	-0.906	3.00
NE-SWER	-0.936	-0.886	-0.900	-0.898	-0.905	3.00
SWF-WER	-0.894	-0.911	-0.777	-0.791	-0.843	7.00
SWF-SWER	-0.911	-0.915	-0.794	-0.811	-0.858	4.00
SSWF-WER	-0.897	-0.913	-0.776	-0.793	-0.845	6.25
SSWF-SWER	-0.914	-0.916	-0.794	-0.812	-0.859	3.25

Table 1 . Correlation Between Recognition Metrics and Retrieval Performance

The table shows that, on average for all 4 retrieval systems, the named entity test set word error rate (ne-wer) and named entity mean story word error rate (ne-swer) metrics provide the best correlation with retrieval performance with mean system correlation coefficient values of .906 and .905 and with minimal (best) mean ranks of 3.0 derived from the individual system correlation coefficients. The CU-HTK, MDS, and Sheffield retrieval systems are all most highly correlated with the named-entity-based metrics. However, the DERA retrieval system seems to be a bit of an outlier since it is more correlated with stemmed stop-word-filtered mean story word error rate (sswf-swer). In any case, all of the metrics including the traditional word error rate metrics are significantly correlated with retrieval performance.

The high correlation between named entity mean story word error rate (ne-swer) and retrieval performance is visually depicted in Figure 9.

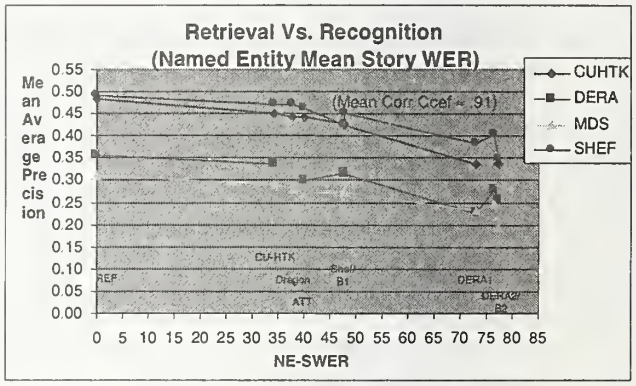


Figure 9. Cross-Recognizer Results: Mean Average Precision vs. Named Entity Mean Story Word Error Rate

We have yet to perform the scoring using the IR-weighted metric. We believe that it may provide a better predictor of retrieval performance than simple word error rate without the cost of annotation associated with named entity word error rate.

5. CONCLUSIONS

In 1997, we found that we could successfully implement a known-item retrieval task using broadcast news. In 1998, we found that we could successfully implement an ad-hoc retrieval task using a larger corpus of broadcast news. The best performance for retrieval using a speech recognizer (.5120 MAP) approached the best performance for retrieval using perfect human-generated reference transcripts (.5668).

We also found that there is a near-linear relationship between recognition word error rate and retrieval performance. We investigated alternative metrics for recognition performance that might be more predictive of retrieval performance. We found that named-entity word error rate was more highly correlated with retrieval performance than word error rate alone. We intend to continue investigating IR-based algorithms that could be applied to building recognition metrics tuned for retrieval

applications.

However, we are hesitant to declare retrieval using speech recognition-generated transcripts a solved problem. The 1998 SDR collection of 2,866 stories is still quite small for retrieval evaluation. The next challenge is to determine how well retrieval performance scales for larger realistic collections of broadcast news and to remove artificially constrained components of the evaluation such as known story boundaries.

6. FUTURE

The 1998 SDR track employed a corpus which was twice as large as that used for the 1997 track. However, the corpus was not collected for retrieval purposes and is not appropriately representative with regard to sources and time. For 1999, we plan to use a 5-month subset of the TDT-2 corpus for the SDR evaluation. The TDT-2 corpus, which was collected by the Linguistic Data Consortium for the DARPA Topic Detection and Tracking Tasks, contains 632 hours/24,503 stories of broadcast news from ABC World News Tonight, CNN Headline News, PRI The World, and several Voice of America programs. It is well-suited for the SDR task in that the broadcast news sources are evenly sampled over a 6-month time period from January through June, 1998.[14] Further, it contains a complementary newswire corpus which can be used by sites who wish to explore the application of rolling language models to the SDR recognition task. Rolling language models evolve over time to changes in language and will address real-world use of recognition for time-continuous tasks. Next year's evaluation will support two language-modeling modes: fixed and rolling.

The TDT-2 corpus does not have Hub-4-style transcripts, but does have closed-caption transcriptions for the television programs and comparable quality transcripts for the radio programs. A minimum of 10 hours of randomly selected stories in the corpus will be transcribed in the Hub-4 style so that speech recognition performance can be benchmarked. It is hoped, however, that the entire corpus will eventually be transcribed so that more extensive benchmarking can be performed.

The NIST assessors will create 50 ad-hoc-style topics for the 1999 SDR track using the existing transcripts. These transcripts will also be used in the reference condition for the evaluation.

There was increased interest at TREC-7 in supporting an evaluation condition in which story boundaries are unknown, which would more naturally model a real implementation of SDR. To support this condition, systems will be permitted to make an optional run on the baseline speech recognizer transcripts and their own recognizer output without story boundaries. The systems will output a time stamp for the top 1000 retrieved stories rather than a story ID. Each time stamp will be mapped to the story that contains it. Duplicate stories will be eliminated, so that systems which output multiple time stamps referring to the same stories will be penalized. The inferred story IDs will then be used to implement traditional TREC_EVAL scoring. In this task, systems should attempt to find the mid-point or "hotspots" in stories. This approach is simpler than other possible schemes that might require systems to output story boundaries and which are scored on distances. However, since story segmentation is not a focus of this track, and since it is desirable to have comparable results for both story-boundary-known and story-boundary-unknown conditions, this has been determined to be the most expeditious approach.

7. ACKNOWLEDGEMENTS

The authors would like to thank Karen Sparck Jones of Cambridge University for her tireless efforts in promoting the SDR Track and for her assistance in developing the evaluation plan. We'd also like to especially thank CMU for their contribution of the SPHINX-III recognizer for use as SDR the baseline recognizer. CMU's Kristie Seymore, Mosur "Ravi" Ravishankar, and Matt Siegler were of great help in getting SPHINX up and running at NIST. Finally, We'd like to thank Gilbert Sanseau of the NIST NLPIR Group for his suggestions of IR algorithms in our exploration of alternative speech recognition metrics.

NOTICE

REFERENCES

[1] Voorhees, E., Garofolo, J., Stanford, V., and Sparck Jones, K., *TREC-6 1997 Spoken Document Retrieval Track Overview and Results*, Proc. TREC-6, 1997 and 1998 DARPA Speech Recognition Workshop, February 1998.

[2] Graff, D., Wu, Z., MacIntyre, R., and Liberman, M., The 1996 Broadcast News Speech and Language-Model Corpus, Proc. DARPA Speech Recognition Workshop, February 1997.

[3] BEOWULF Project, NASA Center of Excellence in Space Data and Information Sciences, http://cesdis.gsfc.nasa.gov/linux/beowulf/

[4] *The Generic NQS Web Site - OpenSource Batch Processing*, http://www.gnqs.org/

[5] Pallett, D.S., Fiscus, J.G., Martin, A., Przybocki, M.A., *1997 Broadcast News Benchmark Test Results: English and Non-English*, Proc. DARPA Broadcast News Transcription and Understanding Workshop, February 1998.

[6] Johnson, S.E., Jourlin, P., Moore, G.L., Sparck Jones, K., Woodland, P.C., *Spoken Document Retrieval for TREC-7*, Proc. TREC-7, November 1998.

[7] Voorhees, E.M., Harman, D., *Overview of the Seventh Text REtrieval Conference (TREC-7)*, Proc. TREC-7, November 1998.

[8] Allan, J, Callan, J., Sanderson, Xu, J., *INQUERY and TREC-7*, Proc. TREC-7, November 1998.

[9] Singhal, A., Choi, J., Hindle, D., Lewis, D.D., Pereira, F., *AT&T at TREC7*, Proc. TREC-7, November 1998.

[10] Przybocki, M.A., Fiscus, J.G., Garofolo, J.S., Pallett, D.S., *1998 Hub-4 Information Extraction Evaluation*, Proc. 1999 DARPA Broadcast News Workshop, March 1999.

[11] Miller, D., Schwartz, R., Weischedel, R., Stone, R., Named Entity Extraction from Broadcast News, Proc. 1999 DARPA Broadcast News Workshop, March 1999.

[12] Douthout, A., *Hub-4 1998 IE-NE Scoring Software with Recognition and Extraction Evaluation Pipeline*, SAIC, ftp://jaguar.ncsl.nist.gov/csr98/official-IE-98_scoring.tar.Z

[13] Porter, M.F., An algorithm for suffix stripping, Program 14 (3), July 1980, pp. 130-137.

[14] Cieri, C., Graff, D., Liberman, M., Martey, N., Strassel, S, TDT-2 Text and Speech Corpus, Proc. 1999 DARPA Broadcast News Workshop, March 1999.

Appendix A: 1998 TREC-7 Spoken Document Retrieval Track Summary Results

Retrieval Results - Mean Average Precision

Site	R1	B1	B2	S1	S2	CR-ATT	CR-CUHTK	CR-DERA1	CR-DERA2	CR-Dragon	CR-Shef
ATT	0.4992	0.4700	0.4065	0.5065	0.5120	0.5065					
CMU1	0.3577	0.3345	0.2590	0.3224	0.2926						
CMU2	0.3936	0.3472	0.2693								
CUHTK	0.4817	0.4272	0.3352	0.4509		0.4419	0.4509	0.3352		0.4428	0.4251
DERA	0.3579	0.3164	0.2551	0.2242	0.2768		0.3375	0.2242	0.2768	0.2990	0.3134
RMIT-MDS	0.3107	0.2753	0.1937	0.0223	0.1063	0.2812	0.2906	0.2309	0.2443	0.2704	0.2730
NSA	0.3907	0.3640	0.2868								
SHEF	0.4916	0.4243	0.3471	0.4495		0.4717	0.4713	0.3836	0.4047	0.4613	0.4495
TNC	0.3970	0.3533	0.2833	0.0436							
UMass	0.5668	0.5063	0.4191	0.5075	0.5000						
UMD	0.4386	0.2557	0.1967								

Speech Recognition Results - Various Metrics (%error)

ASR Metric	R1	B1	B2			CR-ATT	CR-CUHTK	CR-DERA1	CR-DERA2	CR-Dragon	CR-Shef
WER	0.0	33.8	46.6			31.0	24.6	66.0	61.3	29.5	35.6
SWER	0.0	31.9	46.1			27.4	22.2	61.6	58.9	27.3	31.3
NE-WER	0.0	46.8	77.1			40.8	34.2	75.9	75.9	40.6	49.0
NE-SWER	0.0	47.7	77.3			37.8	34.2	73.1	76.5	40.1	47.7
SWF-WER	0.0	35.1	47.2			31.8	25.7	70.0	61.3	30.2	37.0
SWF-SWER	0.0	33.1	45.7			28.4	23.6	65.3	59.0	28.6	32.9
SSWF-WER	0.0	33.2	44.9			29.9	24.0	67.4	58.9	28.3	34.8
SSWF-SWER	0.0	31.3	43.5			26.7	22.1	62.6	56.6	26.8	30.8

Overview of TREC-7 Very Large Collection Track*

David Hawking
CSIRO Mathematics and Information Sciences,
Canberra, Australia
Nick Craswell and Paul Thistlewaite
Department of Computer Science, ANU
Canberra, Australia
David.Hawking@cmis.csiro.au, {pbt,nick}@cs.anu.edu.au

January 22, 1999

Abstract

In line with the wishes of last year's participants, this year's VLC track was essentially a re-run of last year's with a five-fold increase in data size. The data used was a completely new 100-gigabyte collection of Web documents (the VLC2) whose characteristics are presented here. This time, two orders of magnitude scale-up was investigated using 1%, and 10% samples as well as the full collection. Six groups managed to complete the full VLC task, of which five completed last year's track. An overview is given of the track participants, the methods used and the results obtained. One group of participants, using hardware costing less than $US10,000, have shown that a hundred gigabyte collection can be indexed in less than ten hours and that quite good rankings (better than several well-known search engines) can be produced from queries processed in less than one second.

1 Background and Motivation

The arguments for test collection sizes representative of the data sizes encountered in practice have been made by Harman [1992], Hawking and Thistlewaite [1997] and Hawking, Thistlewaite, and Harman [1999]. Within the last eighteen months, Web search engines have crossed the one hundred gigabyte data size barrier and some now closely approach the terabyte level. [Digital Equipment Corporation 1998]

At the same time, many TREC-6 VLC track participants expressed confidence that their systems were capable of indexing and querying collections much larger than 20 gigabytes.

Accordingly, a new 100-gigabyte test collection (the VLC2) has been developed for this year's track. The data used is part of a "Web-crawl" carried out by the Internet Archive in 1997. [Internet Archive 1997]

Naturally, as stated in last year's track overview, it is not feasible to obtain complete relevance judgments for collections of this size. Because of this, effectiveness measures are restricted to those which

*The authors wish to acknowledge that this work was carried out within the Cooperative Research Centre for Advanced Computational Systems established under the Australian Government's Cooperative Research Centres Program.

can be derived from short rankings. (Fortunately, this year, it was feasible to judge a much larger number of runs than was the case last year.)

It was envisaged that TREC participants could examine in detail the effectiveness of their system on the main Ad Hoc task and then, if interested in larger collections, check speed and scalability in the VLC track. The VLC early precision measure exists mainly to ensure that speed is not achieved at the expense of effectiveness.

2 Organisers and Participants

As in the past, the VLC track was organised by the Advanced Computational Systems Cooperative Research Centre (ACSys), whose core participants are the Australian National University, the Commonwealth Scientific and Industrial Research Organisation, Fujitsu, Sun, DEC, StorageTek and Silicon Graphics. Support for the VLC track is a natural extension of ACSys research interests in "managing the information explosion".

ACSys obtained the tapes from the Internet Archive and supplied the human and machine resources to format and distribute the data. It also recruited and employed the VLC assessors. This year, a number of participating groups made financial contributions to the non-trivial cost of tape media and distribution.

Eleven groups received VLC2 data tapes. In the end, seven groups submitted runs: ACSys; City; UMass; UWaterloo; AT&T; the Okapi Group and FS Consulting. Three may be considered commercial organisations, three are universities and one is a government sponsored collaborative research centre.

3 The Data

Additional information on both editions of the Very Large Collection (VLC and VLC2) is available on the VLC web page. [Hawking et al. 1997]

A subset of the data tapes supplied by the Internet Archive was selected and formatted as the 100.426 gigabyte VLC2. From it, uniform 1% (BASE1) and 10% (BASE10) samples were defined as baselines.

The data was distributed on tape using `gzip` compression. The additional compression achieved by `gzip` (compared to standard Unix compress) saved considerably on tape cost and tape writing times. Even more effective compression systems are available but were not used, either because tests showed they would be too slow (up to eight days to compress the data and up to five days to decompress it on a Sun Ultra!) or because of the risk that some participating groups might experience difficulties in obtaining the necessary decompression code.

Complete sets of tapes were shipped to registered participants starting on June 15, 1998, allowing roughly eleven weeks to work on the task up to the submission deadline of September 8.

3.1 Access to the VLC Data

Access to the data is subject to the terms and conditions of the data permission forms available via the VLC Web page. [Hawking et al. 1997] These agreements prevent further redistribution, restrict use of the data to the usual TREC purposes and require recipients to delete documents if requested to do so by copyright holders, ACSys or the Internet Archive.

As previously mentioned, many of the groups participating in the track contributed to the cost of tape media and distribution, but financial contribution was not compulsory.

3.2 Overview of Data

The average document length is 5.67 kB compared to 3.2 for CDs 1-5 and 2.8 for the first edition VLC. The longest VLC2 document occupies about 4 MB. By comparison, the longest document (in the FR94 collection on CD4) in previously distributed TREC data was 6.2 MB.

The 1% and 10% baseline samples were created by selecting every 100th and every 10th compressed file respectively. BASE1 was thus a uniform sample of BASE10. Average document lengths in the samples are within a few percent of that for VLC2.

3.3 Formatting

The software used by the Internet Archive for "spidering" (collecting the pages from the Web) is an in-house system whose details are unknown to the VLC organisers. ACSys used `perl` scripts to convert the supplied data into VLC2 format. These scripts did not remove or convert any page content, merely inserting `<DOC>`, `<DOCNO>`, `</DOC>`, and `</DOCNO>` tags and a unique TREC-style document identifier. In addition, the HTTP header information (an average of 277 bytes) returned by the `httpd` daemon supplying the page, was surrounded by a `<DOCHDR>` `</DOCHDR>` pair. All pages with MIME type "text/html" were included, except a few longer than 2 MB.

It should be noted that this means that the collection includes some pages which are not in the English language, pages which are not in a Roman character set and pages which are in fact not text at all. (Some binary files (GIF files and compressed `tar` files) were erroneously typed by the daemons which served them). It also contains large numbers of duplicate (or near-duplicate) pages and pages which contain no text content (or very little).

Consequently, the VLC2 data is representative of the raw results likely to be obtained by Web spidering and thus representative of the pages likely to be indexed by Web search engines.

The 100.426 gB VLC2 is divided up into 97 collections, each in its own directory. Each collection is written as a separate tar file on tape. Collections each contain about 11 subdirectories, each containing around 48 bundles of documents (gzipped files), each containing on average 370 documents.

It has been distributed in three different tape formats: DLT-4000 (2 tapes, second includes baselines), DDS-3 (three tapes for VLC2, fourth for baselines) and DDS-2 (eight tapes for VLC2, ninth for baselines).

VLC2 document identifiers are structured to allow unambiguous identification of collection, subdirectory and filename. Every document contained the essential "SGML" markers delimiting documents and document identifiers. A program `coll_check` was used to check that each document conformed to this elementary structure and that document identifiers were unique. The only problem detected (initially) by this program was caused by a fragment of a TREC Federal Register document appearing on someone's Web site!

4 The Task

Full guidelines for the VLC track are available on the VLC web page [Hawking et al. 1997]. In essence, participants were required to process queries generated from the TREC-7 Ad Hoc topics (351-400) over both the baselines and the VLC2 datasets and to return for assessment only the first 20 documents retrieved in each case. Elapsed times (as would have been observed by a human with a stopwatch) for indexing the datasets and processing queries were recorded and system details and costs as well as disk space requirements were reported via a questionnaire. The focus was on the ratios of the various measures (see below) across the three data sizes

All retrieved documents were judged.

Participants were encouraged to submit at least one set of results using queries derived automatically from the Title and Description fields of the topic statements.

5 The Measures

M1. Completion. (Can the system process data of this size at all?)

M2. Precision@20. (P@20)

M3. Query response time. (Elapsed time as seen by the user.)

M4. Data Structure Building time. (Elapsed time as seen by the user.)

M5. Gigabyte-queries/hour/kilodollar. (Bang per buck.)

M6. Modified average precision. This is a new measure introduced to take account of the fact that, for some topics, the number of relevant documents in a collection (BASE1, BASE2 or VLC2) may be so small as to artificially limit P@20. M6 is calculated by summing the precision at each point in a ranking where a relevant document occurs and dividing the sum by the lesser of 20 and the total number of relevant documents for the topic in that collection.

M4 represented the minimum possible elapsed time from receiving the data until the data structures necessary to process the queries used in M3 were built, using the chosen hardware and indexing software. Time to actually read the tapes was excluded. The starting point was the compressed data files on disk after unpacking the tarfiles. M4 included the time to build all structures (such as inverted files) which are necessary to process the final query.

6 The Assessments

Four judges were employed to assess the VLC2 document pool. One was a research assistant in Sociology, another a final year Philosophy/Art Curatorship student with employment experience in summarisation of technical articles, another a Science graduate and the fourth a graduate in both Arts/Asian Studies and Science. The first judge was also employed in the TREC-6 track.

Topics were assigned to judges on an arbitrary basis. All judgments for a particular topic were made by the same judge.

Groups were permitted to submit multiple sets of runs but were asked to indicate a priority order for assessment. As it turned out, ALL runs submitted prior to the deadline were assessed and the resulting `qrels` and evaluations were distributed on October 14th.

When it became clear that good progress was being made on judging, groups were offered the chance to submit additional runs (or deeper rankings for previously submitted runs.) These after-deadline runs were all completely judged. Unfortunately, due to restrictions on the availability of judges, it was necessary to transfer responsibility for some topics from one judge to another. In these cases, the new judge re-judged all the before-deadline documents on those topics and the old judgments were discarded. The result is that there are two different sets of qrels. In both sets, only one judge was used per topic.

Table 1: Groups completing the VLC task. Six groups attempted the full 100 gigabyte task and one additional run was submitted using just the BASE1 collection. The hardware configuration shown is the full configuration available. In some cases, groups used only part of the available configuration, even on the 100 gigabyte task and in some cases, groups used less hardware on the baselines. The pair of figures in the I/O column indicate the number of channels and the number of disks used (per CPU, unless otherwise noted). Cost is an estimate of the U.S. list price of a system comparable to the one used.

Group	Software	CPUs	MHz	Total RAM	I/O	Cost
ACSys	PADRE98	8 x DEC Alpha	266	1152MB (dist.)	1,2	$24k
ATT	Smart	20 x SGI R10000	195	8192MB (sh.)	1,1	$115k
City	PLIERS	8 x DEC Alpha	266	1152MB (dist.)	1,2	$24k
FSC	MPS	1 x Sun Ultra	200	256MB	3,19	$15k
Okapi	Okapi	2 x Intel P2 (Solaris)	400	512 MB (sh.)	?,15(tot.)	$37k
UMass	Inquery	4 x Sun Ultra	167	1024 MB (sh.)	?,?	$130k
Waterloo	Multitext	4 x Intel P2	300	512 MB(dist.)	2,4	$8k

Table 2: M2: Precision at 20 documents retrieved. Numbers in parentheses represent ratios to the appropriate baseline measures. The last column characterises the method used to generate this query set. T, D, and N refer to the Title, Description and Narrative fields of the topic. RF refers to automatic relevance feedback, and *Cov. dens.* refers to cover density ranking. *Req. wds.* indicates that automatically generated required words were added to the query.

Group	BASE1	BASE10	VLC2	Q gen.
UMass(1)	.202	.429(2.12)	.625(3.09,1.46)	T+D+N RF
UMass(2)	.204	.441(2.16)	.624(3.06,1.42)	(1) + Req. wds.
UMass(3)	.208	.419(2.01)	.598(2.88,1.43)	T+D RF
Okapi(1pr)	.180	.376(2.09)	.541(3.01,1.44)	T+D
Okapi(1pr.tnd)	-	-	.598(3.32,1.59)	T+D+N
Okapi(3)	-	-	.509(2.83,1.35)	T+D RF
Okapi(3.tnd)	-	-	.545(3.03,1.45)	T+D+N RF
Waterloo(0)	.190	.369(1.94)	.442(2.33,1.20)	T Cov. dens.
Waterloo(1)	.235	.474(2.02)	.598(2.55,1.26)	Manual (6.4 term)
Waterloo(2)	.223	.411(1.84)	.574(2.57,1.40)	Manual (1.9 term)
Waterloo(3)	.110	.288(2.62)	.397(3.61,1.38)	Variant of (0)
ATT (vf)	.188	.384(2.04)	.503(2.68,1.31)	T+D
ATT (vfe)	-	-	.587(3.12,1.53)	T+D RF
ATT (vi)	-	-	.357(1.90,.930)	T+D
ATT (vie)	-	-	.375(1.99,.977)	T+D RF
ACSys (5)	.139	.321(2.31)	.442(3.18,1.38)	T+D (5 term)
ACSys (2)	-	-	.298(2.14,.930)	T+D (2 term)
FSC	.128	.268(2.09)	.345(2.70,1.29)	T
City (1)	.080	-	-	T+D
City (2)	.056	-	-	T+D

Table 3: M3: Average Query Processing Time (Elapsed seconds per query). Numbers in parentheses represent ratios to the appropriate baseline measures.

Group	BASE1	BASE10	VLC2
ACSys (5)	0.061	0.168(2.75)	1.47(24.1,8.74)
ACSys (2)	-	-	0.887(14.5,5.28)
ATT (vf)	1.44	6.41(4.45)	5.80(4.03,0.906)
ATT (vfe)	-	-	12.0(8.33,1.87)
ATT (vi)	-	-	2.18(1.52,0.341)
ATT (vie)	-	-	8.00(5.58,1.25)
City (1)	0.593	-	-
City (2)	1.74	-	-
FSC	0.10	0.46(4.6)	51.8(518,113)
Okapi (1pr)	0.96	3.74(3.88)	25.9(26.9,6.92)
Okapi (1pr.tnd)	-	-	81.5(84.5,21.8)
Okapi (3)	-	-	68.0(70.6,18.2)
Okapi (3.tnd)	-	-	105(109,28.1)
UMass (1)	8.4	85.2(10.2)	712(84.7,8.35)
UMass (2)	9.6	88.8(9.25)	718(74.7,8.07)
UMass (3)	7.2	54(7.5)	526(73,9.73)
Waterloo (0)	0.306	0.294(0.960)	0.708(2.31,2.41)
Waterloo (1)	0.216	0.377(1.75)	1.51(6.99,4.00)
Waterloo (2)	0.251	0.299(1.19)	0.882(3.51,2.95)
Waterloo (3)	0.148	0.212(1.43)	0.619(4.18,2.92)

Table 4: M4: Data Structure Building Time (Elapsed Hours). Numbers in parentheses represent ratios to the appropriate baseline measures. UMass indicated that the starred times are likely to be significant over-estimates of the true values.

Group	BASE1	BASE10	VLC2
ACSys	0.0434	1.71(39.4)	7.73(178.1,4.52)
ATT	0.43	5.14(12.0)	6.55(15.2,1.27)
City(1)	0.0794	-	-
FSC	1	10(10)	100(100,10)
Okapi	0.42	3.85(9.167)	36.1(86.0,9.38)
UMass	2.67*	26.15(9.79)*	35.45(13.28,1.36)
Waterloo	0.0519	0.504(9.71)	5.33(102.7,10.6)

Table 5: MS: Data Structure Sizes (gigabytes). Numbers in parentheses represent ratios to the appropriate baseline measures. In the case of Okapi, the size of the raw text (compressed) must be added for queries which use relevance feedback. The UMass system also expects the raw text to be available.

Group	BASE1	BASE10	VLC2
ACSys	0.255	0.902(3.53)	7.70(30.2,8.54)
ATT	0.329	2.09(6.35)	20.8(63.2,9.95)
City(1)	0.124	-	-
FSC	0.27	2.5(9.26)	24(88.9,9.6)
Okapi	1.58	11.5(7.28)	109.7(69.4,9.54)
UMass	0.68	6(8.82)	53(77.9,8.83)
Waterloo	0.789	6.25(7.92)	44.6(56.5,7.14)

Table 6: M5: Gigabyte-queries per hour per kilodollar for 100 gigabyte runs. For each group the fastest run of queries derived automatically from T+D fields is presented.

Group	Runid	Queries/Hr	kilo$	gB-Q/Hr/kilo$	Last Year
Waterloo	uwmt7v3	5816	8.5	6.84×10^4	7.20×10^3
UMass	inq5vlc3	6.84	130	5.26×10^0	3.8×10^0
ATT	att98vi	1651	115	1.44×10^3	7.89×10^1
FSC	fsclt7a-v100	69.5	15	4.63×10^2	NA
Okapi	ok7vf1pr	139	37	3.76×10^2	6.88×10^1
ACSys	acsys7_100_2	4059	24	1.69×10^4	1.50×10^1

Table 7: M6: Modified average precision. Numbers in parentheses represent ratios to the appropriate baseline measures.

Group	BASE1	BASE10	VLC2
ACSys (5)	.1535	.2326(1.52)	.3108(2.03,1.34)
ATT (vf)	.2098	.3035(1.45)	.3810(1.82,1.26)
FSC	.1490	.1975(1.33)	.2407(1.62,1.22)
Okapi (1pr)	.2155	.2812(1.31)	.3957(1.84,1.41)
UMass_(1)	.2463	.3407(1.38)	.5201(2.11,1.53)
UMass (2)	.2677	.3514(1.31)	.5143(1.92,1.46)
UMass (3)	.2485	.3316(1.33)	.4832(1.94,1.46)
Waterloo (0)	.2472	.2897(1.17)	.3380(1.36,1.17)
Waterloo (1)	.3099	.3902(1.26)	.5005(1.62,1.28)
Waterloo (2)	.2728	.3360(1.23)	.4659(1.71,1.39)
Waterloo (3)	.1428	.2179(1.53)	.2921(2.05,1.34)

References in the present paper to pool sizes and relevant sets relate to the before-deadline submissions and judgments. Groups wishing to compare new runs with official runs reported here, must re-score the official runs using the new qrels in `http://pastime.anu.edu.au/TAR/Qrels/`.

The document pool (derived from baseline *and* VLC submissions) contained 16,292 document/topic pairs of which 4,440 were judged relevant. By contrast, the corresponding figures for the TREC-7 Ad Hoc task (using the same topics but a disjoint set of documents) were 80,345 and 4,674.

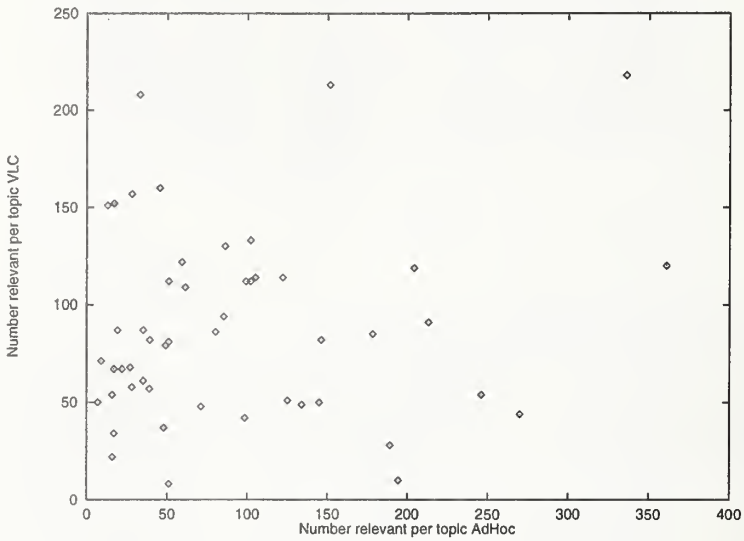


Figure 1: The relationship between numbers of known relevant documents found for the same 50 topics in two disjoint corpora: the VLC2 and the TREC-7 Ad Hoc corpus. VLC2 judgments used were the "before-deadline" set. Pearson $r = 0.13$.

The range of numbers of relevant documents found in the VLC2 (including the baselines) per topic was 8 - 218 (compared to 7 - 361 for the Ad Hoc task.) The figure of 218 is almost certainly an underestimate due to the small number of runs and the shallow judging.

Each point in the scatter plot in Figure 1 plots the number of relevant documents for the VLC2 collection against the corresponding number for the Ad Hoc collection for a particular topic. There is no significant correlation between the number of relevant documents in the two collections (Pearson $r = 0.13$, $p > 0.05$).

In Figure 2, the topics have been ordered by increasing number of relevant documents: a) for the Ad Hoc task; and b) for the VLC (using both before and after-deadline judgments. The number of relevant documents has been plotted against topic rank for each ordering. Considering the before-deadline judgments, the number of relevant documents per topic is generally greater for the VLC2 than for the Ad Hoc collection up to about 85 documents found relevant, after which the contrary is the case. It is possible that this may be the incompleteness of the VLC2 judgment pool starting to manifest itself. Indeed, when using the more complete after-deadline VLC judgments, the VLC2 line remains above the Ad Hoc line up to about 180 documents found relevant.

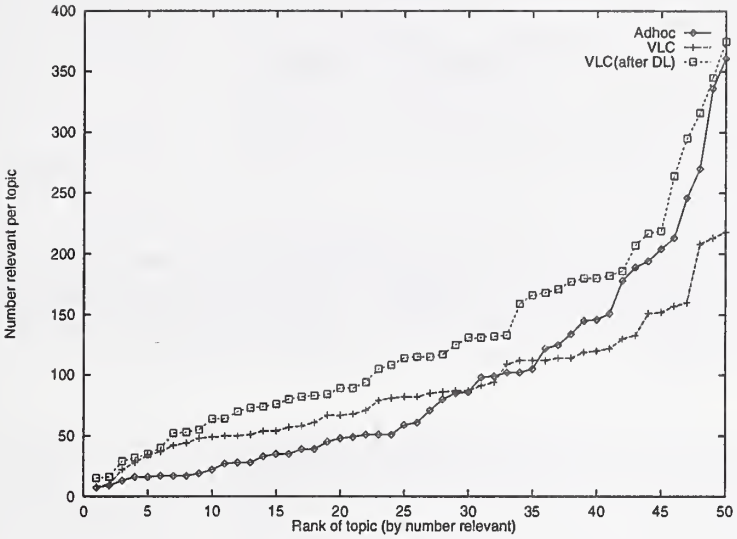


Figure 2: Ordered rankings of number of relevant documents per topic for the same 50 topics in two disjoint corpora: the VLC2 and the TREC-7 Ad Hoc corpus. Two traces are shown for the VLC2 corpus, one using "before-deadline" judgments and the other the revised set incorporating judgments of runs submitted after the deadline.

6.1 Were the Baseline Collections Unbiased Samples?

This is an important question, because it may determine the "scalability" of early precision and perhaps influence other measures. The process of selecting the baseline subsets (described above) is not inherently biased but the results may be biased with respect to a particular set of topics.

Of the 7445 documents retrieved in the runs over the full VLC2, 491 were also in BASE10 and 48 were also in BASE1. The proportion of documents in the VLC2, BASE10 and BASE1 which were retrieved by VLC (not baseline) runs were 4.01×10^{-4}, 2.64×10^{-4} and 2.56×10^{-4}, respectively. Unfortunately, tests of one-sample proportion (with finite sample correction) show that, for both baselines, the sample proportions lie outside the 95% confidence interval. Hence, it appears that samples are biased with respect to proportion of retrieved documents. Consequently, caution must be exercised when assessing increases in precision from baselines to the full collection.

By contrast, the same test applied to the BASE10 pool (4,500 documents) gives no reason to suggest that BASE1 is a biased sample of the BASE10 collection. (The proportions were 2.42×10^{-3} v. 2.27×10^{-3}).

6.2 Baseline Relevant Sets

There were 497 documents judged relevant in the BASE1 collection. There were three topics for which no relevant documents were found in BASE1.

There were 1673 documents judged relevant in the BASE10 collection. There was only one topic for

which no relevant documents were found in BASE10.

7 Characteristics of Submitted Runs

The seven groups which submitted runs are listed in Table 1.

7.1 Hardware Used

Three shared-memory systems were used: ATT's 20-processor Silicon Graphics system, UMass's 4-processor Sun SPARCserver, and Okapi's 2-processor Dell. No group completed the 100 gigabyte task using a single procesor, however Okapi indicated that the second CPU made very little difference to elapsed times in their case.

Note that:

1. A majority of groups used hardware they had access to rather than explicitly choosing it for the task. Their systems may have run just as fast on much cheaper hardware.
2. Few groups were able to run their system in dedicated mode. It is difficult to control for the effect of other users.
3. It is difficult to derive a comparable dollar value for a fraction of a very expensive system or for obsolete systems.

8 Questions Addressed

The main questions addressed by participants basically related to demonstrating the capabilities of their system on the large scale task, measuring speed and effectiveness and also observing the scalability of their systems.

Other questions will presumably be covered by each group in their own paper.

9 The Results

The results are presented in Tables 2-6

1. Table 2 shows that there is a significant rise in P@20 for all systems moving from the sample collections to the full VLC2. This is consistent with observations in last year's VLC track ([Hawking and Thistlewaite 1997; Hawking et al. 1999]) but is confounded by the evidence that the samples appear to have been biased with respect to the particular set of topics. It is almost certainly the case that a major part of the explanation of the precision increase is that precision in the small samples is severely limited by the number of relevant documents in the sample for many topics. Even a perfect retrieval system cannot achieve non-zero P@20 if there are no relevant documents.
2. Table 7 shows that modified average precision decreases less dramatically than P@20 but still increases. Note that nearly all runs included topics which failed to find any relevant documents and consequently scored zero on modified average precision. Modified average precision was zero for

between 6 and 25 topics (median 13) on the BASE1 runs and for between 1 and 7 topics (median 4) on the BASE10 runs. Even on the 100 gigabyte task, there was only one run `uwmt7vi` which found relevant documents on every topic. The number of topics scoring zero on modified average precision ranged from 0 to 8 (median 2) for the VLC runs.

3. Table 3 shows that two groups (ACSys and Waterloo) achieved sub-second query processing time over the full collection. In the UMass case, query processing time was approximately proportional to the corpus size, while for FSC, query processing time grew non-linearly, presumably due to virtual memory effects. ACsys, ATT, Okapi and Waterloo were able to control dilation of query processing time to a factor less than the collection scale-up. In general, this is because when the data-dependent parts of query processing are made more efficient, fixed costs (such as accessing 20 document identifiers and looking up terms in a term dictionary) become significant for small collections. In the ATT case this was also partly due to the use of more hardware for the larger collections.

4. Table 4 shows that, for FSC, Okapi and Waterloo, data structure building time is an essentially linear function of data size. This may have been the case for UMass too, but they have indicated that their baseline timings are not reliable. ATT controlled the increase in running time by deploying more hardware for the larger collections. ACSys used the same hardware for each collection but the running time of its algorithm is heavily dependent both upon the balance between the size of the chunk of text being indexed and the available RAM and upon load balance between workstations. With more RAM, and more uniform load balance the relationship may have been more linear.

5. Table 5 shows considerable variation in index size across the groups, ranging from 8% (ACsys) to 110% (Okapi) of the raw data size. All groups observed a scale-up in index size of between 7 and 10 for the transition from BASE10 to VLC2. However, ACSys observed a much lower scale-up for the transition from BASE1 to BASE10. This is believed to be because the uncompressed term dictionaries are larger relative to the raw data size in the BASE1 case (with many entries replicated over 8 workstations).

6. Table 6 shows that "bang-per-buck" measures have improved considerably since last year.

10 How Would Commercial Web Search Engines Perform?

Table 8: M2/M6: P@20 and modified average precision performance for Web Search Engines, using title-only queries and the real Web. The range of official VLC results (re-evaluated using after-deadline judgments to ensure comparability) is also shown.

Engine	1	2	3	4	VLC range	VLC median
P@20	.306	.288	.231	.377	.283 - .617	.490
Mod. Ave prec	.2228	.2000	.1262	.2693	.1535 - .5154	.3736

Out of interest, TREC-7 title-only queries were fed to four well-known Web search engines. The engines were searching the current Web rather than the VLC2 frozen snapshot. Top 20 results for each

of the topics over the real Web were then presented to the same judge who had judged the documents from VLC2 for that topic, using the same assessment interface and the same concepts and evidence that they had built up.

Results for these search engines are presented in Table 8. As may be seen, all four engines perform well below the median for VLC submissions on both P@20 and modified average precision.

11 Discussion and Conclusions

1. Considerable speed improvement has been achieved by most of the five groups who participated last year. Apart from additional disk storage, the gain was achieved without massive hardware upgrades. In many cases, the appropriate cost to assign to systems used actually declined, either because expensive hardware was not used or because old machines dropped off vendor price-lists and comparable performance is now available much more cheaply.

2. It should be no surprise, given popular experience with large Web search engines, that sub-second query processing is possible over collections the size of the VLC2. What is perhaps surprising is that such performance is possible from relatively small scale hardware.

3. In the absence of huge amounts of RAM, query processing using uncompressed indexes comes to be dominated by disk latencies. For example, if the list of document names (of the order of 300 MB uncompressed) is not memory-resident, then 20 disk accesses at 10-15 msec. each (0.2 - 0.3 sec.) are likely to be needed to produce a top 20 ranking. Accepting that there must be at least one I/O request per query term to retrieve the posting list, it is important to minimize the number of I/O requests required to *locate* the posting list and also important to avoid dividing the collection into multiple sub-collections which must be separately searched.

4. It would be possible to speed up query processing dramatically if huge amounts of RAM were available. For example, in the ACSys case, a total of 8 gigabytes of RAM (as used in each of the current Alta Vista query processing engines) would be sufficient to load ALL data structures for the 100 gB collection into memory, totally obviating the need for disk I/O. Even half of this would suffice for most queries and compression techniques would reduce RAM requirements still further.

5. The effectiveness scores of the public Web search engines are all considerably below the median for the VLC participants, despite the fact that some of them have access to many more documents. Some VLC participants used very much longer queries (and two runs were manually generated), but P@20 results for the two Title-only runs submitted by VLC participants were still better than the best Web engine tested.

12 The ACSys VLC Medal

ACSys offered a medal to any group submitting a 100-gigabyte run which achieved:

1. Average query processing speed of 2 seconds or less;

2. Indexing time under 10 hours.

3. Median P@20 or better.

Three groups (ATT, Waterloo and ACSys (ineligible for the medal)) achieved the first two criteria. ATT also met the third criterion with its vfe run but unfortunately this run did not satisfy the first criterion.

Waterloo met all three criteria with two separate runs, one of which processed queries in an average of less than 0.9 seconds. Both of these query sets were classified as Manual runs but no query generation method restrictions were stated in the medal conditions.

Considering only 100 gB runs which were automatically generated from no more than the T+D fields, six runs were excluded and the median P@20 dropped from 0.525 to 0.442. This left Waterloo run 0 and the ACSys 5-term run exactly on the median.

Accordingly, Waterloo was presented with an ACSys VLC medal during the VLC plenary session at TREC-7.

Acknowledgements

We are very much indebted to Brewster Kahle of the Internet Archive for making available the spidered data (and trusting us with a difficult-to-replace set of tapes) and to Edward King of the Earth Observation Centre for sparing us a considerable amount of his time and expertise in converting tape formats. Thanks also to Donna Harman and Ellen Voorhees of NIST, to John O'Callaghan (CEO of ACSys) and to the TREC Program Committee for supporting the track.

Finally, thanks are due to Sonya Welykyj, Penny Craswell, Nick Clarke and Angela Newey for good work in assessing submissions.

Bibliography

DIGITAL EQUIPMENT CORPORATION. 1998. Digital's Alta Vista search index grows to record heights. `http://www.altavista.digital.com/av/content/pr052798.htm`. Press release.

HARMAN, D. K. Ed. 1992. *Proceedings of the First Text Retrieval Conference (TREC-1)* (Gaithersburg MD, November 1992). U.S. National Institute of Standards and Technology. NIST special publication 500-207.

HAWKING, D. AND THISTLEWAITE, P. 1997. Overview of TREC-6 Very Large Collection Track. In E. M. VOORHEES AND D. K. HARMAN Eds., *Proceedings of the Sixth Text Retrieval Conference (TREC-6)* (Gaithersburg MD, November 1997), pp. 93–106. U.S. National Institute of Standards and Technology. NIST special publication 500-240.

HAWKING, D., THISTLEWAITE, P., AND CRASWELL, N. 1997. *TREC Very Large Collection (VLC) web page.* ACSys Cooperative Research Centre, The Australian National University, Canberra. `http://pastime.anu.edu.au/TAR/vlc.html/`.

HAWKING, D., THISTLEWAITE, P., AND HARMAN, D. 1999. Scaling up the TREC Collection. Accepted by *Information Retrieval* September 1998.

INTERNET ARCHIVE. 1997. Building a digital library for the future. `http://www.archive.org/`.

TOPIC BY TOPIC PERFORMANCE OF INFORMATION RETRIEVAL SYSTEMS

Walter Liggett
National Institute of Standards and Technology
Gaithersburg, MD 20899
walter.liggett@nist.gov

Abstract

Formulation of topic properties is the goal of this paper. These properties are to be used in judging the difficulty of topics appropriate to the Text REtrieval Conference (TREC) ad hoc task. Applying statistical methods to both TREC-6 and TREC-7 information retrieval results, we identify topic pairs that exemplify topic properties useful in relating topic statements to system performance. From two topics that seem the same with respect to the challenge they provide information retrieval systems, we formulate topic properties by relating the corresponding topic statements to what is known about information retrieval systems. Some properties apparent in the topic pairs identified are linked to topic expansion. These pairs exemplify both the need for expansion and the danger in automatic expansion.

1. Topic Properties

System developers would like to be able to judge topic difficulty from reasoning involving topic properties, but appropriate topic properties have not yet been defined or even conceived. For example, a topic property or perhaps a set of topic properties is needed to describe whether only a few key words or several sentences are needed to narrow the topic sufficiently. With a better understanding of topic properties, a system developer might be better able to instruct users on creation of the topic statement, to build a better user interface, or to configure a system that bases topic expansion on topic type.

There is some doubt that formulation of topic properties for these purposes is even possible. "Little is known about what makes a topic difficult," conclude Voorhees and Harman in their TREC-6 overview (1998). Yet, reading papers in this (TREC-7) Proceedings that describe systems for the ad hoc task, one sees discussions of topic expansion illustrated with specific topic statements. Clearly, the authors of these papers see certain topic statements as examples of the topic properties that underlie their expansion strategies. The Query track in TREC-7 is intended to help with topic property formulation. Our analysis of the ad hoc task shows that one can connect system performance to generic topic properties.

The effect of topic properties on system performance depends on the document collection. One consequence is that our statistical methods might connect two topics not because the topic statements share some topic property but because the document collection has some unexpected characteristic. Another consequence is that topic properties formulated through the TREC ad hoc task might not be as useful with other document collections. It is hoped that one can guard against these consequences by requiring that the topic properties formulated seem independent of document collection.

For each topic in the ad hoc task, the TREC evaluation provides system performances, a collection of numbers that might be termed a performance profile. (As "systems," we include what others might call system variants, alternative runs with different configurations of the software created by some group.) These profiles are a partial answer to the question, "How do topics differ?" However, the information retrieval community needs a more general answer, one that can be applied to any topics in the style of the ad hoc task. The purpose of this paper is display of some TREC data in a way that might crystalize the concepts required for such an answer.

This paper presents a pair of TREC-6 topics and three pairs of TREC-7 topics for the reader to study. It is hoped that the reader will be able to offer an opinion on the topic properties each pair exemplifies. These particular pairs occupy places in the data that seem to recommend them for careful study. First, each pair is unusual in that the two performance profiles differ from the average profile for all topics. Second, within each pair, the two topics have similar profiles so that the question of what the two topics have in common is intriguing. Third, being unusual and similar in these senses holds for both performance measures considered. To the study of these topics, the user must bring knowledge of how information retrieval systems operate. For example, the reader might consider how systems perform topic expansion.

Presentation of the four pairs involves two alternative measures of system performance and a method for decomposing a two-way table, namely, the system-by-topic table of performance measurements. One performance measure considered is average precision, which is familiar to TREC participants. The basis of the other is the depth at 25 percent recall, which is the document rank at which 25 percent of the relevant documents have been found. Use of both of these measures, which are detailed in Section 3, seems beneficial because these measures behave differently. The method for table decomposition is the one that underlies two-way analysis of variance. The components in the decomposition describe performance averaged over topic and over system as well as aspects of the system-topic interaction.

This paper is organized to help the reader focus on the correspondence between topics and observed system performance. In Sections 2 through 4, we present two topic pairs for consideration without going into all the statistical details of their selection. In Sections 5 and 6, we provide these

details. In Section 7, we present results for two more topic pairs. Finally, in Section 8, we draw some conclusions about the formulation of topic properties.

2. Topic Pairs for Study

Statistical analysis suggests the following two pairs of TREC-7 topics for careful study, topics 372 and 379 for study with respect to the 40 best systems and topics 372 and 391 for study with respect to the 14 best automatic systems that use the title, description, and narrative parts of the topic statement. These three topics are

Number: 372
Title: Native American casino
Description: Identify documents that discuss the growth of Native American casino gambling.
Narrative: Relevant documents include discussions regarding Native American casino gambling: its social implications, effects on local and Native American economies, and legal aspects related to Native American tribal autonomy.

Number: 379
Title: mainstreaming
Description: Identify documents that discuss mainstreaming children with physical or mental impairments.
Narrative: A relevant document will include the pros and cons of mainstreaming children with physical or mental impairments, the benefits to the impaired child, as well as the attitude, beliefs and concerns of teachers and school administrators with regard to taking time away from the "normal children."

Number: 391
Title: R&D drug prices
Description: Identify documents that discuss the impact of the cost of research and development (R&D) on the price of drugs.
Narrative: Documents that describe how any aspect of the development of a drug affects its price are relevant. Documents that discuss other factors that affect drug prices, such as advertising, without also discussing R&D costs, are not relevant.

Solely on the basis of reading these topics, one might guess that the shared topic property that dominates system performance involves words not in the topic statements that most people would associate with the topics based on their knowledge of current events. For example, association of "Native American" with "reservation" would be helpful in retrieving documents relevant to topic 372. Inclusion of the phrase "public education" might be a useful addition to "mainstreaming" in retrieving documents for topic 379. The words "R&D," "drug," and "price" have variants such as "biotechnology" and "return on investment" that would be useful in a search on topic 391.

The reason that statistical analysis of system performance connects these topics is the appearance of common system successes and common system failures in expansion of these topics. In particular, manual systems seem to do relatively well with topics 372 and 379 whereas automatic systems do relatively poorly.

In trying to conceive of the topic property common to the members of these pairs, one must also recognize other topic properties in which these topics differ. For example, on one hand, "Native American" and "mainstreaming" are terms that must be interpreted only according to their specialized meaning if the search is to be completely successful. On the other hand, to "R&D," "drug" and "price" there correspond equivalent phrases that could be substituted and that must be recognized in a successful search. Conceptualization of a topic property requires attention to such differences.

3. Data Analysis

Comparison of two topics in search of a common topic property involves for each topic, the statement and the performance for a group of systems (or, as some might prefer to say, system variants). We depict performance versus system graphically. As detailed in this section, we use two performance measures, average precision and depth at 25 percent recall. Moreover, we divide performance into components and graph the overall component and the distinctive component separately.

Consider first the determination of average precision. Although the reader may be familiar with this measure through TREC publications, a somewhat different account seems useful because it facilitates comparison with our other measure. Performance for a particular system and topic is based on 1000 documents that the system has identified and ranked according to relevance to the topic. Both performance measures are computed from the ranks of documents deemed relevant by the assessor. In increasing order, we denote these ranks by r_1, r_2, ..., r_i, ... The ratio i/r_i might be regarded as an estimate of the rate at which relevant documents are discovered. Except for adjustment for the relevant documents not discovered, the average precision is the average of these rate estimates. The adjustment consists of regarding the undiscovered relevant documents as having infinite rank, $r_i = \infty$. Thus, if there are n_R relevant documents of which n are discovered by a system, the average precision for that system is given by

$$P = \frac{1}{n_R} \sum_{i=1}^{n} \frac{i}{r_i}.$$

Measures like depth at 25 percent recall have been suggested for example, by E. M Keen (1997). The determination of this depth involves at most two ranks. Roughly, this depth is the rank r_q, where q is the integer part of $.25n_R$, $q = [.25n_R]$. If $\Delta = .25n_R - q$ is greater than 0 and if $q < n$, then we interpolate using $(1 - \Delta) r_q + \Delta r_{q+1}$. If $q > n$, then we compute the extrapolation $.25n_R r_n / n$, and if its value is greater than 1000, we use it. What we do in other cases is shown below. In addition, we subtract $.25n_R - 1$ so that if 25 percent of the relevant documents are found before any other documents, the measure does not depend on the number relevant for the topic. On the basis of these considerations, our algorithm for depth at 25 percent recall is

$$r_{.25} = \begin{cases} (1-\Delta)\, r_q + \Delta\, r_{q+1} - (.25 n_R - 1) & \text{if } 0 < q < n \\ \max[.25 n_R\, r_n / n,\ (1-\Delta)\, r_n + \Delta\, 1001] - (.25 n_R - 1) & \text{if } 0 < q = n \\ \max[.25 n_R\, r_n / n,\ 1001] - (.25 n_R - 1) & \text{if } 0 < n < q \\ \infty & \text{otherwise} \end{cases}$$

Assignment of ∞ to the case in which the system returns no relevant documents, $n = 0$, requires comment. As detailed below, we limit our analysis to the better systems and to easier topics so that $n = 0$ occurs infrequently. In the analysis reported below, a few cases remain and for these we let $r_{.25} = 1500$. This artifice might not be satisfactory were there more than a few. Our analysis is based not on $r_{.25}$ but on $-\log_{10}(r_{.25}) = \log_{10}(1/r_{.25})$. Use of the logarithm reduces the influence of the larger ranks, and use of the minus sign causes larger values to correspond to better performance as is the case with average precision.

These performance measures depend differently on how many relevant documents there are in the collection. If one were to remove half the relevant documents from the collection, the average precision would be more or less cut in half whereas the depth at 25 percent recall would be nearly unchanged. Moreover, undiscovered relevant documents are treated differently in determining each measure. One might argue that one measure is better than the other, but consideration of both seems better than choosing one.

For each performance measure, we compare topics by means of two graphs, one showing the overall component and the other showing the distinctive component. These components are computed from the performances for all topics and systems. One graph compares the two topics in terms of the overall abilities of the systems. The second graph compares the two topics in terms of deviations from these overall abilities. What is perhaps most interesting is the topic-to-topic similarity of the distinctive components as shown in the second graph.

Computation of the overall component requires three steps. Let the number of systems be N_s, the number of topics be N_t, and the performance measure for system i and topic j be y_{ij}. The difficulty of topic j is

$$\hat{\alpha}_j = \frac{1}{N_s} \sum_{i=1}^{N_s} y_{ij}$$

and the (centered) average performance of system i is

$$\hat{x}_i = \frac{1}{N_t} \sum_{j=1}^{N_t} (y_{ij} - \hat{\alpha}_j).$$

In addition, we include a term that describes variation from topic to topic in the effect of overall system abilities:

$$\hat{\beta}_j = \frac{\sum_{i=1}^{N_s} (y_{ij} - \hat{\alpha}_j - \hat{x}_i)\, \hat{x}_i}{\sum_{i=1}^{N_s} \hat{x}_i^2}.$$

One can think of $\hat{\beta}_j$ as representing the degree to which topic j can distinguish the overall abilities of the systems. One can also think of $\hat{\beta}_j$ as representing the degree to which system features that contribute to performance over all topics are effective with topic j. The overall component is given by

$$\hat{\alpha}_j + (1 + \hat{\beta}_j)\, \hat{x}_i.$$

What is portrayed by this component is somewhat familiar to TREC participants.

What is perhaps of more interest is the remainder

$$y_{ij} - \hat{\alpha}_j - (1 + \hat{\beta}_j)\, \hat{x}_i.$$

from which we determine the distinctive component. This remainder reflects interactions, cases where after adjustment for overall performance, one system is better than another for one topic but not for another topic. For example, a system that makes use of only the topic title might do relatively well with titles that adequately delimit the subject but do relatively poorly with diffuse titles. Interactions suggest improvements because they suggest that a system might be assembled from the better parts of existing systems.

The remainder is, however, not easy to interpret because it is noisy. Some appearances of interaction do not reliably predict the results of subsequent evaluations and are not a sound basis for system development. In Section 5, we discuss how we reduce the noise in this remainder and thereby determine the distinctive component. Also, we discuss how we select topics that are worthy of study because their behavior is predictive of future evaluations. Note that adding the remainder to the overall component gives the original performance data. Thus, except for some noise, adding the distinctive component to the overall component gives the original data. The reader will sometimes want to consider the sum of these components.

For our analysis, we select only better systems because differences between good systems are more interesting than other differences. We select the better systems using the median to summarize performance over the topics. The reason for choice of the median is that we would be interested in a generally good system that did poorly on occasion. We adjust the system-topic performance for the overall difficulty of the topic and then find medians over the topics. The algorithm we actually use is Tukey's median polish (Velleman and Hoaglin, 1981). Since we want the same set of systems for analysis by each performance measure, we select the better systems on the basis of depth at 25 percent recall.

Having selected a set of systems, we then select a set of topics. First, we select only topics with more than 10 relevant documents. Second, for the TREC-6 data analyzed in Section 7, we count for each topic the number of selected systems with $r_{.25}$ greater than 2000. We select topics for which this count is no greater than 5. The purpose of eliminating topics is to prevent a few very difficult topics, especially ones for which several systems found no relevant documents, from obscuring more general behavior.

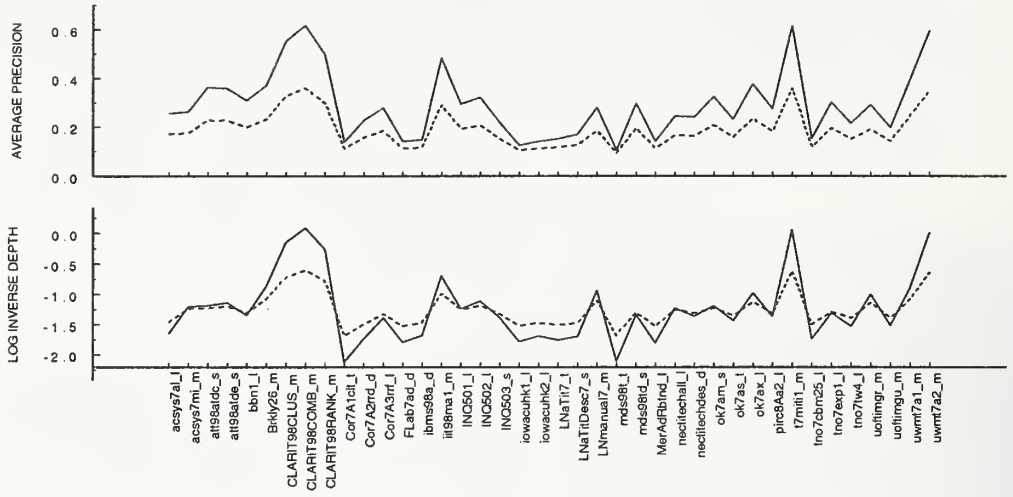


Figure 1. Overall component for topics 372 (solid line) and 379 (dashed line).

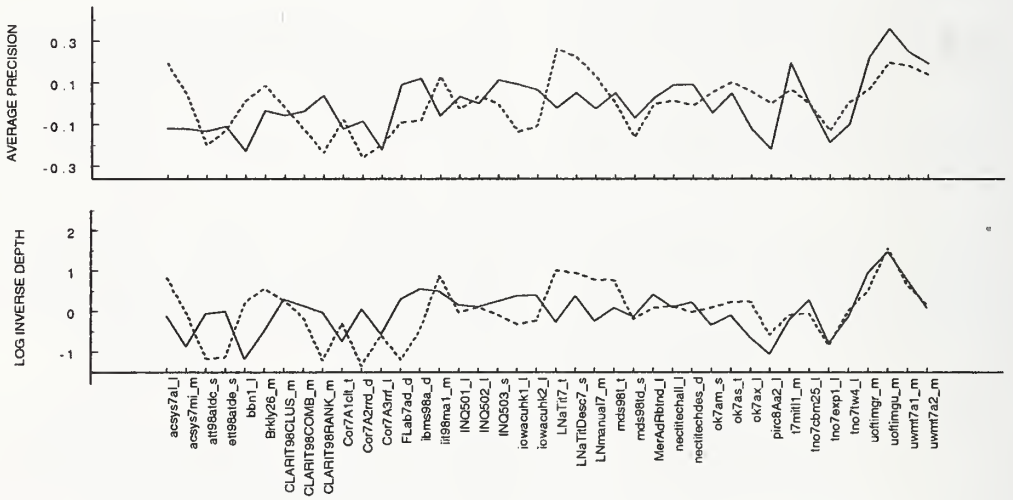


Figure 2. Distinctive component for topics 372 (solid line) and 379 (dashed line).

4. An Interpretation of the Data

Figures 1-6 show performance versus system as detailed in Section 3. In terms of abbreviations explained elsewhere in this Proceedings, system names are given on the horizontal axis with query type appended to each system name: "t" for title only, "d" for description only, "s" for title and description, "l" for title, description and narrative, and "m" for manual. Either average precision or the logarithm of the inverse of the depth is given on the vertical axis. It is the distinctive components shown in Figs. 2, 4, and 6 that suggest that the topic pairs 372-379 and 372-391 share the need for effective topic expansion. Nevertheless, we first consider the overall component shown in Figs. 1, 3 and 5.

In Figs. 1, 3 and 5, we observe that the system-to-system variation in the overall component is larger for one topic than the other. System-to-system variation in the overall component reflects the average performance over all topics. Thus, a topic that varies more with system is related more strongly to average performance. In other words, whatever makes a system perform better or worse for all topics has a greater effect for this topic. Conversely, the topic that varies less with system can be considered to be more singular, less related to all the other topics. In Figs. 1 and 3, we see that topic 379 (mainstreaming) is more singular than topic 372 (Native American casino). In Fig. 5, we see that topic 372 is more singular than topic 391 (R&D drug prices). These observations seem reasonable.

Another thing of note in these figures is that when average precision is considered, one topic has better performance than the other for all systems but when depth at 25 percent recall is

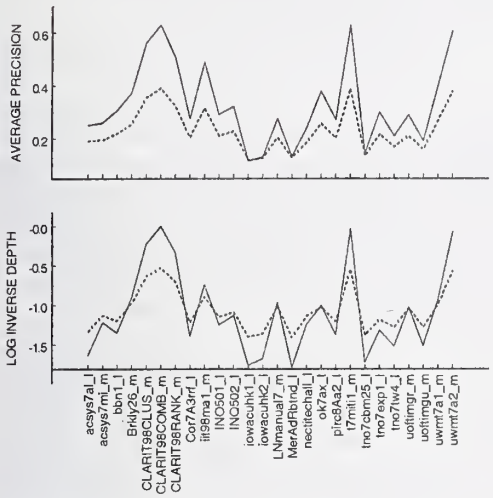


Figure 3. Overall component for topics 372 (solid line) and 379 (dashed line).

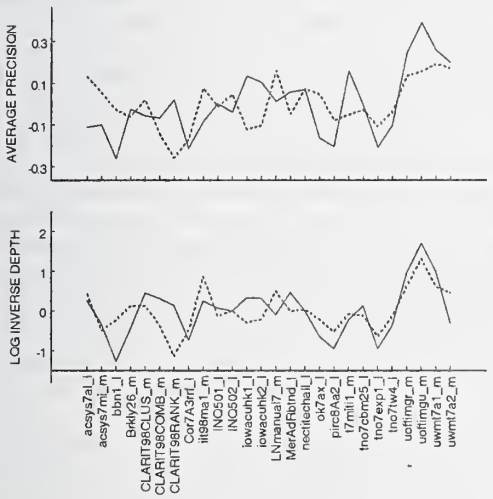


Figure 4. Distinctive component for topics 372 (solid line) and 379 (dashed line).

considered, this does not hold. A topic that has better performance for all systems can be regarded as an easier topic. The discrepancy between the two measures can be largely explained by the dependence of average precision on the number of relevant documents. The number of relevant documents is 16 for topic 379, 49 for topic 372, and 178 for topic 391. We conclude that there is little in these figures to suggest that one topic in a pair is easier than the other.

Figures 2, 4, and 6 show that the topic pairs 372-379 and 372-391 share one or more topic properties related to challenges in topic expansion. This is most clearly shown in Fig. 4 where manual systems largely outperform automatic systems. Presumably, manual systems provide more effective topic expansion. Figure 6, which is based only on automatic systems, shows that some systems perform better than others. One would guess that this is due to better topic expansion.

The figures in this paper reflect the three criteria we use in selecting pairs. Consider Figs. 1 and 2. First, one can see that these topics are unusual by comparing Figs. 1 and 2 and noting that the variation from system to system in the distinctive component is roughly the same size as the variation in the overall component. For a nearly average topic, the variation in the distinctive component would be much smaller. Second, the two topics are at least somewhat similar as shown in Fig. 2 by the fact that the distinctive components largely vary together. Third, this similarity holds for both performance measures as also shown in Fig. 2.

There are further remarks one can make about Figs. 1 and 2. Beyond what has already been said about Fig. 1, note that the two performance measures show the same pattern of system-to-system variation. The agreement between the topics as shown in Fig. 2 is not as compelling as one might like. The agreement seems better on the right than the left. There are particular systems that show agreement such as the system "uoftimgu," which is notable because the distinctive component is high for both topics, and the system "tno7exp1," which is notable because the distinctive component is low for both topics. Also notable are the results for the systems "bbn1" and "CLARIT98RANK" for which there is disagreement. It seems possible to infer the reasons for these disagreements from the papers on these systems elsewhere in this Proceedings. Note in particular the comparison of "CLARIT98CLUS" and "CLARIT98RANK." Generally, one might guess that agreement is not better because all the different query types are included.

We repeated our analysis for just the 26 systems in Figs. 1 and 2 that make use of the entire topic statement, query types "l" and "m." Again, our statistical method chose topics 372 and 379 as different from the rest and similar to each other. The overall component in Fig. 3 is not remarkably different from what is shown in Fig. 1. The distinctive component in Fig. 4 shows a group of manual systems for which relative performance is high for both topics and a group of automatic systems for which relative performance is low for both topics. This seems to be evidence that the need for topic expansion noted in Section 2 can be more effectively achieved with a manual system. Conversely, the topic property that may be inferred from this pair of topics is the need for a type of topic expansion that can be better provided by human interaction with the system than automatically.

Seemingly of interest would be the automatic systems that use the entire topic statement. We repeated the analysis for the 14

systems in Figs. 1 and 2 denoted "l". The two topics that emerged from this analysis are 372 and 391. One would expect a different pair of topics because the choice of pair depends on the systems that enter the analysis. Figures 5 and 6 show the overall and distinctive components for these two topics. Observations on Fig. 5 have largely been covered by our general remarks on the overall component. Figure 6 shows 6 systems, "INQ501", "INQ502", "iowacuhk1", "iowacuhk2", "MerAdRbtnd", and "nectitechall", that did relatively well with both of these topics. Rather than guessing what the underlying

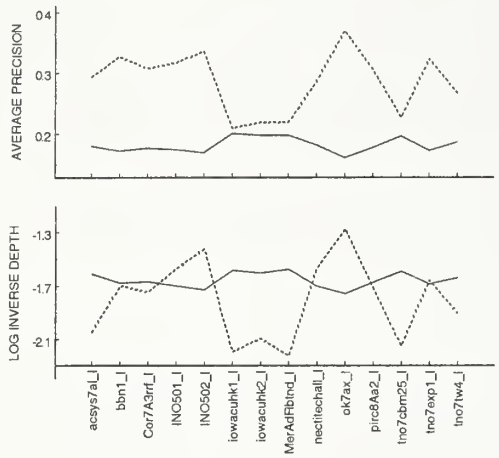


Figure 5. Overall component for topics 372 (solid line) and 391 (dashed line).

topic property is, we ask the following interesting question: What features of these systems causes them to favor topics 372

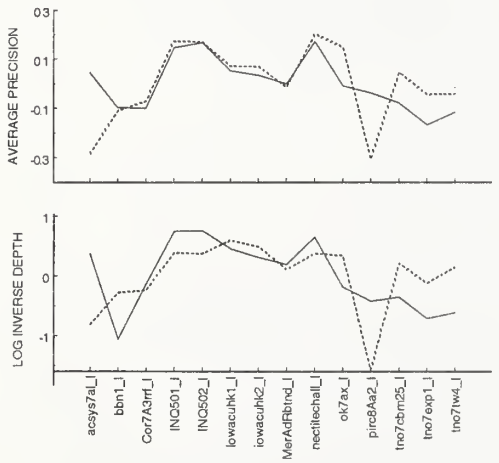


Figure 6. Distinctive component for topics 372 (solid line) and 391 (dashed line).

and 391 when overall the performance of these systems is variable, both better and worse than the average? This is the kind of connection between topic and system that could lead to system improvements. Figure 6 also shows two systems, "acsys7al" and "pirc8Aa2", that did relatively better with topic 372 than topic 391. Did the topic expansion strategies used by these systems fail when applied to topic 391?

The interpretation of Figs. 1-6 is not yet complete because the specific features of the systems portrayed have not yet been fully considered. Some further information useful for interpretation may be included in the system descriptions presented elsewhere in this Proceedings. In any case, those responsible for specific systems may be able to add to the above interpretation.

5. The Distinctive Component

Our argument for studying the topics discussed in Section 2 as well as other topics selected in Section 7 is not based on understanding of information retrieval systems but on statistics. We now present statistical methodology for selecting topics. The purpose of this methodology is interpretation of the $N_s \times N_t$ matrix with elements

$$y_{ij} - \hat{\alpha}_j - (1 + \hat{\beta}_j)\,\hat{x}_i.$$

In interpreting these residuals, one faces the challenge of identifying what is substantial while disregarding appearances that might not generalize beyond the immediate data.

What are we looking for in this matrix? Say that the matrix were given by the product of two column vectors, γh^T, where superscript T denotes transpose. Were this true, the distinctive component for topic j would be $h_j \gamma$. Thus, except for a multiplicative constant, the distinctive component would be the same for each topic. This would imply the existence of a topic property that has for topic j the value h_j (on a scale that is arbitrary up to a linear transformation). We see that interaction of this sort leads directly to a topic property of the type we seek.

Opposite the case of pure interaction is the case of pure noise, the case in which the residual matrix suggests no topic properties that apply beyond the performance of a single system. Banks, et al. (1999) consider this case. They indicate that if the coefficients $\hat{\beta}_j$ are significantly different from zero, then one should conclude that the residual matrix will likely exhibit some interactions. We have included the $\hat{\beta}_j$ coefficients in the overall component. Figs. 1, 3, and 5 suggest that these coefficients are significant. We have applied the appropriate hypothesis test to confirm this.

Beyond the question of whether the residual matrix appears to be pure noise is the possibility that one will over-interpret this matrix. Perhaps our best defense against this possibility is the use of two different performance measures. One might guess that because these measures are calculated differently, the measurement errors for each are largely independent. As our defense, we recommend a pair of topics for study only when

they appear as worthy of study according to both measures.

The singular value decomposition provides an analysis of the residual matrix

$$y_{ij} - \hat{\alpha}_j - (1 + \hat{\beta}_j)\hat{x}_i = \sum_{m=1}^{M_0} d_m u_{im} v_{jm}.$$

The upper limit on the sum M_0 is the smaller of $N_s - 2$ and $N_t - 1$. The coefficients d_m are positive and are ordered according to decreasing size. The vectors $\boldsymbol{u}_m = (u_{im})$ are orthonormal and orthogonal to the vector $\mathbf{1} = (1)$ as are the vectors $\boldsymbol{v}_m = (v_{jm})$. The vectors $\boldsymbol{u}_m$ are also orthogonal to the vector with elements $\hat{x}_i$. In vector notation, the residual matrix is given by

$$\sum_{m=1}^{M_0} d_m \boldsymbol{u}_m \boldsymbol{v}_m^T.$$

Note that the decomposition is a sum of interaction-like terms. If only one term were non-zero, then we would have the case of pure interaction discussed above.

Our approach to separating the distinctive component from the noise is separation of the larger terms in the decomposition from the rest. Based on the size of the coefficients d_m, we choose M terms for the distinctive component, which corresponds to supplanting the residual matrix with a smoothed residual matrix

$$\sum_{m=1}^{M} d_m \boldsymbol{u}_m \boldsymbol{v}_m^T.$$

Figures 2 and 4 are based on $M = 7$; Fig. 6 is based on $M = 5$; and the figures in Section 7 are based on $M = 4$. There is a considerable literature on choosing the number of common factors in factor analysis that applies to the choice of M. In the current situation, we expect that the results will not be particularly sensitive to the choice. We could vary M and see how this affects our choice of topics for study.

One way of thinking about explaining the smoothed residual matrix is to think about reducing the sum of squares of its elements. Each column in this matrix corresponds to a topic and thus to the distinctive component for that topic. We see that the distinctive component is the sum of M terms $d_m v_{jm} \boldsymbol{u}_m$. The sum of squares for each term is $d_m^2 v_{jm}^2$, and the sum of squares for the distinctive component for topic j is

$$\sum_{m=1}^{M} d_m^2 v_{jm}^2.$$

Summing over all the topics, we obtain

$$\sum_{m=1}^{M} d_m^2.$$

We choose topics on the basis of the degree to which each explains this total sum of squares.

Retained in the smoothed residual matrix are the interactions that are of interest, but, unless $M = 1$, we must do more to bring out the character of the interactions. We would like to find individual and pairs of topics that are strongly associated with the interactions. Such topics would appear in the smoothed residual matrix as a term of the form $\gamma \boldsymbol{h}^T$, where $\boldsymbol{h}$ contrasts one or two topics with all the other topics. We choose γ so that $\gamma \boldsymbol{h}^T$ matches the smoothed residual matrix as closely as possible. We take $\boldsymbol{h}$ to be a unit vector that is orthogonal to $\mathbf{1}$, $\boldsymbol{h}^T\boldsymbol{h} = 1$, $\boldsymbol{h}^T\mathbf{1} = 0$. A gauge of this match is given by

$$\sum_{m=1}^{M} d_m^2 - \sum_{i=1}^{N_s} \min_{\gamma_i} \left[\sum_{j=1}^{N_t} \left(\sum_{m=1}^{M} d_m u_{im} v_{jm} - \gamma_i h_j \right)^2 \right].$$

The first term in this gauge is the sum of squares of the elements in the smoothed residual matrix.

The most that the total sum of squares can be reduced by a rank one approximation is d_1^2. This would be obtained with $h_j = v_{j1}$ and $\gamma_i = d_1 u_{im}$.

Working through the minimization that is part of computing our gauge and dividing the result by d_1^2, we obtain what we call the fraction explained by the contrast $\boldsymbol{h}$

$$\sum_{m=1}^{M} d_m^2 (\boldsymbol{v}_m^T \boldsymbol{h})^2 / d_1^2.$$

We begin by asking which topic explains the largest part of the total sum of squares. To compute this for topic j, we let $\boldsymbol{h} = (h_k)$, where

$$h_k = \begin{cases} \sqrt{1 - 1/N_t} & \text{if } k = j \\ -1/(N_t\sqrt{1 - 1/N_t}) & \text{if } k \neq j \end{cases}.$$

The topic that explains the largest amount, we term the most unusual topic, because it corresponds most closely to the part of the residual matrix that exhibits the important interactions.

One could ask what topic property causes the most unusual topic to be so. Generally, however, the answer would be a list of possible topic properties ranging from the subject matter of the topic to the phrasing used to convey the topic. Asking what is in common between a pair of topics is more likely to be fruitful. We ask what contrast between two topics and the others explains a large part of the total sum of squares. To compute this for topics j and j', we let

$$h_k = \begin{cases} \sqrt{1/2 - 1/N_t} & \text{if } k = j \\ \sqrt{1/2 - 1/N_t} & \text{if } k = j' \\ -1/(N_t\sqrt{1/2 - 1/N_t}) & \text{otherwise} \end{cases}.$$

As is the case for the topics discussed in Section 4, the fraction explained for some pairs of topics is larger than for the most unusual single topic. Topic pairs for which this is true are both strongly associated with the interactions and have similar distinctive components. If the distinctive components were not similar but each topic were by itself unusual, the contrast for the pair would likely not explain much of the smoothed residual matrix because the two topics would partially cancel each other.

With one minor exception, each topic pair in Section 4 explains, according to both measures, more than the most unusual topic. The existence of such pairs provides a strong incentive for study

of the question of what the topics have in common. However, there is no guarantee that such a pair will occur. For the systems and topics considered in Section 4, there were other topic pairs that explained more than the most unusual topic but these did not turn up for both measures. We would like a pair chosen for study to be unusual but requiring that the pair explain more than the most unusual topic may be too stringent. Perhaps if the amount explained were within 0.10 of the most unusual topic, this would be enough.

6. Steps in Choosing Topic Pairs

Sections 3 and 5 discuss the statistical methods that we use to choose topic pairs. In this section, we detail the application of these methods that produced the topic pair 372-379 shown in Figs. 1-2.

Clearly, a comparison of topics based on system performance depends on the set of systems considered. There are 103 systems that produced TREC-7 ad hoc task results. We chose 40 for Figs. 1 and 2, 26 for Figs. 3 and 4, and 14 for Figs. 5 and 6. We chose the systems with the best performance (according to a particular criterion) because we believe that in general such a choice produces the most interesting results. On the other hand, there is no reason why the methods described in this paper should not be applied to other sets of systems.

Having chosen a set of systems, we eliminate some topics in part to accommodate our depth measure and in part to eliminate difficult topics that might have performance profiles very different from other topics. In choosing the topic pairs for Figs. 1-6, we eliminated topics 361 and 380 because, for each of these, the number of relevant documents is less than 10. For the analysis of the TREC-6 data in Section 7, we eliminated 10 topics, some because of the number of relevant documents and some because too many systems found no relevant documents for the topic.

Computation of the residual matrix is detailed in Section 3. In the analysis for Figs. 1 and 2, application the singular value decomposition to the residual matrix for log inverse depth gives as the coefficients d_m, 6.81, 6.58, 5.78, 5.70, 5.38, 4.73, 4.11, 3.85, 3.80, 3.63, 3.36, 3.06, 2.84, 2.63, 2.41, 2.29, 2.13, 1.89, 1.79, 1.74, 1.55, 1.41, 1.36, 1.31, 1.20, 0.97, 0.90, 0.81, 0.78, 0.62, 0.59, 0.58, 0.44, 0.41, 0.38, 0.24, 0.18, and 0.16. Application to the residual matrix for average precision gives 1.70, 1.65, 1.45, 1.30, 1.22, 1.10, 1.03, 0.93, 0.89, 0.84, 0.73, 0.69, 0.65, 0.63, 0.60, 0.52, 0.49, 0.45, 0.43, 0.40, 0.36, 0.33, 0.30, 0.29, 0.24, 0.24, 0.20, 0.18, 0.18, 0.14, 0.12, 0.10, 0.09, 0.08, 0.08, 0.06, 0.04, and 0.03. Based on the gaps between the seventh, eighth, and ninth coefficients, we choose M = 7 for the smoothed residual matrix for each measure. In some studies, the researcher might choose M with the idea that this is the number of common factors and that each common factor should be seriously considered. We do not do this, although when more has been learned about topic properties, one might proceed in this way.

As discussed in Section 5, our choice of the topic pair 372-379 in the 40 system context is based on values of the fraction explained. For the depth measure, the topic that alone explains the most is topic 379 with fraction explained of 0.39. Combinations of two topics that explain more along with their fraction explained are 372-379, 0.41 and 397-398, 0.48. For average precision, the topic alone that explains the most is topic 398 with fraction explained of 0.32. Combinations of two topics that explain more are 372-379, 0.33; 372-391, 0.33; and 375-398, 0.41. The pair in common between the two measures is 372-379. For this reason, we selected this pair for Figs. 1 and 2. Clearly, one could investigate other pairs. However, real progress may require focus on a single pair until one is convinced that one has gone as far as possible in finding the common property.

7. Two More Topic Pairs

Along the lines of Sections 2, 4, and 6, we now present two more topic pairs. These pairs suggest somewhat different topic properties.

Complementing our choice in Section 4 of automatic systems that use the whole topic, we now choose the 14 automatic systems from Figs. 1 and 2 that use the topic title, the description, or both. The two topics that emerged from this are

Number: 352
Title: British Chunnel impact
Description: What impact has the Chunnel had on the British economy and/or the life style of the British?
Narrative: Documents discussing the following issues are relevant:
- projected and actual impact on the life styles of the British
- Long term changes to economic policy and relations
- major changes to other transportation systems linked with the Continent

Documents discussing the following issues are not relevant:
- expense and construction schedule
- routine marketing ploys by other channel crossers (i.e., schedule changes, price drops, etc.)

Number: 385
Title: hybrid fuel cars
Description: Identify documents that discuss the current status of hybrid automobile engines, (i.e., cars fueled by something other than gasoline only).
Narrative: A relevant document may include research on non-gasoline powered engines or prototypes that may be fueled by natural gas, methanol, alcohol; cost to the consumer; health benefits derived; and shortcomings in horsepower and passenger comfort.

The overall component shown in Fig. 7 implies that topic 385 (dashed line) is easier than topic 352. (The number relevant is 246 for topic 352 and 86 for topic 385.) Note that the weakest performance (over all topics) occurs for two (of the three) title-only systems.

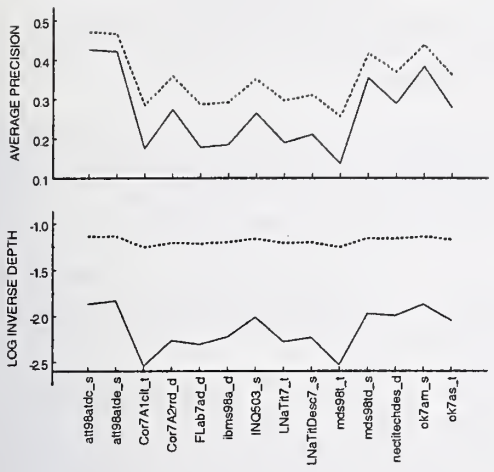


Figure 7. Overall component for topics 352 (solid line) and 385 (dashed line).

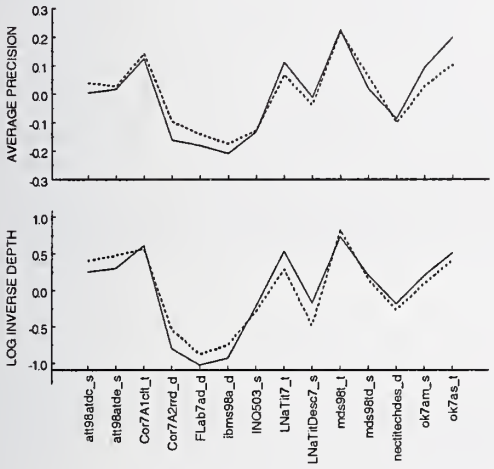


Figure 8. Distinctive component for topics 352 (solid line) and 385 (dashed line).

The distinctive component shown in Fig. 8 exhibits a very close relation between the two topics. The title-only systems perform relatively better than description-only systems in terms of the distinctive component, and, moreover, the title-only runs would perform better if viewed from the sum of the overall and distinctive components. Reasons for this are suggested by the title and description parts of the topic statements. First, for each topic, the key noun phrase differs between title and description. From title to description, topic 352 goes from "British Chunnel" to "Chunnel," and topic 385 goes from "hybrid fuel cars" to "hybrid automobile engines." Second, the more discursive form of the description adds various noun phrases that may do little to make the query more specific. Clearly, the difference in style between the title and the description hurts system performance although a person would say that the description more specifically conveys what is relevant.

For the other pair, we turn to TREC-6. The two topics that emerged are 312 and 316. These topics are

Number: 312
Title: Hydroponics
Description: Document will discuss the science of growing plants in water or some substance other than soil.
Narrative: A relevant document will contain specific information on the necessary nutrients, experiments, types of substrates, and/or any other pertinent facts related to the science of hydroponics. Related information includes, but is not limited to, the history of hydroponics, advantages over standard soil agricultural practices, or the approach of suspending roots in a humid enclosure and spraying them periodically with a nutrient solution to promote plant growth.

Number: 316
Title: Polygamy Polyandry Polygyny

Description: A look at the roots and prevalence of polygamy in the world today.
Narrative: Polygamy is a form of marriage which permits a person to have more than one husband or wife. Polyandry refers to one woman sharing two or more husbands at the same time. Polygyny refers to one man sharing two or more wives at the same time. Primary focus of the search will be the prevalence of these practices in the world today and societal attitudes towards these practices. Also relevant would be discussions of the roots and practical sources of these customs. A modern development in this area is serial polygamy, a phrase coined to label the practice of men who take a series of wives in sequence as a solution to practical welfare, considerations of child care, housing, etc. Documents discussing serial polygamy will not be considered relevant.

The overall component shown in Fig. 9 implies that topic 312 (solid line) is easier than topic 316. The number relevant for topic 312 is 11, and the number relevant for topic 316 is 35.

The distinctive component shown in Fig. 10 shows better performance for four systems, three manual and one automatic. Topics 312 and 316 are notable because of the existence of very specific key words, "hydroponics" and "polygamy." The manual systems that did well seem better able to take advantage of these key words than automatic systems. This may be because a person is better able to see that for these topics, there exist exceptional key words. Automatic systems may go astray by including extraneous words during the process of topic expansion. Voorhees and Harman (1998) note this in their TREC-6 overview.

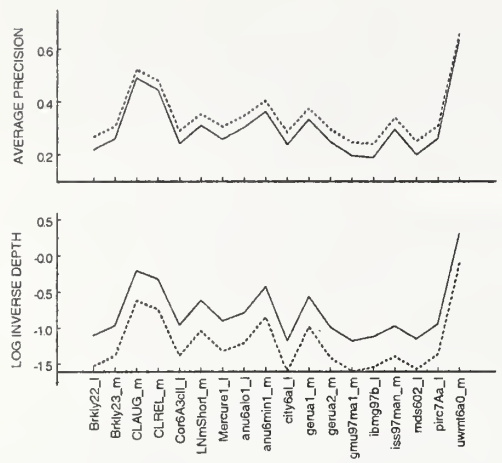


Figure 9. Overall component for topics 312 (solid line) and 316 (dashed line).

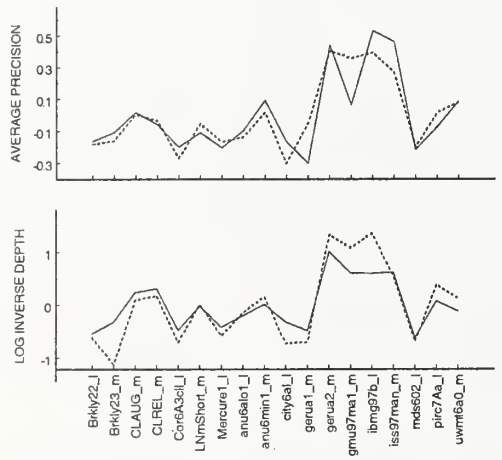


Figure 10. Distinctive component for topics 312 (solid line) and 316 (dashed line).

8. Conclusions

This paper offers four pairs of topics chosen by statistical methods. Faced with the challenge of hypothesizing what each pair has in common, it seems that one has some basis for a response. One topic property that seems to have surfaced is the need for parsimonious topic expansion, and the other is the possibility of confusion when the topic style leads to expansion beyond a succinct delimiting of the topic.

References

D. Banks, P. Over, and N. Zhang (1999). "Blind Men and Elephants: Six Approaches to TREC Data," To appear in *Information Retrieval*.

E. M Keen (1997). "Presenting Results of Experimental Retrieval Comparisons," In K. Spark Jones and P. Willett, editors, *Readings in Information Retrieval*, pages 217-222, San Francisco, CA: Morgan Kaufmann Publishers.

P. F. Velleman and D. C. Hoaglin (1981). *Applications, Basics, and Computing of Exploratory Data Analysis*, Boston, MA: Duxbury Press.

E. M. Voorhees and D. Harman (1998). Overview of the Sixth Text REtrieval Conference (TREC-6). In E. M. Voorhees and D. K. Harman, editors, *Proceedings of the Sixth Text Retrieval Conference (TREC-6)*, pages 1-24, NIST Special Publication 500-240, Washington, DC: U.S. Government Printing Office.

Audio-Indexing For Broadcast News

Satya Dharanipragada, Martin Franz, Salim Roukos

IBM T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

ABSTRACT

In this paper we describe the IBM Audio-Indexing System which is a combination of a large vocabulary speech recognizer and a text-based information retrieval system. Our speech recognizer was used to produce the baseline transcripts for the NIST SDR97 evaluation. We report the performance of the system on the SDR-97 "known item retrieval" task and on a more pertinent TREC-style Audio-Indexing task.

1. Introduction

The goal of an audio-indexing system is to provide the capability of searching and browsing through audio content. The system is formed by integrating information retrieval methods with large vocabulary continuous speech recognition. A large vocabulary continuous speech recognition system is used to to produce time aligned transcripts of the speech. Information retrieval techniques are then employed on these recognized transcripts to identify locations in the text that are relevant to the search request. These locations with time alignments then specify regions of the speech that are relevant for the request.

In this paper we give a description of the speech recognition and information retrieval systems that constitute our Audio Indexing System and report the performance of the system on the SDR97 "known item retrieval" task and on a more pertinent TREC-style Audio-Indexing task.

2. System Description

Our current Audio-Indexing system consists of two components: (1) A large vocabulary continuous speech recognition system, and (2) a text-based information retrieval system. Below we give a brief description of these two components.

2.1. Speech Recognition System

The recognition system is based on the large vocabulary continuous speech recognition system described in [1, 2, 3]. The system uses acoustic models for subphonetic units with context-dependent tying. The instances of context-dependent sub-phone classes are identified by growing a decision tree from the available training data and specifying the terminal nodes of the tree as the relevant instances of these classes. The acoustic feature vectors that characterize the training data at the leaves are modeled by a mixture of Gaussian pdf's, with diagonal covariance matrices. Each leaf of the decision tree is modeled by a 1-state Hidden Markov Model with a self loop and a forward transition. The IBM system expresses the output distributions on the state transitions in terms of the rank of the leaf instead of in terms of the feature vector and the mixture of Gaussian pdf's modeling the training data at the leaf. The rank of a leaf is obtained by computing the log-likelihood of the acoustic vector using the model at each leaf, and then ranking the leaves on the basis of their log-likelihoods.

For this SDR track evaluation we, on purpose, trained our recognizer on data that is different from the SDR (HUB4) training data. This enabled us to judge the performance of our system under mismatch conditions where one wishes to retrieve, as is often the case in practice, spoken documents that come from a domain different from the domain that the speech recognizer is trained on. The acoustic space is parameterized by 60 dimensional feature vectors which are obtained by performing a Linear Discriminant Analysis on a 9 frame window of 24 dimensional cepstral coefficients vectors. The decision tree for identifying the context-dependent sub-phone classes was grown using the WSJ data. The decision tree has around 6000 leaves. An initial set of Gaussians, approximately 35,000 in number, were also trained using the WSJ data. This was our initial system, which we will refer to as **SR-0**. In order to adapt our system to the broadcast domain we ran a MAP adaptation using approximately 1100 studio quality sentences from the 1994 HUB4 (NPR Market Place) data. This system which we will refer to as **SR-1** was used to produce the baseline transcripts for the SDR97 track.

For the language model we use a deleted interpolation trigram model which was also trained on the WSJ corpus with a 64K cased vocabulary. The language model has a perplexity of 253.3 [1] on the WSJ test set.

[1] The language model was trained with the sentence boundary

2.2. Information Retrieval System

An Information Retrieval System typically works in two phases, the *document indexing* phase and *query-document matching* phase. In the document indexing phase each document in the collection is processed to yield a document description, also known as a document-index, which stands in its place during the retrieval. In our system this processing involves part-of-speech tagging of the text, followed by a morphological analysis of the text, followed by removal of function words using a standard *stop word* list. This is in contrast to the simple stemming and filtering used by most of the current systems. Morphological analysis is a form of linguistic signal processing which has great utility in natural language processing. For instance during morphological analysis, among other decompositions, verbs are decomposed into units designating person, tense and mood of the verb plus the root of the verb. Similarly, nouns are decomposed into their roots with (possibly) a tag indicating the plural form. The written request is processed in an identical fashion to yield a *query*. For example, given the request

`Security arrangements in Hebron involving international peace-keepers.`

the following query is obtained after the processing is done.

`security arrangement Hebron to_involve international peace-keepers`

In general our retrieval system uses a 2-pass approach. However, for SDR97 evaluation we employed just the first-pass. In the first pass, given a query, a matching score is computed for each document and the documents are ranked according to this score. The scoring function is simply a weighting scheme that takes into account the number of times each query-term (n-grams in general) occurs in the document normalized with respect to the length of the document. Normalization is essential to remove the bias towards longer documents. The scoring function also favors terms that are specific to a document and thus rare (and hence more significant) across the documents. We use the following version of the Okapi formula [4], for computing the matching score between a document d and a query q:

$$S(d,q) = \sum_{k=1}^{Q} c_q(q_k) \frac{c_d(q_k)}{\alpha_1 + \alpha_2 \frac{l_d}{\bar{l}} + c_d(q_k)} idf(q_k).$$

Here, q_k is the kth term in the query, Q is the number of terms in the query, $c_q(q_k)$ and $c_d(q_k)$ are the counts of the kth term in the query and document respectively, l_d is the length of the document, $\bar{l}$ is the average length of the documents in the collection, and $idf(q_k)$ is the inverse document frequency for the term q_k which is given by:

$$idf(q_k) = log(\frac{N - n(q_k) + 0.5}{n(q_k) + 0.5}),$$

where N is the total number of documents and $n(q_k)$ is the number of documents that contain the term q_k. The inverse document frequency term thus favors terms that are rare among documents. We use a linear combination of unigram and bigram scores in the first pass with weights 0.8 and 0.2 respectively. For unigrams $\alpha_1 = 0.5$ and $\alpha_2 = 1.5$ were used and for bigrams $\alpha_1 = 0.05$ and $\alpha_2 = 0.05$ were used.

In the second pass we re-rank the documents by training a probabilistic relevance model for documents, using the top-ranked documents from the first pass as training data. Details of the second pass can be found in [6].

markers and OOV words included, however, they were not included in the perplexity computation.

3. The VOA Evaluation Corpus

Our Audio-Indexing evaluation corpus consists of approximately 20 hours of radio news broadcasts from the Voice of America covering the time period between May to June 1996. Each day only three broadcasts starting at a different hour and spaced roughly 8 hours apart were downloaded from their internet site. This was done to ensure that the broadcasts are not too similar in content and also to ensure that the collection had several different speakers. The entire collection has about 10 main speakers (both male and female anchors) with several more speakers (correspondents, interviewees etc.) contributing short segments. Each broadcast is typically 6 or 10 mins long and begins with a signature announcement followed by the signature music. A typical news bulletin usually consists of several news stories and often includes reports from correspondents over the telephone line and brief interviews with foreign speakers of English.

The entire speech collection is recognized with a large vocabulary speech recognizer to produce transcripts along with time-alignments for each word in the transcripts. Unlike in the standard information retrieval scenario where the text collection is segmented into documents with each document usually discussing a specific topic/story, story segmentation is not automatically available in this application. We, thus, need a scheme to segment the transcripts into stories. One method is to apply standard topic identification schemes to automatically segment the text into topics, however, a more simplistic solution to this problem is to break the transcript into overlapping segments of a fixed number of words and treat each segment as a separate document. We adopt such an approach in our experiment here, with 100 words in each document, resulting in 3412 documents in the collection.

# queries	53
average length in words	10
average number of relevant documents per query	11

Table 1: Query statistics

3.1. The Test Collection

Evaluating an information retrieval systems requires search requests, together with assessments of the relevance of each document to each of these requests. The search requests were collected from independent sources such as newspapers and other news broadcasts appearing during the same period of time. This method of collecting search requests is similar to the TREC evaluation and in general they form a better test for the information retrieval system than "known item retrieval", where users are asked to compose queries after reading the documents. We compiled 85 requests in this manner. Judging the relevance of each document for each of these queries is a time-consuming task. Instead, we took the following approach. We ran our information retrieval system on the document collection with each of these search requests and made relevance judgment of only the top 30 ranked documents for each query. We found that only 53 of the 85 requests had any relevant documents, which can be attributed to the small size of the database. We discarded the requests that did not have any relevant documents from our evaluation set. The query statistics are shown in Table 1.

4. Performance of the Speech Recognition System

The performance of the above system was tested on a test set composed of two 10 min VOA broadcasts and the results are shown is Table 2. The decoding speed, based on an IBM RS6000/590 machine, is about 30×real-time. On the WSJ test set the above system has a WER of 14.3%.

Corpus	WER (%)
WSJ	14.3
VOA	30.2

Table 2: Performance of **SR-0** on VOA and WSJ.

On HUB4 test data, System-1 had an average WER of about 50%. The WER under different acoustic conditions and speaking styles are shown in Table 3. The higher error rate on the VOA and the HUB4 test sets can

Acoustic/Speaking conditions	WER (%)	# words
Baseline	32.2	120,714
Spontaneous	50.5	88,169
Telephone	63.3	69,595
Speech+Music	64.0	19,903
Degraded acousic conditions	46.9	46,369
Non-native speakers	38.0	1,942
Other	69.6	54,867
Overall	50.0	401,559

Table 3: Performance of **SR-1** on the HUB4 test set.

be attributed to several reasons: (1) the VOA and HUB4 speech has a large proportion of spontaneous speech whereas the WSJ speech is mainly read speech, (2) the VOA speech is of a lower bandwidth (11KHz) and the HUB4 speech has different acoustic conditions than the WSJ speech, and, (3) the language model is not tuned to the VOA or HUB4 corpus.

5. Performance of the Information Retrieval System On Clean Text

For the SDR track, "known item retrieval" performance was evaluated. Overall there were 49 topics or queries and 1451 documents. The performance of our system on reference transcripts is summarized in Table 4

Mean rank:	11.84
Mean reciprocal rank:	0.7923
Known items found at rank:	
≤ 1	37
≤ 5	41
≤ 10	46
≤ 20	47
≤ 100	47
Not found:	0

Table 4: IR system performance on reference transcripts

More often, however, retrieval performance is measured by two measures *precision* and *recall*. Precision is defined as the percentage of the retrieved documents that are relevant to the query and recall is defined as the percentage of the total number of relevant documents that are retrieved. These two measures can be traded off, one for the other. Often a single *average precision* number is computed by first computing the average of the precision at different recall rates for each query, and then by averaging this number across all queries. A more practi-

Total number of documents	Avg. Precision
140	83%
175000	29%

Table 5: IR system performance on TREC4

cal measurement, however, is the precision when a fixed number of documents (often small, between 10 and 20) are retrieved. Another commonly used measure is the rank of the highest-ranked relevant document for each query and the percentage of queries that have relevant documents within a given range of the ranked list of retrieved documents.

We evaluated the performance of our system on a small subset and the entire TREC4 document-collection. The results are tabulated in Table 5.

6. Combining Speech recognition with Information retrieval

The known item retrieval performance on the baseline transcripts produced by **SR-1** is shown in Table 6. In terms of the mean rank, we find a 156% degradation in performance, whereas in terms of the mean reciprocal rank the degradation is 12.6%. This clearly shows that the mean reciprocal rank is a better measure of performance of the system than the mean rank since the mean rank tends to be heavily influenced by outliers [2].

Mean rank:	30.31
Mean reciprocal rank:	0.6921
Known items found at rank:	
≤ 1	30
≤ 5	39
≤ 10	41
≤ 20	42
≤ 100	46
Not found:	0

Table 6: IR system performance on **SR-1** output

We also conducted a TREC-style evaluation of our system using the VOA corpus described in the previous section. All the results reported here are based on the speech recognition system **SR-0**. Figure 1 shows the precision vs recall rate for our audio-indexing system, averaged over the 53 queries. The average pecision after the first pass is computed to be 69.92%. With the second pass the average precision increases to 72.83%, which represents a relative increase of about 4.1%.

As described earlier, another way of presenting the retrieval performance is by plotting the precision vs the number of retrieved documents. This is shown in Figure 2. For example, the precision when the top 10 documents are retrieved is 57.92%. With the second pass this improves to 62.26% which represents a 7% relative improvement in performance.

A third method of measuring the retrieval preformance is by the percentage of queries that have relevant documents within a given range of the ranked list of retrieved documents. This is shown in Table 7. We find, for example, that after the first pass, 87% of the queries have at least one relevant document in the top 5 documents and 96% of the queries have at least one relevant document in the top 10 documents.

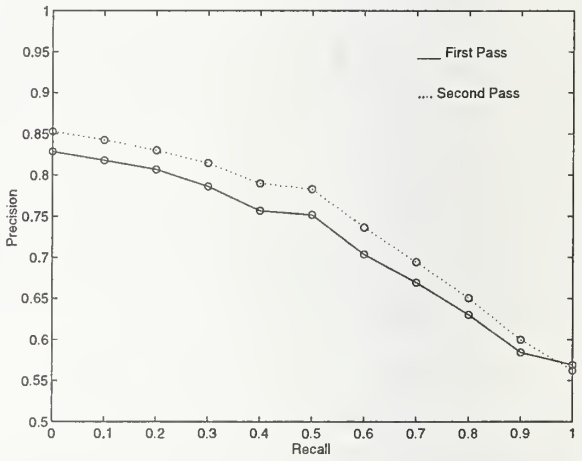


Figure 1: Precision vs Recall rate after the first and second pass.

7. Towards an Open Vocabulary System

One limitation with the current approach to audio-indexing is the finite coverage of the vocabulary used in the speech recognizer – words such as proper nouns and abbreviations that are important from a information retrieval standpoint are often found missing in the vocabulary and hence in the recognized transcripts. One method to overcome this limitation is to complement the speech recognizer with a wordspotter for the out of vocabulary (OOV) words. For this approach to be practical, however, one has to have the ability to detect spoken

[2] The mean rank was also greatly influenced by the assignment of a random rank to the relevant document when several documents shared the same score as the relevant document.

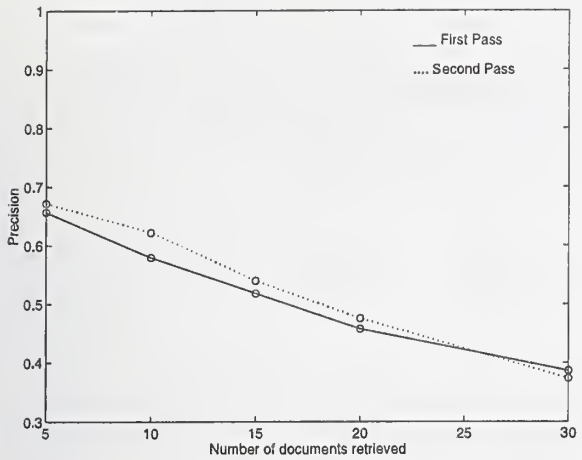


Figure 2: Precision vs number of retrieved documents after the first and second pass.

words in large amounts of speech at speeds many times faster than real-time.

We have developed a novel algorithm that gives us both speed of retrieval and the flexibility of being able to search for any word. We accomplish this by adopting a three-step procedure – a preprocessing step and a two stage search strategy. In the preprocessing step we convert the speech waveform into a representation consisting of a table of phone-ngrams with the times at which it occurs with a high likelihood. This representation allows us to search through the speech very efficiently. The two stage search consists of first a phone-ngram lookup to narrow down the time intervals where the word was likely to have been uttered and then a detailed acoustic match at these time intervals to finally decide more accurately whether the word was actually uttered in that time interval. The algorithm was tested on 10 hours of HUB4 test data using a system trained on the first 50 hours of the HUB4 corpus. On an average, (averaged over 12 OOV words) the detection rate was 48.8% at a false-alarm level of 10 and the average reduction in search was about 80-fold. A more detailed performance analysis is currently being conducted. Details of this algorithm and its performance can be found in [5]. Since there were only 5 OOV words in all topics (queries) of the SDR97 evaluation we did not use our wordspotting component in this evaluation.

Rank (R)	% queries with at least one relevant document in top R ranks
5	86.79%
10	96.25%
15	98.11%
20	98.11%
30	100 %

Table 7: Rank (R) vs percentage queries with at least one relevant document in the top R ranks after the first pass

8. Conclusions and Future work

We presented an overview of our Audio-Indexing System and reported the performance of our system on an audio-indexing task. Our system has an average precision of about 72% with 96% of the queries having a relevant document in the top 10 ranked list. The mean reciprocal rank in a known item retrieval task was 0.69 when the WER was about 50%. We are currently exploring new information retrieval methods that are better adapted to the errorful conditions created by the speech recognizer. Current work is also in progress to augment our system with the new scheme for detecting words that are out of the vocabulary of speech recognizer, yielding a open-vocabulary audio-indexing system.

References

1. L.R. Bahl and P.V. deSouza and P.S. Gopalakrishnan and D. Nahamoo and M.A. Picheny, "Robust methods for context-dependent features and models in a continuous speech recognizer," in *Proc., Intl Conf. on Acoust., Speech, and Sig. Proc.*, 1994.
2. P.S. Gopalakrishnan and L.R. Bahl and R. Mercer, "A tree search strategy for large vocabulary continuous speech recognition," in *Proc., Intl Conf. on Acoust., Speech, and Sig. Proc.*, 1995.
3. L. R. Bahl et al ., "Performance of the IBM large vocabulary continuous speech recognition system on the ARPA wall street journal task," in *Proc., Intl Conf. on Acoust., Speech, and Sig. Proc.*, pp. 41–44, 1995.
4. S.E. Robertson and S. Walker and K. Sparck-Jones and M.M Hancock-Beaulieu and M. Gatford, "Okapi at TREC-3," in *Proc., Third Text Retrieval Conference (NIST special publication)*, 1995.
5. S. Dharanipragada and S. Roukos, "New Word Detection in Audio-Indexing," to appear in *Proc., IEEE Workshop on Automatic Speech Recognition and Understanding*, Santa Barbara, CA, 1997.
6. M. Franz and S. Roukos, "TREC-6 Ad-Hoc Retrieval", to appear *Proc., Sixth Text Retrieval Conference (NIST special publication)*

Deriving Very Short Queries for High Precision and Recall (MultiText Experiments for TREC-7)

Gordon V. Cormack[1] Christopher R. Palmer[1] Michael Van Biesbrouck[1]
Charles L. A. Clarke[2]

MultiText Project
[1] Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada
[2] Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario, Canada
mt@plg.uwaterloo.ca
http://multitext.uwaterloo.ca

Abstract

The main aim of the MultiText experiments for TREC-7 was to derive very short queries that would yield high precision and recall, using a hybrid of manual and automatic processes. Identical queries were formulated for adhoc and VLC runs. A query set derived automatically from the topic title words, with an average of 2.84 terms per query, achieved a reasonable but unexceptional average precision for the adhoc task and a median precision @20 for the 100 GB VLC task. However, these short queries achieved very fast retrieval times — less than 1 second per query over 100 GB using four inexpensive PCs. Two further query sets were derived using post-processing of the results of interactive searching on the adhoc corpus. Queries comprising a single conjunction, averaging 1.86 terms, achieved high precision on both adhoc and VLC tasks, and achieved faster retrieval times than the title-word queries. Compound queries averaging 6.42 terms achieved precision values competitive with the best runs, and retrieval times of 1.51 seconds per query on the 100 GB VLC corpus.

1 Introduction and Background

The MultiText search engine finds passages of text that exactly match a manually or automatically generated boolean query. Documents containing many short matching passages are assumed most likely to be relevant, and are ranked accordingly.

This approach has been found to be effective in several information retrieval contexts: "manual fixed query" (TREC-4 manual adhoc and routing [2]) in which a query is formulated from the topic statement and then run without modification; "interactive query" (TREC-5 adhoc [1]) in which a query is formulated and then refined interactively after viewing the top–ranked documents; "interactive routing" (TREC-5 routing and TREC-6 VLC [1, 5]) in which a query is formulated and then refined interactively after viewing results and judgements on another corpus; "interactive search and judging" (TREC-6 adhoc and high-precision [5]) in which simple queries are formulated, the top-ranked documents are viewed and judged, and the process is repeated – the documents judged relevant are recorded and submitted as the run; "cover density ranking" [3] in which a query is constructed automatically from a very small number of terms.

Each of these approaches offers advantages and drawbacks. Manual fixed queries are appropriate when the search system is slow or off-line, where interaction time is prohibitively expensive, or where the data does not yet exist (e.g. routing or filtering) and no suitable training data is available. Interactive query formulation is likely more appropriate in modern contexts where fast response is available. Yet formulating the "ideal" query is still a difficult task – it is possible to remove poor search terms and to add new ones discovered through interaction, but it quickly becomes difficult to determine whether a given refinement improves or degrades the query. Furthermore, it quickly becomes apparent that many of the same documents are retrieved again and again as the query is revised — to be effective the system must have some provision for previously viewed documents.

This provision for previously viewed documents leads immediately to interactive search and judging. The system cannot simply eliminate previously viewed documents, or the user will be unable to determine whether a refinement improves or degrades a query. Therefore, the user is asked to record a judgement for each viewed document, and these judgements (but not the documents) are reported in subsequent queries. Once judgements are recorded, it would appear that the best approach to adhoc retrieval would be to rank those documents judged relevant first, and to rank those documents judged not relevant last. It turns out that this assumption is not entirely accurate — documents judged not relevant still have a significant probability of being judged relevant in the official qrels — about 10% for our TREC-6 adhoc effort. For routing or filtering, the judgements cannot be used directly, as they apply to documents in the training corpus rather than the test corpus. But they can be used indirectly to test the efficacy of a given query.

All of the methods above require an amount of user interaction ranging from minutes to hours. It is not obvious which approach yields the best results for a given amount of user time: for fixed and interactive manual query construction a fair amount of time is spent composing sophisticated queries; for interactive search and judging, the queries tend to be simple and the majority of time is spent judging documents. It is our impression that interactive search and judging is more effective for any amount of user time, but we have yet to conduct experiments to confirm this impression.

The smallest possible amount of user time is afforded by the use of cover density ranking. A very small number of terms (not greater than 3) are used as a conjunctive query and ranked as described above. If this query yields an insufficient number of documents, a weaker query is constructed automatically. This weaker query consists of a subset of the terms. If there are still an insufficient number of documents, an even weaker query is chosen, and so on. The advantage with this approach is that it requires little user time (none, if you use the title field of a TREC topic). For short queries, cover density ranking yields performance that is comparable to other automatic techniques that do not use pseudo-relevance feedback.

2 TREC-7 Runs

Our primary goal was to construct simple boolean queries that we used for both the adhoc task and Very Large Collection track. Interactive search and judging was used to help develop the queries for the manual runs. The queries that were generated do not include the judgements which allows us to run these queries for the adhoc and VLC tasks. Our official runs were:

`uwmt7a0`,	`uwmt7v0`:	Boolean queries consisting of exactly the title words. Plurals expanded to singular and plural forms. Weaker queries (using subsets of the terms) were derived automatically to be used as necessary for improving recall.
`uwmt7a1`,	`uwmt7v2`:	3-term (or less) boolean queries automatically constructed using the information gathered during interactive search and judging on the adhoc corpus. Weaker queries (with a subset of the terms) derived automatically as necessary to improve recall.
`uwmt7a2`,	`uwmt7v1`:	Larger (6.5 term average) boolean queries automatically constructed using the information gathered during interactive search and judging on the adhoc corpus. No weaker queries used to improve recall.
	`uwmt7v3`:	Title words only; no expansion of plurals; no weaker queries. Not reported further in this paper.

In addition, we conducted a number of unofficial runs:

`uwmt7isj`:	A run consisting of all documents judged "relevant," followed by all documents judged "iffy" followed by all documents judged "not relevant."
`uwmt7a210`:	A run consisting of all documents in `uwmt7a2`, followed by all documents in `uwmt7a1`, followed by all documents in `uwmt7a0`, to a maximum of 1000 per topic.
`uwmt7isj210`:	A run consisting of all documents in `uwmt7isj`, followed by all documents in uwmt7a210, to a maximum of 1000 per topic.

Precision results for the six adhoc and four VLC runs are given in figures 1 through 4. Precision recall graphs are given in figure 5.

3 Title Only Automatic (`uwmt7a0, uwmt7v0`)

For TREC-6, the best automatic runs used only the title terms, expanded using pseudo-relevance feedback. We submitted such a run, achieving an average precision of 0.24. As an unofficial run, we tried cover density ranking with no expansion and achieved an average precision of 0.20. Since our interest this year was in pure boolean queries, we used an enhanced cover density ranking procedure for TREC-7.

We took the title words and expanded all plural words into their plural and singular forms. This expansion used naive suffix matching as shown in figure 6

This was the only form of expansion or stemming performed. The rationale is that we observed from previous TREC efforts that the topics used a collective phrasing when documents describing individual instances were relevant; for example "automobile accidents."

We did not use other forms of stemming because we found previously that stemming compromised early precision, and did not necessarily improve average precision. We did not wish to compromise early precision, especially for our VLC runs.

We constructed weakened queries by taking all subsets of the title terms and ordering them by their inverse frequency of occurrence in the database. For example, topic 366 (commercial cyanide uses) was expanded to:

```
commercial AND cyanide AND (use OR uses)
commercial AND cyanide
cyanide AND (use OR uses)
cyanide
commercial AND (use OR uses)
use OR uses
```

Recall that the second and subsequent queries are used only if the previous query or queries yield insufficient results. Weakened queries were further restricted to be contained in some interval of 128 words. The average number of terms per (un-weakened) query was 2.84, where depluralized words count as 2 terms (e.g. there are 4 terms in the query for topic 366, above).

This technique was applied to the TREC-6 adhoc task yielding an average precision of 0.22; we were hoping to achieve similar results on TREC-7. We have no explanation at this time as to why the average precision was significantly worse (0.19). As expected, on the VLC runs precision @20 improved with collection size, from 0.190 on 1 GB to 0.442 on 100 GB.

4 Interactive Search and Judging

The remaining runs were based on interactive search and judging. For TREC-6, four researchers spent an average of two hours per topic searching and judging documents. Our analysis of this effort lead us to conclude that we could achieve good results with less time [6]. Therefore, we spent less than 30 minutes per topic searching and judging. We tried to find a reasonable number of documents for each topic, but were not exhaustive.

As for TREC-6, documents were judged to be "relevant", "not relevant", or "iffy". In total, we judged 5529 documents: 2561 relevant, 629 iffy, 2339 not relevant. 2237 of the documents we judged relevant were judged also by NIST; 1543 of these (about 2/3) were judged relevant by NIST. This level of agreement is nearly identical to last year. However, NIST judged 2836 documents relevant that we did not judge, and judged relevant a further 133 documents that we judged not relevant. These sets are shown as a Venn diagram in figure 7.

We did not submit ISJ to NIST as an official run. We did, however, evaluate the documents we judged using the official judgements — this run is reported here as `uwmt7isj`. The average precision of 0.3458 is much lower than that achieved by ISJ last year. No doubt this is due to the poorer recall reported above — last year we judged about 2/3 of those documents the NIST found relevant. We were a bit surprised by the low recall of ISJ this year. While we spent considerably less time at it (by more than a factor of 4), we did not feel that we were overlooking large numbers of relevant documents. Apparently our feelings were inaccurate.

We used our ISJ judgements to evaluate `uwmt7a0` (see figure 8). Early precision was significantly better with NIST judgements while average precision and R-precision was significantly better with ISJ judgements. `uwmt7a1` and `uwmt7a2` show much better precision figures, which is no surprise because they were constructed to optimize their performance on these judgements (see next section). `uwmt7isj` and `uwmt7isj210` exhibit perfect scores with respect to this set of judgements because they place all the relevant documents first.

Run	P@1	P@2	P@4	P@20	P@40	RP	AP
uwmt7a0	.5600	.4700	.4700	.3560	.2800	.2289	.1866
uwmt7a1	.7200	.7700	.7000	.5330	.4205	.3402	.2983
uwmt7a2	.7800	.7900	.7300	.5610	.4730	.4012	.3587
uwmt7isj	.8000	.7800	.7400	.6060	.5115	.3952	.3458
uwmt7210	.8200	.7900	.7150	.5620	.4740	.4092	.3868
uwmt7isj210	.8000	.7800	.7400	.6060	.5135	.4384	.4112

Figure 1: Adhoc Runs

Run	P@1	P@2	P@4	P@10	P@20
uwmt7v0	.5000	.4800	.4900	.4680	.4420
uwmt7v2	.6200	.6100	.5950	.5860	.5740
uwmt7v1	.6600	.6700	.6600	.6380	.5980

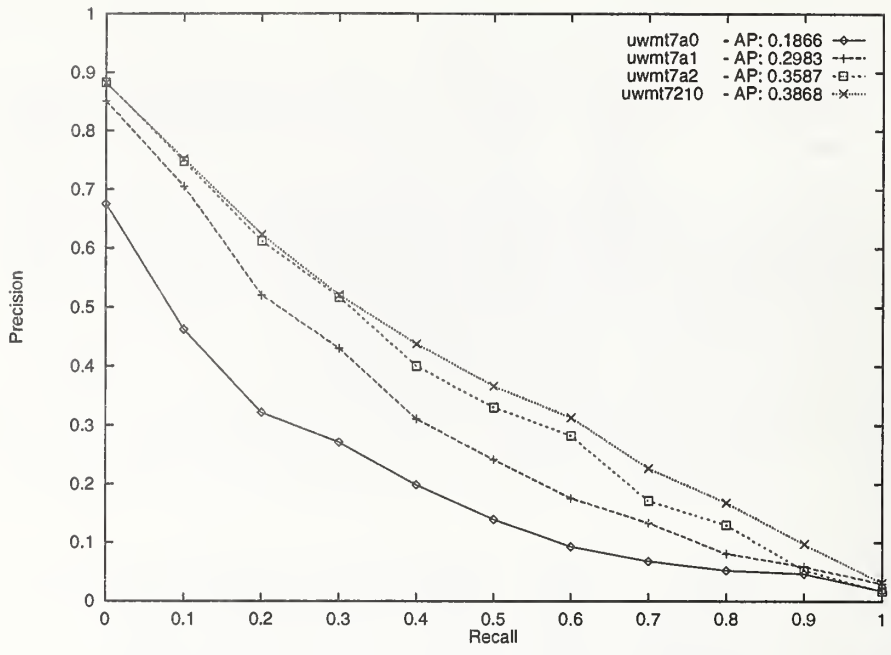
Figure 2: VLC Runs (100 GB Corpus)

Run	P@1	P@2	P@4	P@10	P@20
uwmt7v0b10	.5200	.4500	.4500	.4160	.3690
uwmt7v2b10	.5800	.5500	.5150	.4660	.4110
uwmt7v1b10	.6000	.5900	.5800	.5180	.4740

Figure 3: VLC Runs (10 GB Sample)

Run	P@1	P@2	P@4	P@10	P@20
uwmt7v0b1	.4000	.3200	.3100	.2420	.1900
uwmt7v2b1	.4400	.4000	.3850	.2820	.2230
uwmt7v1b1	.5200	.4500	.4000	.3140	.2350

Figure 4: VLC Runs (1 GB Sample)



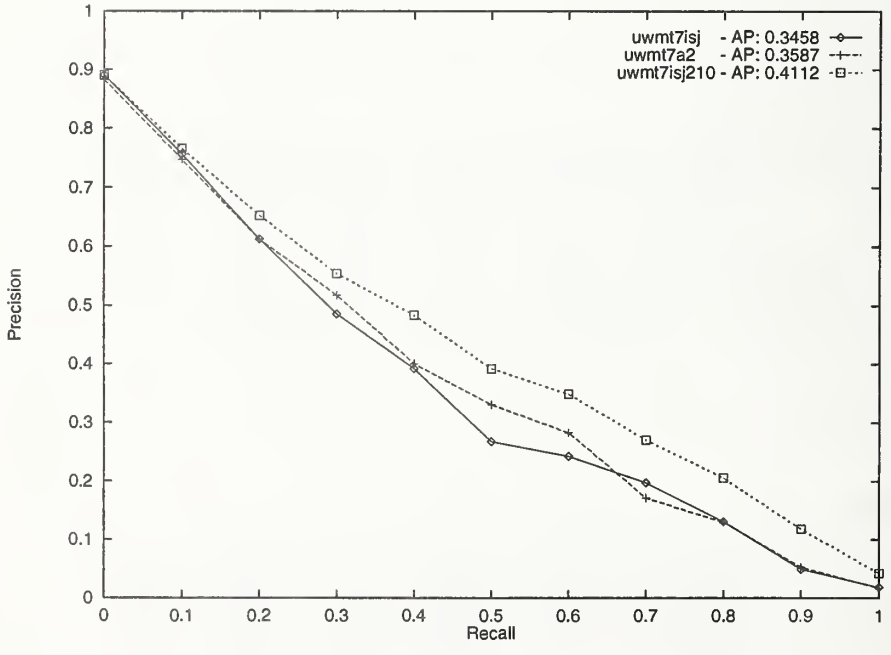


Figure 5: Precision-Recall for Adhoc Runs

5 Automatic Query Extraction

Rather than submit the results of ISJ as an official run, we used an automatic process to extract query terms from the ISJ logs and to build queries that optimized AP on the ISJ judgements.

As mentioned in the introduction, the process of formulating an ideal query can be quite difficult and time consuming. Yet such a query may be desirable for a number of reasons. First, if ISJ is performed on a subset of the corpus, or on a separate training corpus, a query is needed to gather additional documents not subject to ISJ. Second, if ISJ is incomplete, that is, if not all potentially relevant documents are judged, a query may include relevant documents that would otherwise be missed. Third, because of poor judging agreement between searchers and the official judgements, it may be desirable to include documents judged "not relevant" by the searchers.

We had a second purpose in extracting queries from ISJ. We wished to test the hypothesis that queries with a very small number of terms can yield good retrieval performance. It was seen in TREC-6 that the topic titles contained enough information to give the best performance of all automatic approaches. The VLC queries we constructed by hand for TREC-6 were very short and yielded much better performance. We wished to explore the possibility of building still better queries using an automatic optimization procedure.

To this end, we extracted all the queries used in ISJ from our logs. We further reduced each query to sum-of-products form and considered each product separately. We then considered all subsets of each product. For example, the query

```
(amazon OR rain) and forest and (brazil or colombia)
```

would yield:

```
amazon AND forest AND brazil
amazon AND forest AND colombia
rain AND forest AND brazil
rain AND forest AND colombia
amazon AND forest
amazon AND brazil
amazon AND colombia
rain AND forest
rain AND brazil
rain AND colombia
forest AND brazil
forest AND colombia
amazon
rain
forest
brazil
colombia
```

All such queries were evaluated using the ISJ judgements. The run `uwmt7a1` is the single query for each topic that yielded the highest average precision. No query exceeded 3 terms, and the average number of terms was 1.86.

pattern	expansion
*ss	*s, *ss
*sses	*s, *sses
*xes	*x, *xes
*zes	*z, *zes
*ches	*ch, *ches
*shes	*sh, *shes
*eys	*ey, *eys
*ies	*y, *ies
*s	*, *s

Figure 6: Suffix Pattern Matching Rules

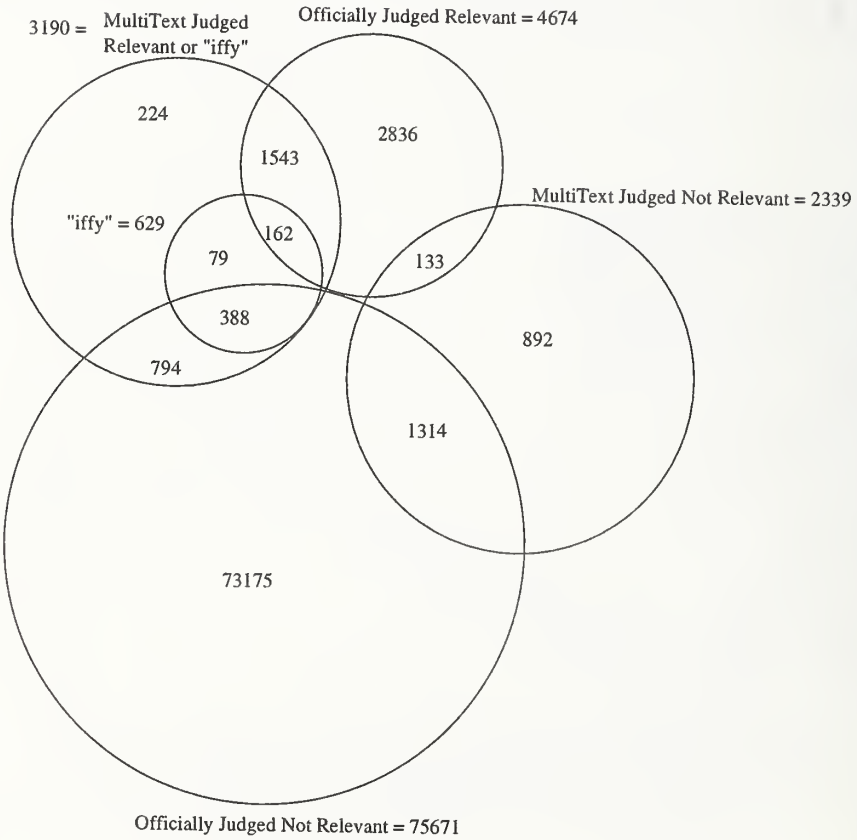


Figure 7: Judging Coverage and Agreement

Run	P@1	P@2	P@4	P@20	P@40	RP	AP
uwmt7a0	.4800	.4500	.4350	.3110	.2280	.2603	.2245
uwmt7a1	.9600	.9100	.8650	.6270	.4635	.5348	.5206
uwmt7a2	.9800	.9600	.8900	.6970	.5480	.6760	.7006
uwmt7isj	1.000	1.000	.9900	.8750	.7097	1.000	1.000
uwmt7a210	.9600	.9600	.8900	.6970	.5490	.6746	.7043
uwmt7isj210	1.000	1.000	.9900	.8750	.7097	1.000	1.000

Figure 8: Adhoc Runs (ISJ Judgements)

For uwmt7a1, we considered combinations of up to five of these elementary queries, choosing the one yielding the best average precision with respect to the ISJ judgements. Then we considered each of the queries in the query log. If one of the original queries (as entered by the searcher) yielded a higher average precision, it replaced the automatically generated query. Such was the case for only 12 topics. The average number of terms per query (counting repeated words only once) was 6.42 terms.

As seen in figures 1 and 5, uwmt7a1 significantly outperforms uwmt7a0 (title only) although the queries are equally simple. uwmt7a2 performs better still. These figures also show that uwmt7a2 and uwmt7isj have nearly identical performance, with uwmt7isj having slightly higher early precision and uwmt7a2 having slightly higher average precision.

6 Combining Results

Some of the runs, most notably uwmt7a2 and uwmt7isj, produced less than 1000 documents for some topics, compromising recall and therefore average precision. To uwmt7a2, we appended the results of uwmt7a1 and uwmt7a0 (up to 1000 documents per topic) to form a new run, uwmt7a210. The performance of uwmt7a210 is, of course, better than uwmt7a2, as shown in figures 1 and 5: average precision, for example, rises from 0.3587 to 0.3868. ISJ (uwmt7isj) can similarly by extended by these other runs, yielding uwmt7isj210. Figures 1 and 5 illustrate that this combination yields the best overall result: average precision improves from 0.3458 to 0.4112.

These combinations show two ways in which extracting queries from ISJ logs can be an advantage: the queries themselves yield good performance, especially when combined; the queries can be used to extend the set of documents found by ISJ. We would expect the latter approach to be particularly effective when the time for interactive search and judging is severely restricted.

7 Very Large Corpus

We used the same queries for the 2 GB adhoc, 1 GB VLC, 10 GB VLC, and 100 GB VLC collections. uwmt7v0 used the same queries as uwmt7a0, fulfilling the requirement for an automatic run based on only title and description fields. uwmt7v1 and uwmt7v2 used the same queries as uwmt7a2 and uwmt7a1 respectively, and may therefore be regarded as routing tasks.

It was observed at TREC-6 that P@20 improves with collection size. This should be no surprise. Indeed, we would expect that P@(k/c) would be constant for a given query, where k is any constant

and c is the size of the collection. We modified the trec_eval program to report relevant P@(k/c) values and observed that this value was roughly constant for our TREC-6 VLC queries, as measured between the 2 GB sample and the 20 GB VLC corpus, and also as measured between the 2 GB adhoc corpus and the 20 GB VLC corpus.

This observation led us to conclude that, when building queries based on ISJ with the 2 GB adhoc corpus, optimizing P@0.4 would yield the best result with respect to the 100 GB VLC2 corpus. On the other hand, optimizing P@4 would be best with respect to the 10 GB VLC corpus, and P@40 would be best with respect to the 1 GB VLC corpus. If we had had sufficient time, we would have constructed such a run in addition to the ones we submitted. Nevertheless, our runs, yielded P@1 (P@0.4 is of course meaningless) of 0.48, 0.96 and 0.98 on our ISJ judgements. Due to judging disagreement, we predicted that the last two might achieve about 2/3 of these values, or 0.64 and 0.65. In fact, our runs achieved P@20 of 0.4420, 0.5740, and 0.5980; about 10 percent less than predicted. As an aside, we note that the P@1 values on NIST judgements (which were, of course, unknown when we prepared these runs) turned out to be 0.5600, 0.7200, and 0.7800.

Our predictions for the smaller sizes were much less accurate. P@40 on the 2 GB adhoc collection should predict P@20 on the 1 GB VLC. P@4 should predict P@20 on the 10 GB VLC. They do not. From this we conclude that the relationship between k and P@(k/c) is radically different between the adhoc and VLC collections. We conjecture that this might be due to a much smaller fraction of relevant documents in VLC. A more appropriate formula might be P@(k/R) where R is the number of relevant documents in the collection. Unfortunately, unlike c, R is very difficult to anticipate for an unknown collection.

The relationship between k and P@(k/c) is invariant among the various VLC collections. For example consider `uwmt7a0`: P@2 (1 GB) and P@20 (10 GB) are 0.3200 and 0.3200; P@2 (10 GB) and P@20 (100 GB) are 0.4444 and 0.4420. The insensitivity of P@(k/c) to c can be seen visually in figure 9.

8 Architecture and Efficiency

For VLC, we used four commodity Intel Pentium II 300 MHz personal computers, connected by a 10 MB/s ethernet. Each computer ran two copies of the search engine, so as to afford CPU/IO overlap during processing. A dispatcher sent each query in turn to all 8 search engines, and waited to receive 20 results from each. Then the best 20 of these results were returned as the result of the query.

The central data structure for each engine is an inverted index, structured so that the entire occurrence list for a word appears contiguously on disk. Furthermore, the index is structured so that a random access into the occurrence list can be performed with a single disk operation [4]. This allows intervals containing a set of terms to be calculated with effort roughly proportional to the frequency of the least common term. We further reduced the effort by restricting the interval size to 128 words or less — this restriction allowed fruitless partial results to be discarded. To further improve performance, the queries were normalized and redundancies were eliminated. Execution times for the VLC runs (seconds/query) are given in figure 10.

The index structures were built in two passes. In the first pass, blocks of approximately 100MB were scanned and indexed, and the index written to disk. In the the second pass, these index blocks

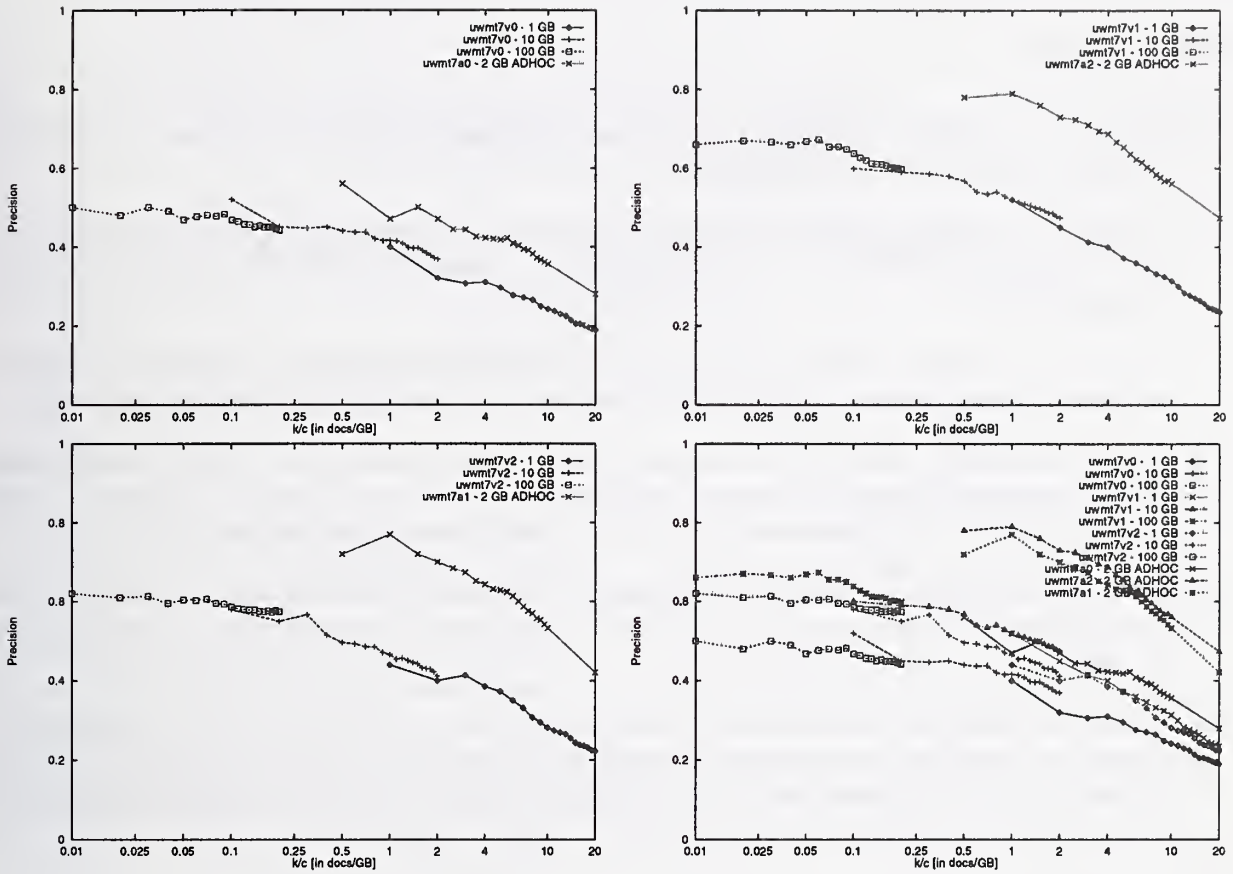


Figure 9: P@(k/c) for each VLC and Adhoc Run

Run	1 GB	10 GB	100 GB
uwmt7v0	.306	.294	.708
uwmt7v2	.251	.299	.882
uwmt7v1	.216	.377	1.51

Figure 10: VLC Query Execution Times

were merged to form a single index. This process was done sequentially for the two engines per computer, and in parallel across the four computers. Total build time for the 1 GB, 10 GB, and 100 GB systems was 0.052 hours (187 seconds), 0.504 hours (30.2 minutes), and 5.33 hours.

References

[1] Charles L. A. Clarke and Gordon V. Cormack. Interactive substring retriveal. In *Fifth Text REtrieval Conference (TREC-5)*, 1996.

[2] Charles L. A. Clarke, Gordon V. Cormack, and Forbes J. Burkowski. Shortest substring ranking. In *Fourth Text REtrieval Conference (TREC-4)*, pages 295–304, Gaithersburg, Maryland, November 1995.

[3] Charles L. A. Clarke, Gordon V. Cormack, and Elizabeth A. Tudhope. Relevance ranking for one to three term queries. In *Fifth RIAO Conference*, pages 388–400, Montreal, June 1997.

[4] Clarlie L. A. Clarke and G. V. Cormack. Dynamic inverted indexes for a distributed full-text retrieval system. Technical Report MT-95-01, MultiText Project, University of Waterloo, 1994. Available at `http://plg.uwaterloo.ca:80/~ftp/mt/TechReports/MT-95-01/`.

[5] Gordon V. Cormack, Charles L. A. Clarke, Christopher R. Palmer, and Samuel S. L. To. Passage-based information retrieval in TREC-6. In *Sixth Text REtrieval Conference (TREC-6)*, 1997.

[6] Gordon V. Cormack, Christopher R. Palmer, and Charles L. A. Clarke. Efficient construction of large test collections. In *Proceedings of SIGIR 98, the ACM Annual Conference on Research and Development in Information Retrieval*, pages 282–289, Melbourne, Australia, August 1998.

BBN at TREC7: Using Hidden Markov Models for Information Retrieval

David R. H. Miller, Tim Leek, Richard M. Schwartz
BBN Technologies
Cambridge, MA USA
{tleek,dmiller,schwartz}@bbn.com

Abstract

We present a new method for information retrieval using hidden Markov models (HMMs) and relate our experience with this system on the TREC-7 ad hoc task. We develop a general framework for incorporating multiple word generation mechanisms within the same model. We then demonstrate that an extremely simple realization of this model substantially outperforms *tf.idf* ranking on both the TREC-6 and TREC-7 ad hoc retrieval tasks. We go on to present several algorithmic refinements, including a novel method for performing blind feedback in the HMM framework. Together, these methods form a state-of-the-art retrieval system that ranked among the best on the TREC-7 ad hoc retrieval task, and showed extraordinary performance in development experiments on TREC-6.

1 Introduction

Hidden Markov models have been applied successfully over the last two decades in a wide variety of speech and language related recognition problems including speech recognition [8], named entity finding [2], optical character recognition [9], and topic identification [18]. For TREC-7, we applied this technology for the first time to the problem of ad hoc information retrieval. On the TREC-7 ad hoc task, our entry ranked among the top tier of systems in average non-interpolated precison [22]. Moreover, our strong development results on TREC-6 hold out the promise of even higher performance from a mature HMM retrieval system.

In all HMM applications, the observed data (*e.g.* audio recording, image bitmap) is modeled as being the output produced by passing some unknown key (*e.g.* words, letters) through a noisy channel. In the ad hoc retrieval problem, we take the observed data to be the query Q, and the unknown key to be a desired relevant document D. The noisy channel is the mind of a user, who is imagined to have some fuzzy notion of which documents he wants, and who transforms that notion into the text of the query Q. Thus, we compute for each document the probability that D was the relevant document in the user's mind, given that Q was the query produced, *i.e.* $P(D \text{ is } R|Q)$, and rank the documents based on this measure.

Using probability models for information retrieval has a history almost four decades long, beginning with the work of Maron and Kuhns [10], and first seeing real application in the "standard probability model" pioneered by Robertson and Sparck-Jones [14]. More recently, however, the introduction of ad hoc constants and non-linear smoothing functions have improved performance steadily at the cost of straying further and further from the probabilistic framework. What started as a reasonable probability model is now masked by numerous heuristics. We believe our new hidden Markov model is more closely tied to its formal probabilistic underpinnings, making it easier to extend and reason about. In addition, the HMM's performance is on a par with the best

automatic query systems.

The remainder of this paper is organized as follows: Section 2 lays out the basic theory of the hidden Markov model system and develops the formulas for a simple realization of it; Section 3 presents experimental results for the basic system on the TREC-6 and TREC-7 ad hoc tasks, and compares the system with the familiar *tf.idf* ranking; Section 4 develops several refinements of the basic HMM system, including a novel method of blind feedback and a more complex HMM which models the production of two-word phrases; Section 5 briefly explores the difference in performance between the TREC-6 and TREC-7 tests; lastly, Section 6 offers some conclusions regarding the system.

2 Probability Model

Given a user-generated query and a set of documents, we wish to rank the documents according to the probability that D is relevant, conditioned on the fact that the user produced Q, *i.e.* $P(D \text{ is } R|Q)$. Applying Bayes' rule, we decompose this into quantities that may be more easily estimated:

$$P(D \text{ is } R|Q) = \frac{P(Q|D \text{ is } R) \cdot P(D \text{ is } R)}{P(Q)} \qquad (1)$$

where $P(Q|D \text{ is } R)$ is the probability of the query being posed, under the hypothesis that the document is relevant; $P(D \text{ is } R)$ is the prior probability that document D is relevant; and $P(Q)$ is the prior probability of query Q being posed.

Since $P(Q)$ will be identical for all documents D, we can safely disregard it for the purposes of sorting documents. We will return in Section 4.4 to the question of estimating the prior probability $P(D \text{ is } R)$, but for now we shall assume that it, too, is constant across all documents. We focus our attention on the remaining term $P(Q|D \text{ is } R)$.

We propose to model the generation of a query by a user as a discrete hidden Markov process dependent on the document the user has in mind.[1] A discrete hidden Markov model is defined by a set of output symbols, a set of states, a matrix of probabilities for transitions between the states, and a probability distribution of output symbols for each state. See [13] for an excellent introduction to hidden Markov models and their application. In the present application, we take the union of all words appearing in the corpus as the set of output symbols, and posit a separate state for each of several mechanisms of query word generation. For example, the states might represent choosing:

- a word from the desired document.
- a word that shares a root with a word appearing in the document.
- a word belonging to a lexicon specific to the topics of the document.
- a word commonly used for information requests.

In fact, the model can be easily extended to accommodate a broad variety of word generation mechanisms.

The process generates the words of the query by traversing a random sequence of states (with probabilities governed by the transition matrix), and at each state producing a word according to the output distribution of the state. Knowing the query that was produced, we can easily compute the probability of its being produced by each of the documents in the corpus. This is the $P(Q|D \text{ is } R)$ term that appears in Equation 1.

In order to use the HMM proposed, we must estimate the transition probabilities and the output distributions for every document in the corpus, since we have a separate HMM for each document. One typically computes estimates of HMM parameters with the EM (Estimation-Maximization) algorithm [5, 3] using training examples, in this case documents paired with queries to which they were relevant. However, such training examples are difficult to come by, and in practice it is usually the case that for the overwhelming majority of documents there are no training queries available. In the face of this difficulty we have proceeded by tying the transition probabilities between states across all models (*i.e.* making the transition matrix document independent), and by

1. In reality, a user rarely has only a single document in mind. However, we assign a probability to each hypothesis "the user has D in mind", and rank the documents using that probability.

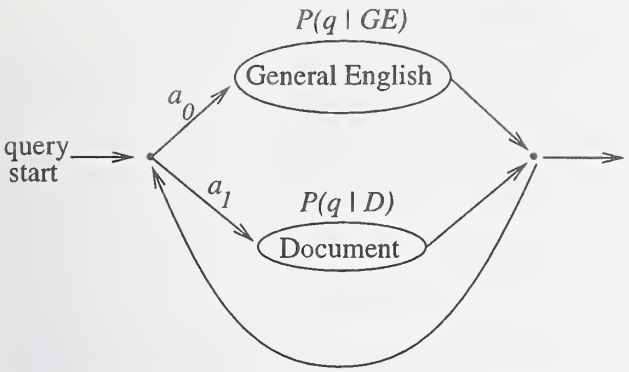


Figure 1: A simple two-state HMM.

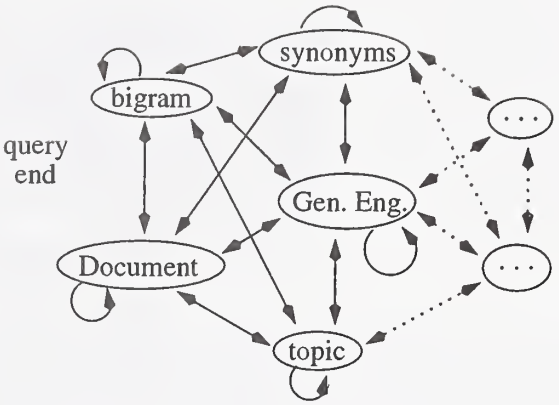


Figure 2: An expanded multi-state HMM.

abandoning EM entirely in favor of simple unigram estimation for the output distributions.

The number and purpose of the HMM states is left open to the practitioner, as is the topology of the transition graph connecting them. A very simple yet remarkably powerful configuration is the two-state, ergodic HMM shown in Figure 1. The first state represents choosing a word directly from the document; the second state represents choosing a word from "general query English." The transition probabilities are constrained so that $P(s_i \rightarrow s_j)$ is the same for all states s_i, which allows us to introduce two null states (states producing no output) and simplify the model. The values of the transition probabilities are estimated by EM. Since $a_0 + a_1 = 1$, there is only a single free parameter to be estimated, for which there is usually ample training data.

The output distribution for the document state is set to be a unigram on words appearing in the document. Ideally, we would like the distribution of the second state to be the unigram on words appearing in all queries. However, since we do not have sufficient training queries to estimate this distribution well, we use the unigram of the entire document corpus as an approximation to this ideal.

The two-state HMM shown in Figure 1 entails an independence assumption which we believe to be a reasonable approximation to the truth. In order to capture context effects better, we later incorporate a third state which models the production of bigrams. Indeed, the HMM framework easily generalizes to include additional states for synonyms, topic specific lexicons, etc., (see Figure 2). Sticking with the two-state model for now, the formula for $P(Q|D \text{ is } R)$ corresponding to Figure 1 is

$$P(Q|D_k \text{ is } R) = \prod_{q \in Q} (a_0 P(q|GE) + a_1 P(q|D_k)). \quad (2)$$

While this formula bears some resemblance to ones used by Ponte/Croft [11] and by Hiemstra/Kraiij [7], it involves a different smoothing term and is arrived at through a different theoretical derivation. Moreover, when extended along the theoretic lines suggested by the HMM (in Section 4) it diverges from these other works considerably.

3 Baseline Performance

In this section we report on ad hoc retrieval experiments we performed on the TREC-6 and TREC-7 test collections using the simple two-state system described in Section 2. We indexed each corpus separately to create inverted index files recording the number of times each word appears in each document. For this indexing, we ignored case and used Porter's algorithm [12] to conflate words with the same stem. We used a list of 397 "stop" words, and replaced all occurrences of these words with the special token *STOP*. In addition, we replaced cer-

$$
\begin{aligned}
tf.idf(Q,D) &= \sum_{q_i \in Q} wtf(q_i,D) \cdot idf(q_i) \\
wtf(q,D) &= \frac{tf(q,D)}{tf(q,D) + 0.5 + 1.5\frac{l(D)}{al}} \\
idf(q) &= \frac{\log \frac{N}{n_q}}{N+1} \\
N &= \text{number of documets in the corpus} \\
n_q &= \text{number of documents in the corpus containing } q \\
tf(q,D) &= \text{number of times } q \text{ appears in } D \\
l(D) &= \text{length of } D \text{ in words} \\
al &= \text{average length in words of a document in the corpus}
\end{aligned}
$$

Figure 3: Comparison *tf.idf* formula.

tain 4-digit strings by the token **YEAR**, suspected dollar amounts by **DOLLAR**, and remaining digit strings by **NUMBER**. We applied the same preprocessing to the queries, and then excluded the stop words from further computation.

Keeping these indices fixed, we ranked documents for each query using the HMM measure of Equation 2, and compared this ranking with that given by the well known *tf.idf* measure. In particular, we used the *tf.idf* measure presented in [15] and reproduced in Figure 3. For the HMM transition probabilities, we used the EM algorithm to train the value of $a_1 = 0.3$ using training examples from the TREC-4 collection.

Table 1 shows the non-interpolated average precision (AveP) achieved by each ranking measure for a variety of test conditions. In all cases, the HMM system dramatically outperforms *tf.idf*, exceeding it by as much as 8 percentage points in absolute terms. Others [23] have reported somewhat better performance from this same *tf.idf* formula (though still not nearly as high as the HMM's performance[2]), which we attribute to differences in indexing which would degrade our results equally for both ranking formulas (e.g. exclusion of different SGML sections, different stop words, different stemming). Since we used the same index for both systems in Table 1, we feel this is a valid comparison.

It is puzzling that the score for the HMM on the full query decreases considerably (3.2%) from TREC-6 to TREC-7, whereas for *tf.idf* it increases slightly. See Section 5 for a discussion of this point.

4 HMM Refinements

Most IR systems do more than just compare the query words with the documents. This section describes four refinements we have added to our system: blind feedback, bigram modeling, feature dependent priors, and query section weighting. We describe and present experimental results for each method separately, and then present the results from using all the methods together.

4.1 Blind Feedback

Blind feedback is a well known technique for enhancing the performance of a retrieval system by conducting a preliminary search with the user's query, automatically constructing a new query based on the top-ranked documents from that initial search, and then conducting a second search with this new query before presenting anything to the user. The Rocchio algorithm [16] is perhaps the best known implementation of this idea, although there are many others as well [17, 4, 24]. We have developed a novel algorithm

2. Dr. J. Xu reported 23.2 AveP on the TREC-6 full queries and 22.6 on the TREC-7 full queries in [23] using the formulas in Figure 3 and the UMass INQUERY indexing.

	TREC-6			TREC-7		
	HMM	*tf.idf*	Diff	HMM	*tf.idf*	Diff
Title	21.6	15.9	+5.8	16.1	11.6	+4.5
Desc	18.1	11.9	+6.2	18.3	14.2	+4.1
Narr	21.5	15.8	+5.7	17.7	14.7	+3.0
Full	27.1	18.9	+8.2	23.9	19.0	+4.9

Table 1: HMM ranking vs. *tf.idf* on TREC-6 and TREC-7.

for blind feedback that is particularly suited for use with hidden Markov models.

Our approach is to augment the initial query with words appearing in two or more of the top N documents, and to adjust the HMM transition probabilities for each word to account for how unexpected those appearances are. For example, seeing the word "very" in 90% of the top N retrieved document carries little information, while seeing "Nixon" in 90% of those same documents is highly informative. We develop a method below that captures this distinction in a principled fashion.

For the two-state HMM, the transition probabilities between the two states can be estimated by the EM algorithm using training queries. For each observation, the EM algorithm distributes the count for that observation to the two-states in proportion to the likelihood of each state's generating that word. Since the document state typically contains only hundreds of words, while the general English state contains hundreds of thousands, whenever the document state has a non-zero probability for a word it usually dwarfs the probability from the general English state. As a result, the estimate from EM for transition into the document state is very close to that obtained by calculating the probability that a query word is in a document, given that the document is relevant. This is[3]

$$P(q' \in D' | D' \text{ is rel. to } Q') = \frac{1}{|\mathcal{Q}|} \sum_{Q' \in \mathcal{Q}} \sum_{q' \in Q'} \frac{|D' \text{ s.t. } q' \in D', D' \text{ is rel. to } Q'|}{|Q'| \cdot |D' \text{ is rel. to } Q'|}. \quad (3)$$

where $mathcalQ$ is the set of available training queries.

Using this insight as our motivation, we consider the case where we have additional query terms taken from the top N ranked documents from a preliminary search for query Q. Given these top N documents, we partition the complete corpus lexicon into $N+1$ disjoint sets of words that we call m-intersections:

$$I_{m,Q} = \left\{ \begin{array}{c} w \text{ appearing in exactly } m \text{ of the} \\ \text{top } N \text{ documents for query } Q \end{array} \right\} \quad (4)$$

for $m = 0, 1, 2, \ldots, N$. For those words $q \in I_{m,Q}$, it is tempting to set the transition probability into the document state to be

$$P(q' \in D' | D' \text{ is rel. to } Q', q' \in I_{m,Q'}). \quad (5)$$

But merely being in a high-order m-intersection is not enough to be an important term. The most common words in the corpus like "the", "a", and "is" would turn up in $I_{N,Q}$ nearly all the time merely as a result of their document frequency, and these carry no information about the query Q.[4]

To compensate for this phenomenon, we condition on and subtract out the baseline document frequency of the words. We define $df(w)$ to be the percentage of documents in the corpus containing word w. We then define

$$\gamma_{m,Q',x} = P\left(q' \in D' \;\middle|\; \begin{array}{l} D' \text{ is rel. to } Q', \\ q' \in I_{m,Q'}, df(q') = x \end{array} \right) \quad (6)$$

3. Here and in discussions below, we use a prime (') to indicate a variable that refers to a generic or training object, while an unmodified variable refers to a specific or test object. Thus Q' is some abstract query while Q is the current query posed to the system.

4. The words "the", "a" and "is" are in our stop list, of course, but the same argument applies for any word with high document frequency that is not in the stop list.

	TREC-6	TREC-7
basic HMM	27.1	23.9
w/blind feedback	30.6	27.4
improvement	+3.5	+3.5

Table 2: Performance gain from blind feedback.

and set the transition probability for query terms in $I_{m,Q}$ with a particular document frequency $df(q)$ to be

$$a_0 = \gamma_{m,Q,df(q)} - df(q). \tag{7}$$

In order to estimate this parameter, we take many training queries and run a preliminary search with each of them to obtain the top N ranked documents. We then count the number of documents each query term appears in. Since these are training queries, we know the complete set of documents that are relevant to each query. With this information we can estimate $\gamma_{m,Q',x}$ by the formula

$$\frac{1}{|Q|}\sum_{Q'\in Q}\sum_{q'\in Q'}\frac{\left|\begin{array}{l} D' \text{ s.t. } q'\in D', D' \text{ is rel. to } Q', \\ q'\in I_{m,Q'}, df(q')=x \end{array}\right|}{|Q'|\cdot\left|\begin{array}{l} D' \text{ s.t. } D' \text{ is rel. to } Q', \\ q'\in I_{m,Q'}, df(q')=x \end{array}\right|}. \tag{8}$$

Blind feedback produced a large and robust performance improvement. We used the top 6 documents from the first retrieval to form m-intersections. We discarded the terms in I_0 and I_1 unless they appeared in the original query as well. We trained the transition probabilities using the 50 queries of the TREC-6 collection, and tested on both the TREC-6 and TREC-7 collections. The improvement of 3.5 AveP on the TREC-6 queries (unfair test on training) carried over exactly to the fair test condition of the TREC-7 queries (see Table 2), indicating that we have not overtuned our parameters to the training data.

4.2 Bigrams

Many words have a distinctive meaning when used in the context of another word, or in a larger phrase. For example, a query using the phrase "white house" is much more likely to be satisfied by a document using those two words in sequence than by one that has them separately. Other systems have attempted to model this phenomenon by fusing selected phrases into a new single term (*e.g.* "white_house", "Pope_John_Paul_II") and using it either instead of or in addition to the individual words [1]. This approach, however, requires that all sentences, whether in documents or queries, be segmented into terms (*e.g.* is "white house secretary" transformed into "white_house secretary" or "white house_secretary"?).

We have taken an alternate approach, in which the words of a query are modeled as always being generated one at a time, but the probabilities governing this generation are conditioned on the identity of the previous word generated. This is accomplished by adding to our HMM a third, document-dependent bigram state (see Figure 4). The output distribution of this state[5] is given by

$$P(q_n|D,q_{n-1}) = \frac{|q_{n-1}q_n \text{ in } D|}{|q_{n-1} \text{ in } D|} \tag{9}$$

where q_n is the current word of the query and q_{n-1} is the previous word. In the event that a document does not contain the previous word of the query the computation backs off to the two-state model, as the denominator of the bigram state output probability would be zero.

Generating a word via this state corresponds to the user's continuing a two-word phrase that was initiated in the previous word. Since the bigram state output probabilities are typically one to three orders of magnitude greater than those in the unigram states, a document containing a bigram that matches the query gains a big boost in likelihood.

The three-state system has a second free parameter, a_2, in the transition probabilities. We optimized the values for a_1 and a_2 to maximize AveP on the TREC-6 task, arriving at $a_1 = 0.29, a_2 = 0.01$. Table 3 shows the effect of using the bigram-state with these transition values for both the TREC-6 and

5. Strictly speaking, the output distribution of an HMM state cannot be dependent on any of the previous outputs. However the unorthodox HMM presented here is equivalent to a strict HMM having one state per distinct word of the document in place of the the single bigram state shown in Figure 4.

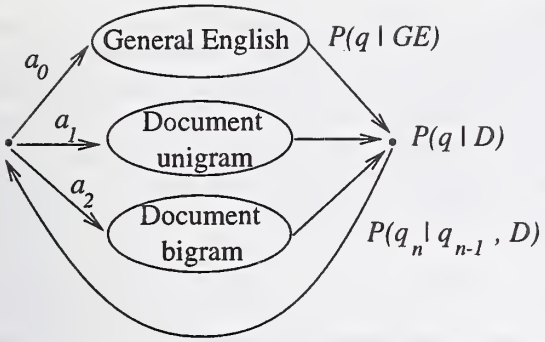


Figure 4: An HMM that models bigram production.

	TREC-6	TREC-7
basic HMM	27.1	23.9
w/bigrams	28.1	24.4
improvement	+1.0	+0.5

Table 3: Performance gains from adding a bigram state.

TREC-7 tasks. The fair gain, while solid, is only half as big as the unfair improvement seen on the TREC-6 task. As there are only two free parameters being tuned, statistical variance between test sets seems a more likely explanation for the discrepancy than over-training.

4.3 Query Section Weighting

Examining the topics from past TREC evaluations, it was clear that the words in the "Title" section were more important than those in the remainder of the topic (although it was unclear whether the "Description" section was more or less useful than the "Narrative" section). In a more general context, a user may wish to designate some portions of his query as more important than others. To exploit this observation, we imagine a simple model in which a user repeats a word multiple times in a query to indicate its greater importance. Under this model, the "Title" declaration is taken simply as shorthand for "repeat these words ν times". Applying these repetitions to

	TREC-6	TREC-7
basic HMM	27.1	23.9
w/query weights	30.0	25.1
improvement	+2.9	+1.2

Table 4: Performance gains query section weighting.

Equation 2 yields

$$P(Q|D_k \text{ is } R) = \prod_{q \in Q} (a_0 P(q|GE) + a_1 P(q|D_k))^{\nu_{s(q)}} \tag{10}$$

where $\nu_{s(q)}$ is the weight (*i.e.* number of repetitions) for the section of the query in which q appears.

We optimized the weights to maximize AveP for the TREC-6 task, which produced values of $\nu_{title} = 5.7$, $\nu_{desc} = 1.2$, $\nu_{narr} = 1.9$. The gain from applying these weights to the TREC-6 task is unfairly optimistic, but Table 4 shows that using these same query section weights improves AveP by 1.2 on TREC-7 in a fair test. In an interactive setting, it would be easy to make term or section weights available to user manipulation.

4.4 Document Priors

In the discussion of Section 2, we made the simplifying assumption that the prior probability of relevance, $P(D \text{ is } R)$, is constant for all documents. However, it is reasonable to think that longer documents may be more useful in general than short ones, or that articles from a refereed journal may be more informative than those from a supermarket tabloid. With this in mind, we searched for features that could predict prior relevance on TREC-6. The most predictive features we found were source, length, and average word-length. Conditioning the document prior on these features and estimating the marginals on TREC-6 yielded a small gain for that corpus, but this gain did not carry over to the fair test set of TREC-7 (see Table 5). Nonetheless, we believe that using a non-constant prior is a good idea and have retained this mechanism in our system.

	TREC-6	TREC-7
basic HMM	27.1	23.9
w/non-constant prior	27.6	24.0
improvement	+0.5	+0.1

Table 5: Performance with non-constant prior.

	TREC-6	TREC-7
basic HMM	27.1	23.9
w/blind feedback	+3.5	+3.5
w/query weights	+2.9	+1.2
w/non-constant prior	+0.5	+0.1
w/bigrams	+1.0	+0.5
HMM w/all refinements	33.2	28.0

Table 6: Performance gains from refinements to the HMM system.

4.5 Additivity of Refinements

Table 6 summarizes the improvements in AveP due to the various extensions described in this section. The first row shows AveP for the basic HMM system, the next four rows show the gain from using any one of the techniques by itself, and the final row shows the result of using all four techniques together. The overall improvement (+6.1 for TREC-6, +4.1 for TREC-7) is roughly 77% of the sum of the individual improvements (+7.9 for TREC-6, +5.3 for TREC-7), indicating that the information captured by these techniques are largely orthogonal to each other.

5 Discussion

Although the HMM system performed well in TREC-7, it did not yield the extraordinary performance we saw in our development work on TREC-6 (our AveP of 33.2 is a full 4.4 points better than the best result reported in the TREC-6 conference [20]). We have tried to understand what accounts for the difference in these results.

Considering first the performance of the basic HMM presented Section 3, we are puzzled by the observation that the score for the HMM on the full query decreases considerably (3.2% absolute) from TREC-6 to TREC-7, whereas for *tf.idf* it increases slightly (see Table 1). In this comparison, the index was held fixed, as was the query pre-processing. Only the ranking function was changed. Since both measures use only exact matches on the stems in the query, problems with stemming, stopping, or document handling should have affected both systems equally.

We decomposed the results on TREC-6 and TREC-7 into query groups having similar numbers of relevant documents (see Table 7). We found that the degradation of the HMM performance across TREC's is entirely localized to those queries that have fewer than 20 relevant documents in the entire corpus (34.5% AveP on TREC-6, 14.6% AveP on TREC-7). Moreover, the standard deviation of per-query AveP on all 50 queries for the HMM (23 on TREC-6, 19 on TREC-7) is considerably higher than the per-query standard deviation for *tf.idf* (15 for TREC-6, 14 for TREC-7). In the end, statistical variance may be the only explanation for the apparent inverse movement in results between the two systems.

Since the overall AveP is an unweighted average across queries (not across documents), a small change in the documents retrieved for the most specific queries has a disproportionate effect on the overall AveP. In search of a measure less sensitive to the location of any single document, we computed the weighted AveP for our systems:

$$\text{wAveP} = \frac{\sum_Q r(Q)\text{AveP}(Q)}{\sum_Q r(Q)}, \qquad (11)$$

where $r(Q)$ is the total number of documents in the corpus judged relevant to Q. As Table 8 shows, the results for both *tf.idf* and the HMM are much more stable using this measure than using the unweighted AveP. This is to be expected, since our training is on query/relevant document pairs, without regard to numbers of documents per query. Perhaps a weighted training method is needed to optimize the model's performance when averaged across queries rather than documents.

Looking next at the extensions and refinements presented in Section 4, we see that the overall improvement was +6.1 for TREC-6, but only +4.1 for TREC-7. Of course, many of the parameters of the

	TREC-6			TREC-7		
#rel docs	#q	HMM	*tf.idf*	#q	HMM	*tf.idf*
0-20	15	34.5	20.4	9	14.6	13.5
21-80	17	25.5	18.4	19	30.5	23.4
> 80	18	22.5	18.2	22	22.3	17.5
All queries	50	27.1	18.9	50	24.0	19.0

Table 7: AveP as a function of total number of relevant documents for a query.

	HMM	*tf.idf*
TREC-6	22.1	18.0
TREC-7	23.2	18.9

Table 8: Weighted AveP for TREC-6 and TREC-7.

algorithms were tuned for the TREC-6 data, making the performance there overly optimistic. To understand better the effect of unfairly tuning to TREC-6, we retuned the entire system to optimize performance on the TREC-7 test. The AveP increased by only 0.7 to 28.7. Put another way, a more realistic estimate of our fair performance on TREC-6 is $33.2 - 0.7 = 32.5$ AveP. This still leaves a gap of 1.3 between the gains from refinements on TREC-7 and the gains from refinements on TREC-7, for which we have no explanation at present.

6 Conclusion

We have presented a novel method for performing information retrieval using hidden Markov models. This framework offers a rich setting in which to incorporate a variety of techniques, both new and familiar. We have experimented with a system that implements blind feedback, bigram modeling, query weighting, and document-feature dependent priors. Our official submission for the ad hoc task of the TREC-7 conference achieved an AveP of 28.0 and was among the top tier of systems [22]. Our own, unofficial test results on the TREC-6 ad hoc task show an AveP substantially higher than any of the official results reported in [20].

We believe that this approach holds great promise beyond its already demonstrated success. The work we have reported represents BBN Technologies' initial foray into the field of information retrieval. The system was conceived, developed, and debugged with only 1.5 people working for eight months. Naturally, there are many familiar ideas that we were unable to incorporate into our system due to time and labor constraints. Among the most glaring examples are an absence of passage retrieval, explicit synonym modeling, and concept modeling. We believe that the HMM approach can be extended to accommodate these and many other ideas under a unified, well-grounded framework. More work still needs to be done.

References

[1] J. Allan, J. P. Callan, W. B. Croft, L. Ballestros, D. Byrd, R. Swan and J. Xu, "INQUERY does battle with TREC-6". In D. K. Harman, editor, Proceedings of the Sixth Text Retrieval Conference (TREC-6), NIST Special Publication 500-240 (1996).

[2] D. Bikel, S. Miller, R. Schwartz, R. Weischedel, "Nymble: a high-performance learning name-finder." Fifth Conference on Applied Natural Language Processing, (published by ACL), pp 194-201 (1997).

[3] W. Byrne, *Encoding and Representing Phonemic Sequences Using Nonlinear Networks*, Ph.D. Dissertation, University of Maryland, College Park, 1993.

[4] W. Cohen and Y. Singer, "Context-sensitive learning methods for text categorization". In *Proceedings of the 19th Annual International ACM*

SIGIR conference on Research and Development in Information Retrieval, pp. 307-315, (1996).

[5] A. Dempster, N. Laird and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society (B)*, Vol. 39, No. 1, pp. 1-22, 1977.

[6] D. Harman, "Overview of the Fourth Text REtrival Conference." In D. K. Harman, editor, Proceedings of the Sixth Text Retrieval Conference (TREC-6), NIST Special Publication 500-236, pp. 1-24 (1996).

[7] D. Hiemstra and W. Kraaij, "TREC-7 working notes: Twenty-One in ad-hoc and CLIR" In D. K. Harman, editor, Proceedings of the Seventh Text Retrieval Conference (TREC-7). Elsewhere in this volume (1999).

[8] J. Makhoul and R. Schwartz, "State of the art in continuous speech recognition", Proc. Natl. Acad. Sci. USA 92, pp 9956-9963 (1995).

[9] J. Makhoul, R. Schwartz, C LaPre, I. Bazzi, "A script-independent methodolgy for optical character recognition." Pattern Recognition, Vol 31, No. 9, pp. 1285-1294 (1998).

[10] M. E. Maron and K. L. Kuhns, "On relevance, probabilistic indexing and information retrieval." Journal of the Associations of Computing Machinery, 7, pp. 216-244 (1960).

[11] J. Ponte and W. B. Croft, "A Language Modeling Approach to Information Retrieval." In *Proceedings of the 21st Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 275-281, (1998).

[12] M. F. Porter, "An Algorithm for Suffix Stripping." Program, 14(3), pp. 130-137 (1980).

[13] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition", Proc. IEEE 77, pp. 257-286 (1989).

[14] S. E. Roberston, and K. Sparck Jones "Relevance weighting of search terms." Journal of the ASIS, 27, pp. 129-146 (1976).

[15] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, "Okapi at TREC-3." In D. K. Harman, editor, Proceedings of the Third Text Retrieval Conference (TREC-3), NIST Special Publication 500-226 (1995).

[16] J. J. Rocchio, "Relevance feedback in information retrieval". In *The SMART Retrieval System–Experiments in Automatic Document Processing*, pp. 313-323, Englewood Cliffs, NJ, 1971. Prentice Hall, Inc.

[17] R. Schapire, Y. Singer, A. Singhal, "Boosting and Rocchio Applied to Text Filtering". In *Proceedings of the 21st Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 215-223, (1998).

[18] R. Schwartz, T. Imai, F. Kubala, L. Nguyen, J. Makhoul, "A maximum likelihood model for topic classification of broadcast news." Proc. Eurospeech '97, Rhodes, Greece, pp. 1455-1458 (1997).

[19] E. Voorhees and D. Harman, "Overview of the Sixth Text REtrival Conference." In D. K. Harman, editor, Proceedings of the Sixth Text Retrieval Conference (TREC-6), NIST Special Publication 500-240, pp. 1-24 (1998).

[20] E. Voorhees and D. Harman, "Appendix A: Adhoc Results." In D. K. Harman, editor, Proceedings of the Sixth Text Retrieval Conference (TREC-6), NIST Special Publication 500-240, Appendix A (1998).

[21] E. Voorhees and D. Harman, "Overview of the Seventh Text REtrival Conference." In D. K. Harman, editor, Proceedings of the Seventh Text Retrieval Conference (TREC-7). Elsewhere in this volume (1999).

[22] E. Voorhees and D. Harman, "Appendix A: Adhoc Results." In D. K. Harman, editor, Proceedings of the Seventh Text Retrieval Conference (TREC-7). Elsewhere in this volume (1999).

[23] J. Xu. Personal communication, October, 1998 (1998).

[24] J. Xu and B. Croft, "Improving the Effectiveness of Information Retrieval with Local Context Analysis", 1998. To appear in ACM Transactions on Information Systems (1999).

Effectiveness of Clustering in Ad-Hoc Retrieval

David A. Evans, Alison Huettner, Xiang Tong, Peter Jansen, Jeffrey Bennett

CLARITECH Corporation
A Justsystem Group Company

Abstract In this paper, we describe the experiment underlying the CLARITECH entries in the TREC-7 Ad Hoc Retrieval Track. Based on past results, we have come to regard accurate, selective relevance feedback as the dominant factor in effective retrieval. We hypothesized that a clustered rather than a ranked presentation of documents would facilitate judgments of document relevance, allowing a user to judge more documents accurately in a given period of time. This in turn should yield better feedback performance and ultimately better retrieval results. We found that users were indeed able to find more relevant documents in the same time period when results were clustered rather than ranked. Retrieval results from the cluster run were better than results from the ranked run, and those from a combined run were better still. The difference between the ranked and combined runs was statistically significant for both recall and average precision.

1 Introduction

The most successful approaches to ad-hoc retrieval in recent TREC evaluations have typically involved a combination of manual query formulation, interaction with a user to determine some number of candidate relevant documents, and "relevance" feedback to the system for use in expanding a query and automatically generating a final set of ranked documents. Based on our results in TREC 6 in particular [1], we have come to regard accurate, selective relevance feedback as the dominant factor in determining a successful outcome. In any practical system, such relevance feedback depends on the ability of a user to review and judge a sample of documents in a relatively short amount of time.

Virtually all TREC systems that have utilized user feedback have presented the user with "relevance ranked" lists of documents to review. But such lists may not represent unbiased samples of potentially relevant documents. Serial-order presentation may not be the most appropriate way to organize results. Making the user read or browse documents in isolation may not contribute to the user's efficiency, in particular, in deciding whether to continue reviewing documents, to stop, or to reformulate the query and try again.

Our ad-hoc retrieval experiments in TREC 7 were designed to assess the effectiveness of clustered groups of documents as an alternative to relevance-ranked lists in assisting the user in making relevance judgments. The fifty queries (351–400) were entered into the CLARIT system and edited by a member of the team; these constituted a fixed set of initial, manually prepared queries in all subsequent steps. Eight subjects were enlisted as "users". Their task was to submit the initial queries to a database consisting of the target corpus (excluding the Federal Register collection), and judge results. Results were presented either as relevance-ranked lists of 300 returned documents (the baseline or "ranked" run) or as clustered groups of the top 150 returned documents (the "cluster" run). Each user was assigned some number of queries; half were processed as baseline and half as cluster runs. Users' judgments (documents marked relevant, non-relevant, or merely viewed) were automatically collected at 10, 15, 20, and 30 minutes. During the first 15 minutes, no user interactions with the system were allowed except for the reading and marking of documents. Between 15 and 30 minutes, users were also allowed to reformulate queries and retrieve potentially new results. All fifty queries were processed in each mode; each user processed a query only once in one or the other mode.

In terms of efficiency alone, we observed a positive effect for cluster representations. At all collection points (10, 15, 20, and 30 minutes) the average number of positive judgments per query is higher for the cluster mode. For example, the average number of marked-relevant documents at each point is 8.7, 11.8, 13.9, and 18.7 for the baseline and 9.1, 12.6, 15.9, and 20.5 for the cluster runs. In terms of overall performance—average precision and recall—our official results further demonstrate the higher performance of the cluster runs.

In the following sections, we report on our experimental design, the results we obtained in the several modes of processing, our overall performance on the TREC-7 task, and the results of several follow-up analyses we conducted.

2 Experiment Design

The CLARIT TREC-7 ad hoc retrieval experiment was designed to measure the effect of document clustering on the speed and quality of user relevance judgments. To conduct our experiment, we needed a group of subjects ("users"), an interactive retrieval system with the ability to present results in relevance-ranked lists or in organized clusters, and a design that would insure, as much as possible, that the essential variables in performance would be due to user judgments of documents.

For subjects, we enlisted eight members of the CLARITECH staff. We chose only native speakers of English and tried to

avoid people who had participated in past interactive-retrieval experiments using the CLARIT system. (In fact, only one of the seven subjects had had previous experience using the system.) Among the users were three of the authors of this paper, two other CLARIT developers, and three non-technical volunteers.

For an interactive retrieval system, we chose a version of the CLARIT system that supports both conventional presentation of ranked retrieval results and also automatically clustered results. Since the system has many parameters, we selected a default set and held them constant across all subsequent experiments.

All fifty ad-hoc queries were prepared in advance by two members of the CLARITECH research staff. This was designed to insure that all users would begin their interactions with the same initial queries and that no variability in results would be due to the relative skill (or lack of skill) that individual subjects might have in formulating queries. (We should note that, in the CLARIT system, initial query formulation is typically based on a natural-language statement of the topic and the optional addition of one or more global "constraints" on individual terms. In practice, query formulation is a quick and easy step.)

The 50 ad-hoc topics were randomly[1] divided into six sets of 7 topics and two sets of 4; each user was assigned two sets of topics. (Two users participated only half-time, using the smaller sets.) For one topic set, the user viewed query results in a simple ranked list; for the other set, the top 150 documents retrieved were clustered using CLARIT clustering techniques, and the user was presented with the clusters. Half of the users worked on ranked documents first, while the other half worked on clusters first. Each query was addressed once in each mode, by two different users.

For each topic, users began their interactions by being presented with the initial query and the corresponding initial search results, in ranked or clustered format. Users were instructed to identify as many relevant documents per topic as they could find in 30 minutes, and, along the way, to mark any non-relevant documents that could be useful for negative feedback. For the first 15 minutes, interaction was restricted to review of retrieved documents. Users could scan the terms characterizing a cluster (clustering runs only), read the titles of retrieved documents, or view document text or automatically-generated document summaries to assess document relevance. For the second 15 minutes, users were also permitted to modify the initial query or formulate a new query for the topic. They could reweight query terms, add or delete query terms, modify query constraints, or incorporate system-assisted query enhancements. (This general flow of processing is illustrated in Figure 1.) Judgments were saved automatically at 10 minutes, 15 minutes, 20 minutes, and 30 minutes. User-modified queries were saved at 20 minutes and at 30 minutes.

Subsequent processing of results was fully automatic. We used the accrued judgments collected at the 30-minute point for relevance feedback in the final step of processing over the full TREC target corpus. In each case, we used Rocchio scoring to rank and select 250 "positive" terms from the marked-relevant documents and 15 "negative" terms from the marked-non-relevant documents to supplement the version of the query as formulated at the 30-minute point. This final query was used to retrieve and rank 1,000 documents. In the final submission, we automatically re-sorted the retrieved documents to insure that all previously identified relevant documents were ordered first, followed by the remaining ranked results. We prepared three TREC-7 manual ad-hoc submissions, as follows. (1) A combined set (CLARIT98COMB) based on the unique union of relevance judgments from each mode. (If a document was judged as relevant by one user and as non-relevant by another user, we treated it as relevant.) (2) A cluster set (CLARIT98CLUS) based on the results from the cluster mode. (3) A baseline set (CLARIT98RANK) based on the baseline (relevance-ranked list) mode.

3 General User Performance

From the point of view of general performance, users who interacted with the system in cluster mode rendered more "positive" relevance judgments than users who interacted in ranked mode. In particular, as shown in Table 1, users who interacted in ranked mode recorded 936 positive judgments and 1,626 negative judgments for the 50 topics. Users who interacted in cluster mode recorded 1,025 positive and 1,494 negative judgments.

We did a paired T-test on the numbers of documents users marked as relevant in the ranked sessions versus the number they marked relevant in the cluster sessions, for each time point (10 minutes, 15 minutes, 20 minutes, 30 minutes). At the 15-minute point, in particular, the average number of documents marked should be precisely comparable if clustering has no effect. We found that the cluster sessions runs have higher averages at every time point, although the differences are not statistically significant (see Table 2).

We also regressed the baseline (ranked document) sessions, and found a slope of +7.2. The cluster sessions regressed to a line with a slope of +7.9. A higher slope for clustering could mean that clustering enables the user to find relevant documents faster; however, once again the difference was not significant. We plotted the (*cluster – baseline*) difference points and regressed those data; the slope should be 0 if there is no benefit to clustering. The slope is +1.0—positive but not significant.

[1] The topic sets were randomly generated, except for a restriction that four of the sets contain only topics whose initial queries were written by the same CLARIT researcher. This was necessary because the two researchers who wrote the queries both participated as experimental subjects. To avoid bias, each one worked only with queries written by the other.

4 Retrieval Performance

The official TREC results represent one measure of the relative effectiveness of the two modes of interaction that users engaged in. Though users in the cluster mode submitted more documents to the system as candidate relevants, it was possible that their judgments were inaccurate and that the greater number of documents they nominated would lead to a degradation in system performance. As can be seen in the official results as summarized in Table 3, however, that was not the case. The cluster runs outperformed the ranked ones on ali measures, in particular, on both average precision and total recall.

We note also in Table 3 that the overall performance of CLARIT98COMB was superior to that of CLARIT98CLUS on all measures except initial precision. The differences between CLARIT98RANK and CLARIT98COMB are statistically significant at 95% confidence, so we can conclude that clustering has a positive incremental effect. Table 4 provides further information relative to the performance of the three runs. We note again the superior performance, especially in front-end precision, of the CLARIT98CLUS run.

In terms of TREC-group performance, all three submissions performed well above average, as can be seen in Table 5. CLARIT98COMB was below median for only seven topics; and, interestingly, CLARIT98CLUS scored seven "bests".

5 Effects of Relevance Judgments

In our post-TREC experiments, we compared the NIST judges' and CLARIT users' relevance judgments, and evaluated the relative impact of judgment differences on retrieval performance.

Tables 6–8 summarize the differences between NIST and CLARIT relevance judgments for the documents that were judged by both the NIST judges and CLARIT users for the 50 topics. Table 6 shows the CLARIT user judgments from the ranked run, Table 7 from the cluster run, and Table 8 from the "combined" run, in which we merged the judgments from the baseline and cluster runs for the automatic relevance-feedback retrieval step.

The agreement between the two judgments is calculated by dividing the *number with same judgment* by the *total number of judged documents.*

For the ranked run, agreement is (680 + 1076) / (680 + 1076 + 204 + 256), or 0.7924. For the clustered run, agreement is (703 + 984) / (703 + 984 + 177 + 322), or 0.7717. For the combined run, in which we artificially "lowered" the criterion for "yes" by taking an "OR" operation when merging, agreement is (983 + 1691) / (983 + 1691 + 227 + 512), or 0.7835. This level of agreement is comparable to the levels reported by NIST in studies of inter-rater reliability among TREC judges.

We conducted two sets of experiments to test the effect of the difference in relevance judgments on retrieval performance. We compared our official CLARIT ad hoc runs with the results of experiments that use two different document relevance assessments: first, the "corrected" relevance judgments of the CLARIT users and, second, the relevance judgments of the NIST judges.

Experiments RANKCOR, CLUSCOR, and COMBCOR use the "corrected" relevance judgments of the CLARIT users. CLARIT users' relevance judgments were revised to reflect the relevance judgments of the NIST judges wherever there was a conflict; in cases where CLARIT users judged a particular document but NIST judges did not, the CLARIT user judgment was retained. The resulting numbers of relevant and non-relevant judgments used in the batch feedback step are shown in Table 9.

In all other processing, the RANKCOR, CLUSCOR, and COMBCOR runs were identical to CLARIT98RANK, CLARIT98CLUS, and CLARIT98COMB, respectively. Comparison of retrieval performance is given in Table 10.

In general, we can see that "corrected" relevance judgments have a dramatic impact on the performance of the system. The improvement in recall when judgments are "corrected" is statistically significant at 95% confidence for the ranked-document run; the larger improvement in average precision and precision at 100 documents is statistically significant for all three runs. Of course, from the point of view of the CLARIT users, all the documents they marked as relevant were "correct". Thus, depending on one's point of view, this evaluation can be regarded as giving either a practical upper limit on the performance of the system (the results with NIST judgments substituted selectively for CLARIT user judgments) or a measure of the distortion introduced by conflicting user judgments, which cannot be avoided in actual retrieval applications.

The second set of experiments compares the effectiveness of the relevance feedback based on complete NIST relevance judgments on the set of documents judged by CLARIT users. In RANKNIST, CLUSNIST, and COMBNIST, CLARIT users' relevance judgments were revised to reflect the NIST judges' relevance judgments in all cases, including no judgment if the document was not judged by NIST. The resulting numbers of positive and negative judgments are shown in Table 11.

Note that the numbers of positive judgments for the ranked, cluster, and combined runs are exactly the same when CLARIT judgments are "corrected" (Table 9) and when NIST judgments are substituted (Table 11). This reflects the fact that the NIST judges concentrated on documents that one or more TREC contestants judged relevant; there simply are no documents that CLARIT users judged relevant and NIST judges left unjudged. The numbers of negative judgments do differ from the "corrected" runs to the NIST-only runs, however. The results of the NIST-only run are shown in Table 12. The slight differences between RANKCOR, CLUSCOR and COMBCOR in Table 10 and RANKNIST, CLUSNIST, and COMBNIST in Table 12 must be attributed

to the change in the number of non-relevant judgments included in the batch feedback step.

6 Effects of Timing

Since our TREC experiment automatically saved CLARIT user judgments at 10-, 15- , and 20-minute points, as well as at the end of each 30-minute session, we were able to compare numbers of documents judged at timed intervals (Table 2), and to use these sets of judgments in the batch feedback step in additional experiments (Table 13).

The difference between the ranked and cluster runs was statistically significant at 95% confidence for both recall and precision at 20 minutes; it was not significant at 10, 15, or (the official) 30 minutes.

7 Conclusion

We consider our experiments to be a first step in the direction of assessing the effects of information organization on user and system performance. We consider such effects to be critical, especially in a system such as CLARIT which already supports the user in the efficient discovery of relevant information and performs extremely well with a relatively small amount of user feedback.

These first experiments are somewhat inconclusive. We see that clustered representations of retrieved documents lead to overall better performance, both by the user and by the system, but the magnitude of the improved performance is not statistically significant in all cases. The fact that we see such statistically significant improvement at the 20-minute point is especially encouraging, however, since we would hope to see an impact in shorter, not longer, periods of interaction.

The full assessment of the role of information organization in user/system interactions will require a great deal more research. Even in our current design, there are many obvious questions that bear further investigation.

As one example, it would be interesting to know whether the initial queries we used in our experiment were "too good". One of our hypotheses is that proper clustering and results organization will assist users by concentrating related relevant documents (segregated from non-relevant ones) among a large set of retrieved results. Such an effect would tend to be especially dramatic in the case of a poor or limited initial query, since the relevant documents that respond to such a query are likely not to be serially adjacent to one another—or at the top—in a ranked list. Thus, any process that identifies similar documents and groups them might well succeed in isolating the few relevant documents that respond to a poor query, giving the user an opportunity to identify them more easily. We intend to rerun our experiments with new subjects and impoverished queries to test this hypothesis.

In particular, we intend to repeat the experiments with sparser initial queries, not only to determine whether the original initial queries were "too good" and thus dampened the positive effects of clustering that we observed, but also to determine whether there is a "lower bound" on the effect—a minimal query such that no difference in effectiveness can be observed.

As another example, it will be important for us to experiment with some of the many parameters that exist in our system for clustering, to assess their effect on user performance. Can users work more efficiently with larger or with smaller clusters? Should clusters be summarized via terms or discursively? Should documents within clusters be ranked or organized further with respect to the initial query? There are many such issues that we are only beginning to investigate.

References

1. [Milic-Frayling et al. 1998] Milic-Frayling, Natasa, Chengxiang Zhai, Xiang Tong, Peter Jansen, and David A. Evans, "Experiments in Query Optimization, the CLARIT System TREC-6 Report". In Voorhees, E.M., and Harman, D.K. (Editors), *The Sixth Text REtrieval Conference (TREC-6)*. NIST Special Publication 500-240. Washington, DC: U.S. Government Printing Office, 1998, 415–454.

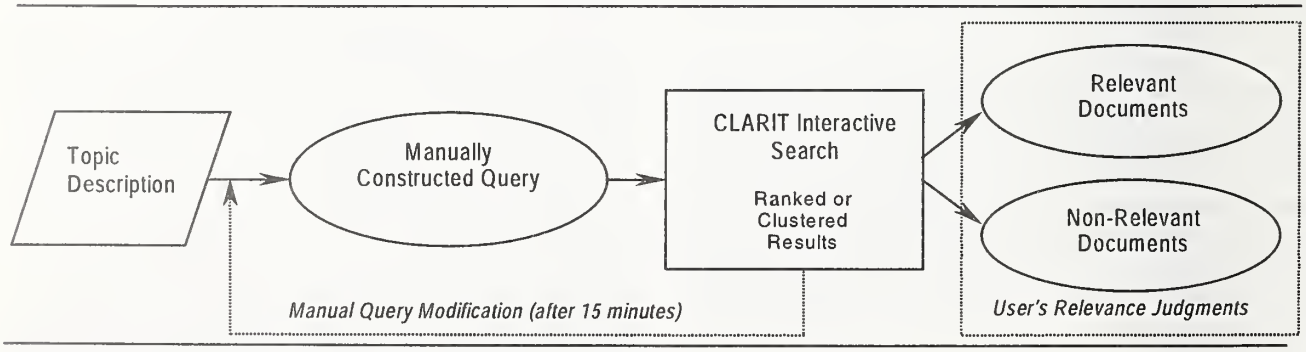


Figure 1. Interactive query formulation and relevance assessment.

Positive judgments	Ranked session	Cluster session
Total	936	1025
Average	18.72	20.50
Maximum number for one topic	59	75
Minimum number for one topic	1	2
Number of topics w/o positive judgments	0	0
Negative judgments	**Ranked session**	**Cluster session**
Total	1626	1494
Average	32.52	29.88
Maximum number for one topic	116	140
Minimum number for one topic	3	1
Number of topics w/o negative judgments	2	6

Table 1. Statistics about CLARIT users relevance judgments.

	10 minutes	15 minutes	20 minutes	30 minutes (basis for TREC runs)
Ranked runs				
Positive judgments	433	591	693	936
Negative judgments	514	910	1187	1626
Cluster runs				
Positive judgments	457	629	797	1025
Negative judgments	514	847	1080	1494

Table 2. Number of positive and negative judgments in each mode at timed intervals.

Run	Recall	Avg. Precision	Initial Precision	Exact Precision	Prec. 100 docs
1. CLARIT98RANK	3198	0.3351	0.8814	0.3726	0.2864
2. CLARIT98CLUS	3310	0.3525	0.9066	0.3730	0.2982
(over (1))	(+ 3.50%)	(+5.19%)	(+2.86%)	(+0.11%)	(+0.63%)
3. CLARIT98COMB	3417	0.3702	0.8796	0.4140	0.3178
(over (1))	(+6.85%)	(+10.47%)	(-2.98%)	(+11.11%)	(+10.96%)
(over (2))	(+3.23%)	(+5.02%)	(-0.20%)	(+10.99%)	(+6.57%)

Table 3. Performance statistics for CLARIT98RANK, CLARIT98CLUS, and CLARIT98COMB.

	CLARIT98RANK	CLARIT98CLUS	CLARIT98COMB
At 5 docs	0.6720	0.7600	0.6920
At 10 docs	0.6440	0.6940	0.6940
At 15 docs	0.6320	0.6360	0.6613
At 20 docs	0.6050	0.5870	0.6180
At 30 docs	0.5327	0.5100	0.5653
At 100 docs	0.2864	0.2982	0.3178
At 200 docs	0.1975	0.1979	0.2142
At 500 docs	0.1074	0.1106	0.1147
At 1000 docs	0.0638	0.0662	0.0683
Exact Precision	0.3726	0.3730	0.4140

Table 4. Precision at N retrieved documents for CLARIT98RANK, CLARIT98CLUS, and CLARIT98COMB.

Run	Average Precision			
	>= median	< median	= best	= worst
CLARIT98RANK	33	17	4	0
CLARIT98CLUS	38	12	7	0
CLARIT98COMB	43	7	4	0

Table 5. CLARIT ad hoc results compared to TREC group performance.

		CLARIT Ranked Run		
		Yes	No	*Total*
NIST	Yes	680	204	884
	No	256	1076	1332
Total		936	1280	2216

Table 6. Comparison of CLARIT user judgments on ranked run with NIST judgments for the same documents.

		CLARIT Cluster Run		
		Yes	No	*Total*
NIST	Yes	703	177	880
	No	322	984	1306
	Total	1025	1161	2186

Table 7. Comparison of CLARIT user judgments on cluster run with NIST judgments for the same documents.

		CLARIT Combined Run		
		Yes	No	*Total*
NIST	Yes	983	227	1210
	No	512	1691	2203
	Total	1495	1918	3413

Table 8. Comparison of merged CLARIT user judgments with NIST judgments for the same documents.

	CLARIT98RANK	RANKCOR	*CLARIT98CLUS*	CLUSCOR	*CLARIT98COMB*	COMBCOR
Relevant	*936*	884	*1025*	880	*1495*	1210
Non-relevant	*1626*	1678	*1494*	1639	*2542*	2827

Table 9. Positive and negative judgments after "correction" of CLARIT judgments (official TREC runs shown for comparison, in italics).

Run	Recall	Average Precision	Initial Precision	Exact Precision	Prec. at 100 docs
CLARIT98RANK	**3198**	**0.3351**	**0.8814**	**0.3726**	**0.2864**
RANKCOR	3238	0.4118	0.9828	0.4192	0.3006
(over above)	(+1.25%)	(+22.90%)	(+11.51%)	(+12.51%)	(+4.95%)
CLARIT98CLUST	**3310**	**0.3525**	**0.9066**	**0.3730**	**0.2982**
CLUSCOR	3316	0.4165	1.0000	0.4193	0.3062
(over above)	(+0.18%)	(+18.17%)	(+10.30%)	(+12.41%)	(+2.68%)
CLARIT98COMB	**3417**	**0.3702**	**0.8796**	**0.4140**	**0.3178**
COMBCOR	3410	0.4579	0.9806	0.4550	0.3254
(over above)	(-0.20%)	(+23.70%)	(+11.48%)	(+9.92%)	(+2.39%)

Table 10. Effects of user feedback based on CLARIT users' judgments and "corrected" relevance judgments.

	CLARIT98RANK	RANKNIST	*CLARIT98CLUS*	CLUSNIST	*CLARIT98COMB*	COMBNIST
Pos. judg.	*936*	884	*1025*	880	*1495*	1210
Neg. judg.	*1626*	1332	*1494*	1306	*2542*	2203

Table 11. Positive and negative judgments using NIST judgments only (official TREC runs shown for comparison, in italics).

Run	Recall	Average Precision	Initial Precision	Exact Precision	Prec. at 100 docs
CLARIT98RANK	3198	0.3351	0.8814	0.3726	0.2864
RANKNIST	3236	0.4114	0.9829	0.4188	0.3006
CLARIT98CLUST	3310	0.3525	0.9066	0.3730	0.2982
CLUSNIST	3312	0.4164	1.0000	0.4196	0.3060
CLARIT98COMB	3417	0.3702	0.8796	0.4140	0.3178
COMBNIST	3406	0.4577	0.9806	0.4552	0.3258

Table 12. Effects of user feedback based on NIST judgments only.

Ranked run	Recall	Average Precision	Initial Precision	Exact Precision	Prec. at 100 docs
10 min	3036	0.2806	0.8635	0.3228	0.2488
15 min	2994	0.3007	0.8729	0.3400	0.2608
20 min	3017	0.2982	0.8734	0.3416	0.2638
30 min	3198	0.3351	0.8814	0.3726	0.2869
Cluster Run					
10 min	3060	0.2847	0.8430	0.3189	0.2468
15 min	3141	0.3099	0.8507	0.3426	0.2766
20 min	3171	0.3324	0.8773	0.3616	0.2848
30 min	3310	0.3525	0.9066	0.3730	0.2982

Table 13. Results using judgments at timed intervals, ranked and cluster runs.

Threshold Calibration in CLARIT Adaptive Filtering

Chengxiang Zhai, Peter Jansen, Emilia Stoica, Norbert Grot, David A. Evans

CLARITECH Corporation
A Justsystem Group Company

Abstract In this paper, we describe the system and methods used for the CLARITECH entries in the TREC-7 Filtering Track. Our main aim was to study algorithms, designs, and parameters for Adaptive Filtering, as this comes closest to actual applications. For efficiency's sake, however, we adapted a system largely geared towards retrieval and introduced a few critical new components. The first of these components, the *delivery ratio* mechanism, is used to obtain a profile threshold when no feedback has been received. A second method, which we call *beta-gamma regulation*, is used for threshold updating. It takes into account the number of judged documents processed by the system as well as an expected bias in optimal threshold calculation. Several parameters were determined empirically: apart from the parameters pertaining to the new components, we also experimented with different choices for the reference corpus, and different "chunk" sizes for processing news stories. Gradually increasing chunk sizes over "time" appears to help profile learning. Finally, we examined the effect of terminating underperforming queries over the AP90 corpus and found that the utility metric over AP88-AP89 was a good predictor. All of the above innovations contributed to the success of the CLARITECH system in the adaptive filtering track.

1 Introduction

Filtering in general, and adaptive filtering in particular, is one of the most challenging problems in information retrieval.

This year's TREC Filtering track was redesigned to accommodate a more realistic evaluation of practical filtering systems. One major difference was the absence of initial training information in the Adaptive Filtering task. Another was the more realistic nature of feedback: judgments were provided only for documents accepted by the system, and restrictions were placed on information available at the time of that decision.

These changes necessitated important modifications to our Filtering Evaluation system from previous years. Our goal was to evaluate the basic CLARIT adaptive filtering approach, which is based on standard CLARIT retrieval and routing techniques. [1,2,3,4] While the CLARIT system can be (and has been) extended to support real-time filtering—processing each incoming document in real time—we could not afford the time to adapt such a real-time filtering system for TREC evaluation. Therefore we used our standard CLARIT retrieval and profile training mechanism to make batch filtering decisions and perform batch updating on a succession of "chunks" of source documents. This improved efficiency but at the cost of some precision and sensitivity, as any feedback information can only be applied from the next chunk on.

Our basic approach to filtering still involves a two-step procedure similar to the one used in many other systems. For each document-profile pair, we compute a relevance score and then apply a score threshold to make the (binary) decision to accept or reject the document. Therefore, in this paradigm the two most important technical procedures to be worked out are scoring and threshold setting.

For our TREC-7 Filtering Track experiments, we decided to focus primarily on the problem of threshold setting, in large measure because (1) we did not understand it as well as the problem of scoring, and (2) it may have the greater impact on perceived performance (utility). The threshold-setting problem can be subdivided into two parts: (a) initial threshold setting, before there are any relevance judgments from the user; and (b) threshold updating, at any point when relevance judgments are fed back to the system. We used different techniques to set an initial threshold and to update the threshold during filtering.

Although we participated in both the adaptive filtering task and the batch filtering task, our focus was on adaptive filtering. We made two submissions for each utility measure for adaptive filtering. The first submission for each utility represented an optimal threshold parameter configuration as determined in our preliminary experiments. The second submission differed from the first only in that we adopted the rather user-unfriendly strategy of refusing any documents from AP90 for those topics that have an accumulated negative utility over AP88 and AP89. A comparison of the two submissions allows us to see how well a negative training utility can identify "difficult" topics.

In the following section, we describe our general procedure for adaptive filtering experiments. In Section 3, we discuss our main new algorithms, for initial threshold setting and for threshold updating. Our parameter space and the parameter settings we found to be best in pre-TREC runs are described in Section 4. Section 5 reviews our results and findings based on the experiments. Batch filtering, though not our main focus, nevertheless led to interesting insights and is discussed in Section 6. Finally, in Section 7, we summarize the main points and our plans for further work.

2 Adaptive Filtering Experimental Procedure

Conceptually, a profile (i.e., a binary document classifier) consists of three elements: a term vector, IDF statistics, and a score threshold. The first two are used to assign a score to any document, and the third is used to make the binary decision whether to accept the document.

The initial profile term vector for each topic was created automatically by parsing the original topic descriptions. We used all the fields (except the definition field, if any) in the topic description. The initial IDF statistics were derived from an unrelated reference corpus (*Wall Street Journal* 1987[1]). The initial profile threshold is set using the delivery ratio method described in the next section.

Source documents (i.e., the AP data) are segmented into a number of chunks, possibly of different sizes. Each chunk is indexed on noun phrases and individual words using the standard CLARIT phrase indexing technique. [1,2,4] Chunks are processed sequentially. At each chunk, we iterate over all the profiles, and run each profile as a query over the current chunk corpus. The matching function is a vector space dot product over terms in the profile space. To avoid using statistics based on "future" information, including the current chunk, we only used IDF statistics based on "earlier" chunks for matching (or WSJ87 data for the first two chunks — see Table 1). All the documents scored above the profile threshold are accepted.

Relevance judgments for the accepted documents are then obtained and the current chunk is used as a training corpus to update each profile independently.

Updating consists of two stages. The first stage involves term vector updating (i.e., expansion). We used the same general procedure that we employed in our TREC-6 Routing experiments. [4] Rocchio feedback, on relevant documents only, is used to expand the current term vector. [5,6] Specifically, the centroid vector of the relevant document vectors is computed and the terms are ranked by their centroid weight.[2] The K best-ranked terms are selected. Unlike standard Rocchio feedback, we assign a uniform weight to the selected terms before merging them into the current term vector. K grows heuristically with the number of relevant documents (N) available for training, according to the function $K = 10 + 10 * \log(N+1)$.

The second stage involves threshold updating (i.e., re-estimating a threshold for the new term vector to be used when processing the next chunk). We use a method we call "beta-gamma regulation" to set a threshold for the new vector based on the current chunk (as an approximation of the next chunk), the future matching IDF statistics (from the "seen" documents up to the current chunk), and the partial relevance judgment on the current chunk. The details of this method are described in the next section.

The new, updated profile is then used to process the next chunk, and the above process is repeated until the last chunk (i.e., AP90) has been handled. Finally, the accepted documents for all the chunks are combined as the results for evaluation.

Note that a drawback of retrieval over chunks is that relevance information cannot be used immediately (according to the needs of each profile independently). If the threshold has been set inappropriately, there is no way for the system to correct this until the end of the chunk, at which time considerable damage may have been done to the performance.

3 Threshold Setting and Threshold Updating

To estimate an *initial profile threshold*, we used a new method, which we call the "delivery ratio" method. The rationale behind this method is that, in the absence of evidence pertaining to document relevance scores and stream topic density, a plausible utility metric may be the number of documents delivered to a user. A threshold can be set to best approximate the desirable number of documents to deliver. For a given time period, a desirable amount of delivery can be projected to a delivery ratio based on an estimate of the stream volume. A small reference corpus can be used to estimate an approximate threshold score at which the desirable ratio would be achieved.

Specifically, assume that the user wants to have a certain fraction (r), say 10% of the news delivered, we can run the profile vector as a query on the reference corpus using the same IDF statistics as would be used for matching future documents. The delivery ratio threshold is set to the score of the K-th document in the ranked list of retrieved documents, where $K = r*N$ and N is the number of documents in the reference corpus. In special cases when $K < 1$ or K is larger than the size of the list of all matched documents, heuristic extrapolation is applied.

For *threshold updating* we used beta-gamma adaptive threshold regulation. This technique selects a threshold, θ, by interpolating between an "optimal" threshold, θ_{op}, and "zero" threshold, θ_{zero},.

The *optimal threshold* is the threshold that yields the highest utility, given the newly updated term vector, over the accumulated training data. The *zero-threshold* is the highest threshold below the optimal threshold that gives a non-positive utility over the training data under the assumption that all documents that were rejected are non-relevant.

There are two reasons for believing that the "optimal" threshold our training procedure derives from the training data serves as an upper bound for the threshold, and is biased towards higher values. First, only an incomplete set of documents has been judged. As all un-judged documents are assumed to be non-relevant, the true optimal threshold, assuming complete knowledge of relevance judgments,

[1] Note that we avoided using any data from the time period covered by the test data, as these data might have had some overlap and would not have been available in a real application.

[2] While normalized TF is often used in the CLARIT system, we used the raw within-document frequency for Rocchio feedback here. Our goal in doing this was to emphasize TF over IDF in the presence of very few relevant training examples.

could only be lower and never higher than the estimated optimal threshold. Second, the scores of the positive training examples tend to be higher than the expected score of any randomly selected relevant document, since the term vector as trained with training examples favors the terms in the training documents. In other words, using the same training data for vector updating and threshold setting may lead to over-fitting. In addition, for learning and experimentation (especially in the beginning), we want to use a threshold somewhat lower than the true optimal threshold, even should we be able to estimate its value accurately.

At the lower end of the range, preliminary experiments indicated that using a zero utility threshold as a lower bound is a safe procedure, even though it is theoretically possible that the actual optimal threshold is lower still.

To obtain an actual threshold to use for a profile, we interpolate between θ_{opt} and θ_{zero}. Our pre-TREC experiments were geared towards finding an appropriate interpolation scheme. We first experimented with simple linear interpolation, using a constant parameter α, and called this method "alpha regulation" where α plays the role indicated in the following formula.

$$\theta = \alpha * \theta_{zero} + (1-\alpha) * \theta_{opt}$$

After several experiments, and some study of the method's behavior, we decided to express α as a function of two further parameters, β and γ, related to the two factors in the threshold bias identified above. We postulated the following formula, in which M is the number of judged training documents.

$$\alpha = \beta + (1-\beta) * e^{-\gamma * M}$$

In writing α in terms of β and γ, we attempt to capture both aspects of the bias present in the optimal threshold calculation: (1) β is a score bias correction factor that compensates for the relatively higher scores of relevant documents in the training corpus, and (2) γ expresses our belief that the estimated optimal threshold approximates the true optimal threshold more closely when more training examples are available. Note that γ is the inverse of the number of documents at which we place the threshold at approximately the midpoint of our range. If fewer than $1/\gamma$ training examples are available, the threshold will be somewhat lower; if more, somewhat higher.

Figure 1 illustrates the idea behind the formulas graphically.

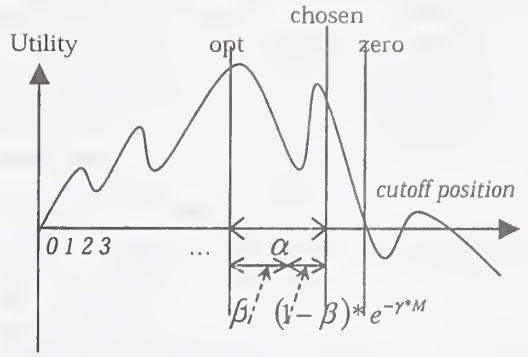


Figure 1. Beta-gamma regulation parameters.

Given a ranked list of all the documents in the training database sorted by their scores, their relevance, and a specific utility criterion, we can plot the utility value at each different cutoff position. Each cutoff position corresponds to a score threshold. Figure 1 shows how a choice of alpha determines a cutoff point between the optimal and the zero utility points, and how β and γ help us to adjust alpha dynamically according to the number of judged examples in the training database.

4 Configuring Experimental Parameters

In this section, we describe the values of several parameters of our system led to the best performance in our preliminary experiments. The parameters with the largest impact were the initial profile threshold, the document chunk sizes, subdocument size, and the β and γ factors used in threshold updating.

Delivery Ratio. The threshold for the initial profiles was set using our *delivery ratio* method, with a ratio of 0.0005, i.e., 1 out of 2,000 documents. A collection of all available 1987 Wall Street Journal articles was used as a reference corpus, approximating somewhat a possible earlier news stream.

Chunk Size. Another main parameter (though a direct consequence of our approximate method to simulate real-time filtering) was the *size of successive chunks* of news articles (roughly corresponding to periods of time over which news is accumulated). Using smaller chunks tends to be more "robust" as this limits the damage from a bad threshold in the overall utility. It also provides more flexibility in the presence of changes. But as smaller chunks contain fewer examples, they may provide less reliable profile learning and be overly sensitive to random fluctuations (we only used the examples in the previous chunk for updating).

In the first stages of learning, both the term vector and threshold are less reliable, and smaller-sized chunks are preferable. In addition, it is sometimes useful to lower the threshold to boost the number of judged examples presented to the system to speed up initial learning. This introduces the risk that many non-relevant documents might be accepted. In later stages, the profile can be assumed to be more stable, and the threshold more reliable. Hence larger chunks are to be preferred for better training.

In fact, our preliminary experiments with *Wall Street Journal* data bore out these hypotheses: chunks of increasing sizes generally led to better performance than chunks of equal sizes. Our post-TREC experiments confirm that using increasing-size chunks helps learning on AP as well.

In practice, we segmented AP88 and AP89 into 15 chunks with increasing sizes starting at 3,000 articles and going up to over 20,000 articles. Our hope was that both the term vector and the threshold would become stable enough to handle the AP90 collection as one chunk.

To simulate the accumulation of information about the news stream over time, we pre-built the reference corpus for each chunk (used for matching IDF statistics) so as to provide a compromise between availability, recency, size, and convenience. This resulted in the following arrangement.

Current Chunk	Reference corpus (IDF) used
Chunk 1 (3,000)	WSJ87
Chunk 2 (3,000)	WSJ87
Chunk 3 (4,000)	Chunk 1 + Chunk 2
...	...
Chunk 8 (9,000)	Chunk 1 + ... + Chunk 7
Chunk 9 (10,000)	Chunk 1 + ... + Chunk 8
Chunk 10 (12,000)	Chunk 1 + ... + Chunk 9
Chunk 11 (14,000)	Chunk 1 + ... + Chunk 10
Chunk 12 (17,000)	Chunk 1 + ... + Chunk 11
Chunk 13 (20,000)	Chunk 1 + ... + Chunk 11
Chunk 14 (24,000)	Chunk 1 + ... + Chunk 11
Chunk 15 (22,597)	Chunk 1 + ... + Chunk 11
Chunk 16 = AP90	AP89

Table 1. Size of source and reference chunks

Subdocument Size. Another parameter we varied in our experiments is the *subdocument size* used for indexing. Although intuitively subdocument indexing is appealing, preliminary experiments indicated that indexing on whole documents performed better, though only slightly.[3]

Threshold Regulation. The beta-gamma threshold regulation method was used to set a new threshold using the formula described in Section 3. In our preliminary experiments with *Wall Street Journal* data, we explored a large space of beta and gamma values and found that the best performance was fairly consistently reached, for both F1 and F3, at a setting of $\beta = 0.1$ and $\gamma = 0.05$. We used this setting in all our official runs.

5 Analysis of Adaptive Filtering Results

As a general trend, participating systems did relatively poorly for AP88, much better for AP89, and again somewhat worse for AP90. This effect can generally be attributed to an inherent instability in experimenting over part of the AP88 corpus (to train the profile and set the threshold), attained stability in performance for the rest of AP88 and AP89, and perhaps deteriorating stability (and the use of more defensive strategies) for AP90. This would indicate that most systems did indeed learn. For F1 (and a fortiori for F3) it turned out to be possible to obtain an overall positive average utility.

CLARITECH submitted four runs for adaptive filtering, two per utility. Except for F1 for AP88, our runs ended up at the top of participating systems. In this section we try to identify the factors that contributed to this result.

Apart from an indication that our system works satisfactorily, that Rocchio is a dependable term selection method, and that we did not make major errors, we can offer the following observations, the first three of which we discuss further in separate subsections:

1. Eliminating "bad" topics from consideration for AP90 yields a significant benefit for F1, and does not harm F3.
2. The more complicated beta-gamma regulation algorithm is better than a simple alpha regulation.
3. The use of increasing chunk sizes helps learning.
4. The delivery ratio method, with a conservative initial parameter setting, is a good method for initial threshold setting in the absence of training data.

5.1 Topic Elimination

For certain topics, for example those with only a few relevant documents in the news corpus, it is not possible to obtain a good profile (i.e., a profile for which the precision is > 0.4 for F1, or > 0.2 for F3).[4] For such topics, the highest utility (viz., 0) is achieved by not accepting any documents at all.

The simplest criterion we could think of was whether the total training utility over AP88 and AP89 was positive or not, and though this gives somewhat conservative results, we were not able to find better predictors.

To assess the benefit of this technique, we submitted two runs that differed only in whether they eliminated topics for AP90 or not. For F1, this approach was invasive, but very beneficial: half of the topics (25) were eliminated from consideration, resulting in an approximately 40% reduction in accepted documents and a 266 point increase in the total utility score for all 50 topics (302 vs. 36, i.e., an eight-fold increase in average utility!). Of the 25 eliminated topics, 16 would indeed have accumulated a negative utility (average value of -19.5) over AP90, whereas the remaining 9 would have contributed positive utilities (average value of 5.11). In comparison with the other groups, the median was tied or exceeded 10 more times, and the (zero) maximum 13 more times.

For F3, the impact was less substantial. Sixteen topics were eliminated, resulting in a reduction in the number of accepted documents by approximately 10%, and a total utility increase by 15 points, i.e., by less than 1%. Only 8 of the 16 topics showed a benefit (11 points on average), whereas 7 topics would have accumulated an average positive utility of 10.4. Here both the median and the maximum were tied or exceeded for four more topics.

[3] In practice, whole document indexing is achieved by using a very large subdocument size as a parameter for the CLARIT indexing procedure.

[4] This can be because of many noisy non-relevant documents or ineffective training.

5.2 Beta-Gamma Threshold Regulation

Before attempting a finer control over the threshold adjustments, we used simple linear interpolation with a constant coefficient α (alpha regulation). In this section we compare the average utility of this method with the results from use of the beta-gamma algorithm.

A comparison of average total utility per topic over AP88 and AP89 for our best runs using alpha regulation and our best runs with beta-gamma regulation is given in Table 2.

Average utility over AP88-89	F1	F3
Alpha (best)	6.98	54.96
Beta-Gamma	10.46	72.34
Increase	3.48 (50%)	17.38 (32%)

Table 2. Comparison of alpha, and beta-gamma regulation

Another interesting observation was that the best setting for beta-gamma also was less sensitive to small changes than the best setting for alpha. Furthermore, we found the maximum for beta-gamma (for the settings of our submissions, viz, β = 0.1 and γ = 0.05) to be stable even across databases (*Wall Street Journal* data as well as AP data).

Figure 2, below, demonstrates another aspect of the superiority of beta-gamma regulation. Shown in the graphs are the average utilities for equal-size chunks for our best runs with the respective methods. From these graphs, we learn that the difference between the two methods is most pronounced for "bad" chunks. In other words, beta-gamma regulation appears to be more stable than alpha regulation for both F1 and F3.

5.3 Learning Factors and the Effect of Chunk Size

A difficult but not unimportant question is how to evaluate the extent to which the system has improved the topic profiles over time as a result of learning. Two aspects of learning can be considered: improvement in scoring and improvement in threshold setting.

One way to assess the learning effect is to compare the actual utility scores obtained for each chunk with the maximum possible utility given the current term vector. The maximum possible utility is a measure of the quality of scoring, which is related to term vector training. How well the actual utility approximates the optimal utility, on the other hand, indicates the quality of threshold setting.

The actual and maximum average utilities for each chunk are shown, together with their ratios, in the graphs given in Figures 3a and 3b. In each case the comparison is shown for both F1 and F3. In Figure 3a, equal-sized chunks were used, and Figure 3b corresponds to our official run with increasing-size chunks. In the latter case, utility values are normalized with respect to the size of each chunk.

We see that in all cases there seems to be a *gradual small decrease* (at least not an increase) in optimal utility value. Although this may indicate that we obtain little benefit from profile training, it may also mean, for example, that the relevant documents are not evenly distributed over the stream, or that confusing non-relevant documents start appearing later on.

On the other hand, whereas actual and optimal utility values remain far apart (ratio relatively constant or even decreasing) for equal-sized chunks, the ratio clearly increases for increasing-sized chunks, indicating a gradual improvement in threshold setting.

Though this confirms our intuition that it is better to use chunks of increasing size, we need to point out that several factors play confusing roles. It is important, for example, which reference corpus was being used for the IDF calculations. The decrease in scoring quality from chunk 2 to chunk 3 in our official run, for example, may in part be explained by the switch from a large reference corpus (WSJ87) to a reference corpus only 8% its size (first two chunks of AP88). Also, certain chunks contain strongly varying numbers of relevant documents for some topics, leading to increased variance of the average utility.

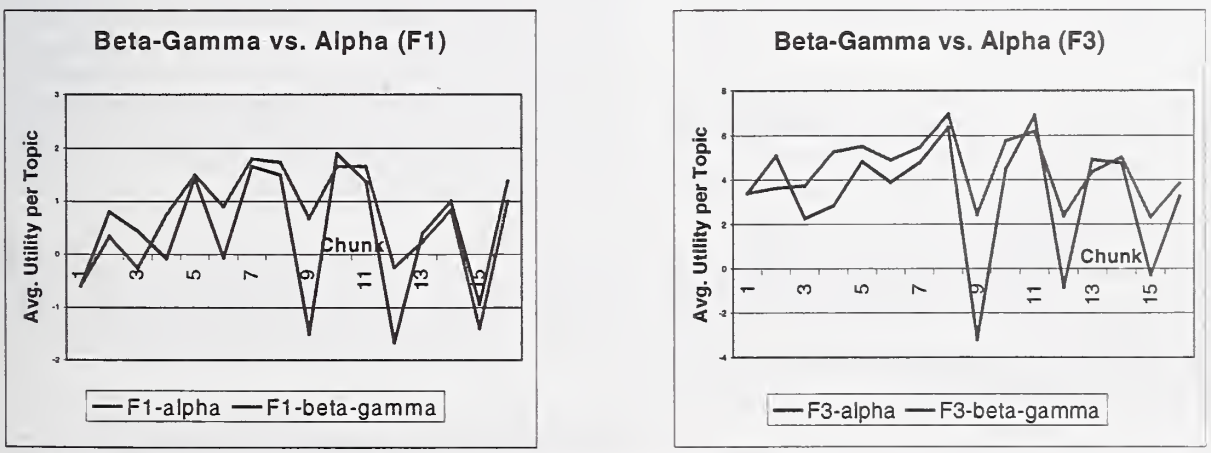


Figure 2. Comparison of chunk utility of beta-gamma regulation and alpha regulation

6 Batch Filtering

In this section we describe, more briefly, our experiments on batch filtering task.

6.1 Experimental Procedure

For Batch Filtering, we used essentially the same system that we used for Adaptive Filtering. The differences were as follows.

- As initial profiles, we took term vectors trained over AP88 instead of the vectors generated from the topic description.
- We used different chunk sizes.[5]
- AP88 was used as the initial reference corpus (for IDF, and delivery ratio).
- We used (standard) subdocument indexing.

As in adaptive filtering, we used the delivery ratio method to estimate an initial threshold. For this threshold estimation, the relevance judgments available for AP88 were not used.

6.2 Analysis of TREC-7 Batch Filtering Results

We submitted two runs (one each for F1 and F3) for the batch filtering task. Our runs were both clearly below median, though better for F3 than for F1.

By itself such a poor performance is not a surprise, as we did not exploit the complete set of relevance judgments on AP88 to establish a better initial threshold than the delivery ratio threshold. But a direct comparison showed that our adaptive runs over the same data would in fact have achieved a better performance, and instead of significantly below median, would have been very near median instead. There are then two questions:

1. Do the batch filtering runs perform worse as a result of lower profile quality (i.e., a problem with the training method)?
2. If not, what is the reason for the observed performance hit?

We tried to settle the first question by means of follow-up experiments.

6.3 Post-TREC Experiments

One possible hypothesis is that the profiles obtained by adaptive filtering at the end of AP88 are better than the profiles obtained from batch training over the same corpus. To test this hypothesis, we compared three different versions of initial profile vectors:

A. The original (untrained) profile vector ("NoTrain").
B. A trained profile vector based on adaptive filtering over judgments for accepted documents in AP88 ("AdaptTrain").
C. A trained profile vector using batch training over all judgments in AP88 ("BatchTrain").

Per topic, the initial threshold was kept constant. Evaluation was based on the average utility over all 50 topics over AP89 and AP90. We found that BatchTrain performed better than AdaptTrain, which was in turn better than NoTrain. This is as it should be, since BatchTrain uses more training examples than AdaptTrain, and NoTrain does not use any training at all.

Another way to evaluate relative performance is to compare the number of topics in which one method outperforms another. Again, AdaptTrain was clearly better than NoTrain, though now the difference between BatchTrain and AdaptTrain was less clear for F1.

We can conclude that the hypothesis stated above is false (the answer to our first question is "no") and, therefore, we need to look elsewhere for explanations of observed performance.[6]

6.4 Topic-By-Topic Analysis

We have not as yet performed a thorough topic-by-topic analysis, but such an analysis might prove to be very interesting in general. A classification of topics according to different criteria, and an analysis of which filtering methods work better for which class of topics can only lead to more insight and better filtering systems.

In the batch filtering context, a cursory inspection of relative per-topic performance showed that a big hit in performance was due to under-delivery for one topic in particular (topic 22). This topic was, with over 800 relevant documents, the topic with the highest density in relevant documents (and hence the highest median utility). Smaller hits occurred for other high-density topics. This under-delivery points at some specific characteristics of the methods we used, and is the result of a combination of the delivery ratio method and the beta-gamma regulation.

6.5 Discussion: Defects and Remedies

To explain this, we need to look into the beta-gamma regulation method in some more detail. A critical observation is that, when the threshold is set considerably higher than optimal for the topic, many relevant documents score below threshold. These documents are considered non-relevant by the system when it computes the optimal and zero-threshold. In such cases, the zero-threshold may be well above the true optimal threshold. Although the beta-gamma algorithm will lower the threshold in small steps, it may take many updates before the threshold approaches the true optimal threshold for the profile.

This phenomenon occurs in situation where the initial threshold is too high, for example because the estimated ratio of delivery was underestimated (forcing a higher score). This occurs precisely in high-density queries.

[5] AP89 was broken up in 6 chunks consisting of 5,000, 8,000, 11,000, 16,000, 20,000, and approximately 25,000 documents, respectively.

[6] Incidentally, the data do suggest that a better initial profile leads to more effective learning in later updating stages. This hypothesis certainly warrants further analysis.

Both our batch-mode and adaptive mode runs suffered from this effect. But the situation was much worse for batch filtering because there were many more threshold updates for the adaptive runs (16) than for the batch mode runs (6). In addition, our batch filtering system made no use of relevance information over AP88 for initial threshold setting, whereas some other systems did.

Although there are potential risks associated with a less conservative initial threshold setting, we could try to improve our system in the following ways.

1. Estimate the individual topic density from the AP88 corpus and use this density to obtain a different delivery ratio for each topic.
2. Use smaller chunks, or a real document-based filtering system to allow more rapid detection of and adjustment to underdelivery.
3. Use the shape of the utility function over a sorted list of accepted documents to estimate density. Although this function tends to behave rather chaotically, it may still be possible get a rough estimate of the topic density and take action in extreme cases.

Each of these aspects of the process suggests interesting directions for further study.

7 Summary and Further Work

We evaluated the basic CLARIT adaptive filtering approach by participating in the TREC-7 Adaptive and Batch Filtering tasks. Our results show that using our standard retrieval and routing techniques in combination with heuristic threshold setting leads to reasonably good performance. Three major positive contributors to this performance were (1) a heuristic beta-gamma threshold regulation algorithm, (2) the use of increasing chunk sizes, and (3) the elimination of difficult topics. Our results also suggest that the delivery ratio method is an effective initial thresholding method. Less clear at this point is the benefit of a better initial term vector.

In the future, we intend to study in more detail the behavior of the beta-gamma threshold regulation algorithm, in particular, how its effectiveness varies with different topics. One example is the problem of slow learning for high density topics that may have damaged our performance. Another is the possibility of a combination with logistic regression for density estimation, which showed some promise in our approach to filtering in TREC 6. [4]

We also intend to investigate actual real-time filtering algorithms, as well as profile-specific updating. Other interesting aspects of filtering are related to an intelligent exploitation of historical training data based, for example, on recency and confidence. Finally, we believe that topics related to the learning effect and behavior over time, such as user interest drift and topic tracking, are important future research issues.

Acknowledgements

We are indebted to Dr. Alison Huettner for her comments on earlier drafts of this paper and to Ms. Lisa Stewart for help with the layout and formating of the final document.

References

1. Evans, David A., Kimberly Ginther-Webster, Mary Hart, Robert G. Lefferts, Ira A. Monarch, "Automatic Indexing Using Selective NLP and First-Order Thesauri". In A. Lichnerowicz (Editor), *Intelligent Text and Image Handling. Proceedings of a Conference, RIAO '91.* Amsterdam, NL: Elsevier, 1991, 624-644.

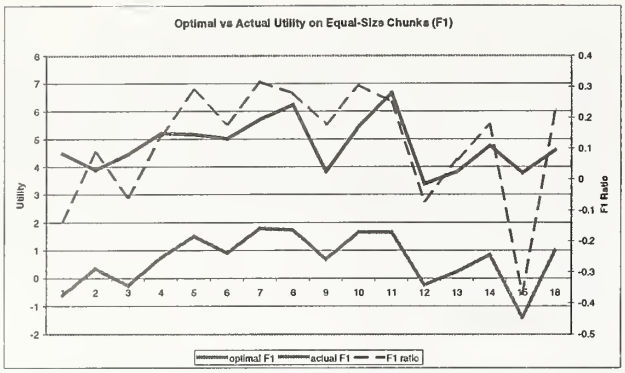
2. Evans, David A., and Robert G. Lefferts, "CLARIT-TREC Experiments". *Information Processing and Management*, Vol. 31, No. 3, 1995, 385-395.

3. Evans, David A., Alison Huettner, Xiang Tong, Peter Jansen, and Jeff Bennett, "Effectivenss of Clustering in Ad-Hoc Retrieval," This Volume.

4. Milic-Frayling, Natasa, Chengxiang Zhai, Xiang Tong, Peter Jansen, and David A. Evans, "Experiments in Query Optimization, the CLARIT System TREC-6 Report". In Voorhees, E.M., and D.K. Harman (Editors), *The Sixth Text REtrieval Conference (TREC-6)*. NIST Special Publication 500-240. Washington, DC: U.S. Government Printing Office, 1998, 415-454.

5. Rocchio, J.J., "Relevance Feedback in Information Retrieval", In: Salton, Gerard (Editor), *The SMART Retrieval System*, Prentice-Hall, Englewood NJ. 1971, 313-323.

6. Salton, Gerard, *Automatic Text Processing*, Addison-Wesley, Reading, MA, 1988.



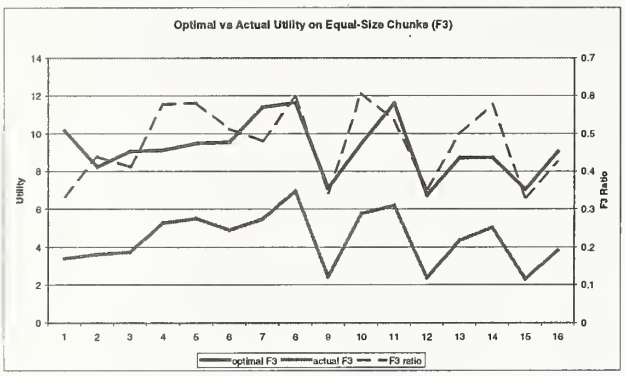
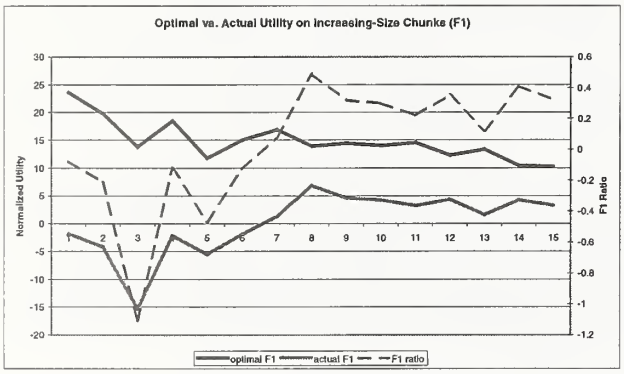


Figure 3a. Learning effect: optimal vs. actual utility for equal-size chunks (F1 and F3).



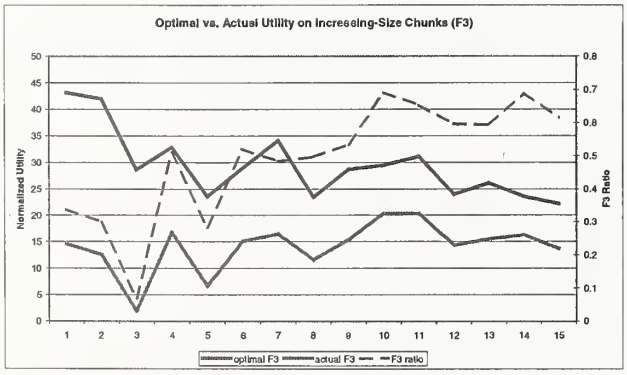


Figure 3b. Learning effect: optimal vs. actual utility for increasing-size chunks (normalized) (F1 and F3).

Ad hoc and Multilingual Information Retrieval at IBM

Martin Franz, J. Scott McCarley, Salim Roukos
IBM T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
<franzm,jsmc,roukos>@watson.ibm.com

January 27, 1999

1 Introduction

IBM participated in two tracks at TREC-7: ad hoc and cross-language. In the adhoc task we contrasted the performance of two different query expansion techniques: local context analysis and probabilistic model. Two themes characterize IBM's participation in the CLIR track at TREC-7. The first is the use of statistical methods. In order to use the document translation approach, we built a *fast* (translation time within an order of magnitude of the indexing time) French⇒English translation model trained from parallel corpora. We also trained German⇒French and Italian⇒French translation models entirely from comparable corpora. The unique characteristic of the work described here is that all bilingual resources and translation models were learned automatically from corpora (parallel and comparable.) The other theme is that the widely varying quality and availability of bilingual resources means that language pairs must be treated separately. We will describe methods for using one language as a pivot language in order to decrease the number pairs, as well as methods for merging the results from several retrievals.

2 Adhoc

2.1 System Description

We used two different multi-pass strategies in TREC-7 automatic ad-hoc experiments, both of them based on improving the document scores given by the Okapi formula [1] by combining them with scores obtained with expanded queries. To

construct the expanded queries we tried the local context analysis approach [2] and also the probabilistic model [3].

The data preprocessing stage was the same as the one applied in our TREC-6 system and described in [4]. We used statistical tokenizer, part-of-speech tagger [5] and morphological analyzer on both the description fields of the queries and content bearing fields of the documents. Filler query prefixes were filtered out by mechanism similar to the one described in [4]. We have collected unigrams and bigrams based on the morphed data using a 540459 word vocabulary and a list of 514 stop words.

We used Okapi formula [1] for the first-pass scoring the same way as in [4]. Unigrams and bigrams in the intersection of the query and document contributed a score of:

$$s = \frac{tf}{c_1 + c_2 \times \frac{dl}{avdl} + tf} \times w^{(1)}, \tag{1}$$

where tf and qtf are the document and query counts for a given n-ngram, dl is the document length, $avdl$ is the average length of the documents in the collection and $w^{(1)}$ is the inverse document frequency, computed as:

$$w^{(1)} = log(\frac{N - n + 0.5}{n + 0.5}),$$

where N is the total number of documents in the corpus and n is the number of documents containing a given n-gram. In the Eq.(1) we used $c_1 = 0.5, c_2 = 1.5$ for unigram scoring and $c_1 = 0.05, c_2 = 0.05$ for the bigrams. The first pass score was a linear combination of unigram and bigram scores given by Eq.(1), with the unigram scores weight set to 0.8 and bigram scores weight equal to 0.2. First pass results for query description and title fields are summarized in Table 1, line 1 and Table 2 line 1, respectively.

2.2 Query Expansion with Local Context Analysis

In this experiment we applied an approach similar to the one described in [2], with some modifications in the way the inverse document frequencies were handled and in expanded terms weighing.

We used passages of 200 non-stop words, overlapping by a half of their length. The original queries were expanded by adding 100 unigrams based on top 100 passages.

The expanded queries were used to score both the documents and the passages with Okapi formula. Passage scores were later converted into new document scores in a way where the document score was given by the score of its highest scoring passage. Final scores were obtained as weighed combination of the two, with the ratio between document and passage scores set to 40/60. The results of these experiments for query description and title fields are listed in Table 1, line 2 and Table 2 line 2, respectively.

	TREC-5		TREC-6		TREC-7	
	AveP	P20	AveP	P20	AveP	P20
pass1	0.1757	0.2650	0.1769	0.3050	0.1865	0.3760
LCA, os	0.2010	0.3050	0.2057	0.3090	0.2336	0.4010
px, ps	0.1951	0.2850	0.1901	0.3010	0.2075	0.3770
px, os	0.1974	0.3000	0.1863	0.3070	0.2047	0.3750

pass1: first pass Okapi scoring, unigram nad bigram terms
LCA: local context analysis query expansion
px: probabilistic model query expansion
ps: second pass probabilistic model scoring
os: second pass Okapi scoring

Table 1: Results of experiments on TREC-5, TREC-6 and TREC-7 sets: ad-hoc, automatic, short topics.

	TREC-5		TREC-6		TREC-7	
	AveP	P20	AveP	P20	AveP	P20
pass1	0.1234	0.1820	0.1943	0.3250	0.1749	0.3440
LCA, os	0.1714	0.2520	0.2392	0.3390	0.2502	0.3720

pass1: first pass Okapi scoring, unigram nad bigram terms
LCA: local context analysis query expansion
os: second pass Okapi scoring

Table 2: Results of experiments on TREC-5, TREC-6 and TREC-7 sets: ad-hoc, automatic, title.

2.3 Query Expansion with Probabilistic model

Our probabilistic model based query expansion technique was the same as the one described in [4]. After the first pass Okapi scoring, top 40 documents were used to determine the additional unigrams, by thresholding the probabilistic model scores. New bigrams were the ones found in at least 15 of the top 40 documents.

We tried using both probabilistic model and Okapi formula for second pass scoring with expanded queries. Both the first and second pass scores were modified using the method for scoring correlated features, described in [6]. Scores of the original and expanded terms were then combined using 80/20 weighing ratio. The results of these test runs may be found in Table 1, lines 3 and 4.

2.4 Conclusion

We have experimented with various query expansion and scoring algorithms in the context of TREC-5, TREC-6 and TREC-7 tasks. All the query expansion methods brought an improvement of average precision over the baseline first pass Okapi scoring. Among the query expansion methods, LCA technique caused the most significant benefit, with the two of probabilistic model based methods bringing roughly the same but smaller improvement.

3 Crosslanguage track

3.1 Introduction

IBM's participation in the cross-language track at TREC-7 involved building separate systems for all four document languages : English, French, German, and Italian. We focused our attention on the English queries, although the techniques we studied would also have been applicable to the other query languages. Four runs were submitted, covering both long and short queries. ("Long" queries used all three fields, <Title>, <Description>, and <Narrative>. "Short" queries used just the traditional <Description> fields.) All query processing was fully automatic. We varied our strategy somewhat between runs: this paper will focus on the techniques used in runs *ibmcl7cl* and *ibmcl7cs*. A unifying theme of these runs is the extensive use of statistical methods, reflecting the long history of statistical approaches to machine translation in our group. [7] In fact, all bilingual dictionaries and translation models used in these runs were learned automatically from corpora. [1] We treated each document language as a separate IR system. Unlike last year's task [8], this year's task involved merging the ranked lists of documents from each system.

Our overall approach to cross-language information retrieval has been to translate the documents, rather than queries, since there is more varied context in the documents. Once the documents have been translated, we use familiar IR techniques such as the Okapi formual [1], and probabilistic models [3] that have been successful used by our group in the ad-hoc tasks at previous TREC's. [9, 4] Most of our work in cross-lingual retrieval has focused on French. We developed a "Fast Document Translation" algorithm that was trained on a parallel corpus, incorporated word sense disambiguation (also learned from the parallel corpus) and by ignoring word order, was able to translate the entire French section of the SDA in a reasonable amount of time (in fact, within an order of magnitude of the amount of time spent indexing the collection!) For retrieval from Ger-

[1] The runs *ibmcl7al* and *ibmcl7as* used all of the above methods, but also incorporated query translation from English to German and Italian using Altavista (Systran, http://babelfish.altavista.digital.com). The motivation was that different translation systems would complement each other. The incorporation was through a linear combination of scores. The result was a modest improvement in overall performance.

man text, we did not have a parallel corpus available, so we used comparable corpus methods to create appropriate training data for our machine translation methods. Italian was treated identically to German. We also studied the use of French as pivot language, so that we could combine our resources for retrieving French documents with an English query with our resources for retrieving German documents with a French query to produce a system for retrieving German documents with an English query. Finally we also explored simple schemes for merging disjoint sets of documents retrieved from different IR system.

3.2 Statistical Machine Translation

The statistical approach to machine translation assumes that with any pair of English and French sentences (E, F) (of length $|E|$ and $|F|$ words, respectively), with $E = e_1 ... e_{|E|}$ and $F = f_1 ... f_{|F|}$ one can associate a probability that E is a translation of F. The most probable English translation E of a given French sentence is then given by

$$\hat{E} = \arg\max_{E} P(E|F). \tag{2}$$

Modeling $P(E|F)$ depends upon being able to factor it into terms representing individual pairs of words. This factorization is accomplished by introducing a word-by-word alignment between the sentences, motivated by the idea that there are many words in one language ("perfume") which are highly correlated with a word in the other language ("parfum"). We denote the alignment of a sentence pair as A. A typical representation of the alignment is to assign to each word e_i in E an integer $a_i \in \{1...|F|\}$ indicating that it is associtated with f_{a_i}.

There are many ways to factor Eqn. (2) into terms involving words. We follow [7] and introduce Model 1

$$p(E, A|F) = \frac{\epsilon}{(|E|+1)^{|F|}} \prod_i t(e_i|f_{a_i}) \tag{3}$$

As originally described, Model 1 was used in a source-channel framework. This approach is computationally expensive and therefore difficult to incorporate into a document-translation based IR system. The principle difficulty is that the search space in Eq. (2) covers variation in word order and other features that are largely irrelevant to information retrieval. However, extracting a bilingual dictionary from the trained model is easy: for each French word f tabulate the English word e that maximizes $t(e|f)$.

3.3 "Fast Translation"

We have extended Model 1 into a more versatile method that is able to translate phrases and to disambiguate the sense of words during translation. [10] In

order to incorporate context into the model, we note that the existence of the alignment A allows each English word to have a context not only of surrounding English words, but also of French words surrounding the word to which it is aligned. We denote the number of values of i for which $a_i = j$ as the *fertility* n_j of the j'th word in F. We have proposed a different decomposition of the basic equation into word-by-word terms:

$$p(E, A|F) = \left[\prod_{i=1}^{|F|} p_n(n_i|n_1^{i-1}, F)\right] p_a(A|N, F) \left[\prod_{j=1}^{|E|} p_s(e_j|e_1^{j-1}, A, F)\right] \quad (4)$$

where the fertilities $n_1, ...n_{|F|}$ are collectively denoted N.

In order to generate the translation of a French sentence, one first picks the fertilities of the French words with probabilities p_f (as opposed to the total number of English words, as in Model 1), then one picks an alignment with probability p_a as constrained by the fertilities. Finally, English words are picked with probabilities p_s as translations of the French words, based on the context in the French sentence. Different "senses" or meanings of the French word, as disambiguated by its context, are reflected in the different choices of Enlgish words generated (i.e. *pomme* may be rendered as *potato* or as *apple* depending upon whether or not it is followed by *de terre*.) This model is trained under the observation that most of the probability is likely to be associated with the Viterbi alignment (or at most a few neighboring alignments) [7]. We note that an approximate alignment can be easily computed from many other models, and have had considerable success using alignments computed from Model 1, as described above. (Alignment probabilities are easily found from a rearrangement of Eq. (3).) Approximating the fertility and sense terms so that only local context matters leaves us with fertility and sense models which are simply 4-gram langauge models:

$$p_n(n_i|n_1^{i-1}F) \approx p_n(n_i|f_i, f_{i-}, f_{i+}) \quad (5)$$

$$p_s(e_j|e_1^{j-1}, A, F) \approx p_s(e_j|f_{a_j}, f_{a_j-}, f_{a_j+}). \quad (6)$$

Here we will take local context of a word f_i to be the previous and next non-stop words, denoted f_{i-} and f_{i+}, respectively. and treat the middle factor on the right-hand-side of Eq. (4) as an irrelevant constant. The translation model is completely specified by these functions. The two functions that must be modeled are simply conditional probabilities of the occurrence of a word given some information about the local context of the word, a problem familiar from speech recognition. [11]

In order to translate French text with model, for each French word, the fertility is predicted with p_n. Then, p_s is used to select which n of several possible choices of English words are likely translates. The resulting translation is incorporated into our information retrieval system by simply indexing the translated documents. Translating the 3 years of SDA newswire required an average of 28

hours for each year of newswire text on an RS-6000 Model 590. This translation rate is much faster than other published accounts of using document translation on corpora of comparable size [12] and, in fact, is within an order of magnitude of the amount time spent on other processing of the documents (part-of-speech tagging, morphological stemming, building the inverted index, etc.) The combined (MT+IR) system achieved an average precision of 0.3400 on TREC-6 long queries, the best result of which we are aware on that query set. Incorporating the fertility and sense models results in an 18% – 19% improvement in average precision over merely using the statistical constructed dictionary implicit in Model 1.

3.4 Comparable Corpora

The English-French retrieval system was trained using a parallel corpus, a particularly valuable linguistic resource. An important issue is whether a similar system can be trained using more readily available linguistic resources, such as a comparable corpus. A comparable corpus differs from a parallel corpus in two important aspects. First the similar documents in the corpus are not translations of each other, but are composed independently. Second, these comparable documents may not be aligned with each other. The SDA newswire itself is an example of a comparable corpus. The French, German, and Italian sections are not translations of each other, but since they report news events from the same time periods (including local Swiss events), we nevertheless expect that articles about the same event contain correlations useful for the training of statistical machine translation algorithms.

In order for this corpus to be useful for machine translation purposes, we select an aligned subset of comparable articles and treat it as a parallel corpus. (Another approach to comparable corpora involves comparing the context of words across the languages without aligning specific articles. [13]) Our approach is motivated by the observation that names are frequently spelled identically in French, German, and Italian. Passages that contain the same name (or better, the same names) even though they are in different languages, are more likely to be about the same event. Of course, names that are more common are less informative. Such an approach to comparable corpora alignment has already been utilized. [14] These features suggest the following algorithm:

(1) Index the French and German SDA into passages of, for example, 50 words.

(2) Formulate an initial dictionary of bilingual word-pairs (either known translates or words that are spelled identically in both languages)

(3) Compute Okapi scores of documents in one language against those in the other, counting those word-pairs from the bilingual dictionary as equivalent or matching. It is convenient to score only those documents that were published on approximately the same date. For each French passage, retrieve the best German passage. For each German passage, retrieve the best French passage.

(4) If a French and a German passage retrieve each other, regard them as aligned.

(5) Treat the subset of aligned German and French passages as training data for the machine translation.

(6) Train a machine translation system, and extract a new dictionary of bilingual word-pairs.

(7) Repeat starting at step (3).

We performed two alignments with this procedure: French-German, and French-Italian. Both were seeded with identically spelled words. After two iterations we obtained 23261 "aligned" French and Italian passages, and 90453 "aligned" French and German passages. There were approximately 35% more "aligned" articles after the second iteration than after the first. We did not check the quality of the alignments, but regarded them as a test of our ability to train a machine translation system with a noisy training data. Translating the German corpus (including the NZZ) into French with the dictionary produced by this method, and retrieving using the French TREC-6 queries (long version) produced an average precision of 0.2361, which was about 68% of our (German) monolingual performance. As a percentage of monolingual performance, this was similar to that obtained with a French-English dictionary constructed from parallel corpora, although we caution that the tasks are not strictly comparable.

3.5 Pivot Language

Having developed resources for retrieving French documents given English queries, and for retrieving German documents given French queries, it is desirable to be able to combine these resources in order to retrieve German documents given English queries. There are several methods of combining these resources.

(1) Direct translation: Combine the German⇒French translation system with the French⇒English translation system directly, by translating the German documents into French, and then translating them into English.

(2) Convolution: Convolve the German⇒French translation system with the French⇒English translation system to obtain a German⇒English translation system. This operation is suggested by the mathematical structure of the translation model.

$$t(g|e) = \sum_f t(g|f)t(f|e) \tag{7}$$

In effect, we sum over several possible translations in the intermediate language.

An alternative approach is to combine the information retrieval systems themselves, rather than the underlying translation systems, by using query expansion. An appealing feature of this method is its generality: different implementations of cross-language IR systems (document translation, query translation, LSI, etc.) can be combined. Our approach is as follows:

(1) Use the English query in the English-French CLIR system to retrieve French documents.

system	AveP	P20	%baseline
1	0.3478	0.4136	100%
2	0.2361	0.2955	68%
3	0.1577	0.1977	45%
4	0.1301	0.1455	37%
5	0.2295	0.2636	66%

Table 3: Retrieval from German documents, TREC-6 long queries: (1) = German queries (monolingual) (2) = French queries (G⇒F translation), (3) = English queries (G⇒F⇒E translation) (4) = English queries, convolution translation, (5) = English queries, French query expansion (see text).

(2) Formulate a French query based on the top-n French documents.

(3) Use this query in the French-German CLIR system to retrieve German documents.

Although more sophisticated query-expansion techniques could easily be incorporated, we formed the new French query by simply merging all non-stopwords in the top-n French documents. We found that $n = 3$ worked much better than $n = 2$ or $n = 1$, and that there was a relatively smaller loss in average precision for $n > 3$. We found that the query-expansion technique substantially outperformed the methods involving combining the translation models (see table 3.)

3.6 Merging

We implemented the English-French and English-German retrieval systems as described above, guided by the results of the above experiments. We implemented the English-Italian system by blindly following the structure of the English-German system. Since the goal of the CLIR track is to produce a single list of relevant documents across all languages, it is necessary to merge the results from each system. Scores produced by the Okapi formula (or similar IR formulae) are not directly comparable, because of the different languages and differing quality of the underlying translation resources. What is needed is a simple means to estimate the probability that a document D is relevant $Pr(\rho)$ based on previous performance of the IR system. Note that probabilistic models such as [3] are not comparable either, even though the mathematics suggests that they are modeling the probability of relevance. In fact, because they are trained by pseudorelevance feedback (off of first-pass Okapi scores) they are no more comparable than the scores of the first pass.

We can estimate probability of relevance from precision at rank $P(R)$ by

$$(R+1)P(R+1) - RP(R) = Pr(\rho|R) \tag{8}$$

Because of the very limited amount of training data available, it is essential that

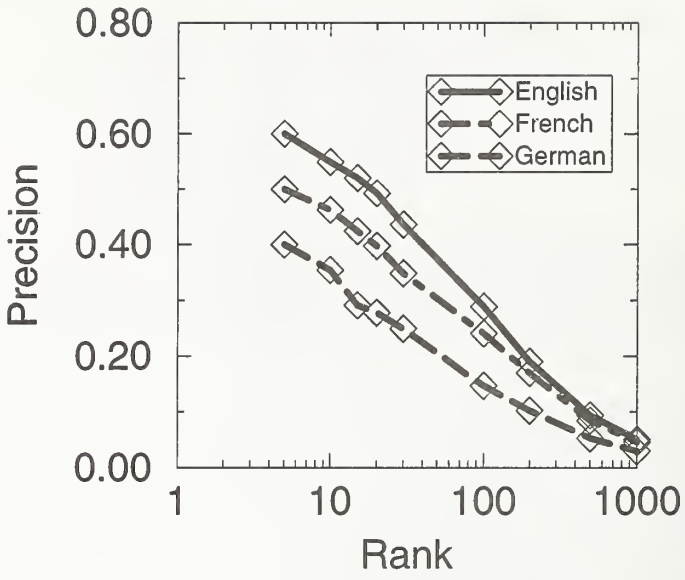


Figure 1: Precision vs. log(Rank)

	TREC-7 (short)	TREC-7 (long)
interleave	0.1912	0.2574
merged	0.2212	0.2942
% rel.gain	15%	14%

Table 4: Merging of documents: interleaving (top) and modeling a system-wide probability of relevance (bottom).

we use only a few parameters to describe each system. We note in Figure 2 that precision is approximately a linear function of the $logR$. (We do not claim that there is an underlying scaling law, or that we expect the linearity to hold over more decades, merely that this is a simple interpolation scheme that allows us to describe the precision of the system, using only two parameters.) Thus we can estimate the probability of a relevance of a document as simple function only of its rank in the original retrieval. Thus we can merge a disjoint set of documents retrieved from different systems by sorting on the estimate of $P(\rho)$. We note that this procedure does not use any information about the magnitude of the scores. Furthermore it merges documents in the same proportions and in the same order for all queries.

We calibrated our merging system on the TREC-6 queries against French and German documents, and assumed that Italian would be similar to the German. We compare this estimate from the simple interleaving of equal numbers of

documents that would be obtained if $Pr(\rho|R)$ is chosen to be any arbitrary decreasing function of R identical for all systems. The result was a substantial improvement in overall performance, even though the system was calibrated on the results of TREC-6 queries.

3.7 Conclusion

We emphasize that all of the methods described here are statistical in nature and that all bilingual lexicon used were learned automatically from corpora. Although statistical machine translation has long relied on parallel corpora, we have shown how these methods can also be extended to non-parallel, comparable corpora. Since linguistic resources vary widely in both size and quality between language pairs, it is necessary to develop separate systems for each language pair. Therefore we have also developed methods to address the merging problem, and successfully used a pivot language in order to reduce the number of language pairs.

4 Acknowlegments

This work is supported by NIST grant no. 70NANB5H1174.

References

[1] S.E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, M. Gatford, "Okapi at TREC-3" in *Proceedings of the Third Text REtrieval Conference (TREC-3)* ed. by D.K. Harman. NIST Special Publication 500-225, 1995.

[2] J. Xu and W. B. Croft 1996 Query Expansion Using Local and Global Document Analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* Zurich, Switzerland, pp. 4-11.

[3] Ernest P. Chan, Santiago Garcia, Salim Roukos 1998 Probabilistic Modeling for Information Retrieval with Unsupervised Training Data. In *Proceedings, The Fourth International Conference on Knowledge Discoveryand Data Mining,* AAAI Press, pg.159.

[4] M. Franz and S. Roukos, "TREC-6 Ad-hoc Retrieval", in *The 6th Text REtrieval Conference (TREC-6).*

[5] B. Merialdo 1990 Tagging text with a probabilistic model. In *Proceedings of the IBM Natural Language ITL,* Paris, France, pp. 161-172.

[6] M. Franz, S. Roukos 1998. A Method for Scoring Correlated Features in Query Expansion. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* Melbourne, Australia, pp. 337-338.

[7] P. F. Brown et al. "The mathematics of statistical machine translation: Parameter estimation", *Computational Lingustics*, 19 (2), 263-311, June 1993.

[8] D.Harman and E.Voorhees, "Overview of the Sixth Text REtrieval Conference (TREC6)",in *The 6th Text REtrieval Conference (TREC-6).*

[9] E.Chan, S.Garcia, S.Roukos, "TREC-5 Ad Hoc Retrieval Using K Nearest-Neighbors Re-Scoring" in *The 5th Text REtrieval Conference (TREC-5)* ed. by E.M. Voorhees and D.K.Harman.

[10] J.S. McCarley and S.Roukos, "Fast Document Translation for Cross-Language Information Retrieval", in *Machine Translation and the Information Soup* ed. by D.Farwell, L.Gerber, and E.Hovy. (1998)

[11] L.R. Bahl, F.Jelinek, and R.L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition", in *IEEE Transactions on Pattern Analysis and Machine Intelligence* 5 (2), 1983.

[12] D.W. Oard, P.Hackett, "Document Translation for Cross-Language Text Retrieval at the University of Maryland", in *The 6th Text REtrieval Conference (TREC-6)* ed. by E.M. Voorhees and D.K.Harman.

[13] P.Fung, "A Statistical View on Bilingual Lexicon Extracction: From Parallel to Non-parallel Corpora", in *Machine Translation and the Information Soup* ed. by D.Farwell, L.Gerber, and E.Hovy. (1998)

[14] B.Mateev, E.Munteanu, P.Sheridan, M.Wechsler, and P.Schäuble, "ETH TREC-6: Routing, Chinese, Cross-Language and Spoken Document Retrieval" in in *The 6th Text REtrieval Conference (TREC-6)* ed. by E.M. Voorhees and D.K.Harman. (1997)

TREC-7 Evaluation of Conceptual Interlingua Document Retrieval (CINDOR) in English and French

Anne Diekema, Farhad Oroumchian
Páraic Sheridan, Elizabeth D. Liddy

TextWise LLC
2-212 Center for Science & Technology
Syracuse, NY 13244

Abstract:

TextWise LLC. participated in the TREC-7 Cross-Language Retrieval track using the CINDOR system, which utilizes a *"conceptual interlingua"* representation of documents and queries. The current CINDOR research system uses a conceptual interlingua constructed around the Princeton WordNet, which we are mapping into French and Spanish. The use of an interlingual representation of documents and queries allows us to perform retrieval on any combination of supported languages, rather than having to rely on pairwise translations, while the use of a resource like WordNet allows us to match equivalent terms (including synonyms) across languages.

Although the analysis of our TREC-7 results is clouded somewhat by the kinds of system errors which inevitably occur in a first-time evaluation over large TREC corpora, our evaluation of the conceptual interlingua approach suggests that it provides highly effective cross-language retrieval performance. In particular, we notice that the CINDOR system achieves cross-language retrieval results equivalent in many cases to corresponding monolingual queries, without the loss in retrieval precision observed in many other approaches to cross-language retrieval.

Future work on the CINDOR system, which was evaluated here in its research prototype form, will focus on improving further the coverage of our conceptual interlingua resources and the efficiency of our document processing modules. We are also investigating the construction of an interlingual resource of proper nouns, using technology from other TextWise products, since proper nouns constitute the largest category of 'out-of-vocabulary' terms with respect to our current conceptual interlingua knowledge base. We will also continue to adapt the CINDOR system to handle more languages.

1. Introduction.

The TREC-7 Cross-Language Retrieval track was set as the arena for the very first evaluation of the CINDOR system, a prototype retrieval system developed by TextWise LLC. with the explicit aim of facilitating language-independent document retrieval based on natural language concepts. The CINDOR system currently supports retrieval over English, French and Spanish with a fourth, yet to be determined, language due for inclusion from the beginning of 1999. In contrast to many approaches to cross-language retrieval, which translate either documents or queries between the languages in question [Oard & Diekema 1998], CINDOR takes the approach of translating *both* documents and queries - into its interlingual representation. This has the advantage of permitting matching and retrieval based on any combination of the languages involved, rather than relying on pairwise translations between language pairs.

The current CINDOR system is built around Princeton WordNet version 1.5 [Miller 1990], with the conceptual interlingua consisting of WordNet synset ids. We use the term "conceptual interlingua" to refer to a knowledge base of language-independent concept representations. Our current conceptual interlingua is a hierarchically organized concept lexicon, following WordNet. Concepts in the hierarchy are considered to be essentially language neutral and are then linked to their relevant terminological instantiations in various languages. A more detailed examination of this conceptual interlingua resource is presented in Section 2.

The use of a conceptual resource like WordNet for document retrieval raises specific issues in the indexing and matching of documents and queries. Although there is the advantage of being able to directly match term synonyms rather than only words with identical surface (or lexical stem) form, this can quickly turn to disadvantage without a mechanism for distinguishing between different word *senses*. While it has been claimed that document retrieval as an application is remarkably robust to word sense differences [Sanderson 1997], we believe that retrieval performance would be unlikely to remain unaffected if multiple word senses were further expanded by synonyms and the whole lot then treated as a 'bag of words' for retrieval. Our initial experiments have confirmed that a naïve approach to WordNet indexing is indeed ineffective and the word sense problem must be tackled in some way. This aspect of our work, together with other retrieval issues we have encountered in our approach, is presented in Section 3.

The CINDOR project is currently at the half way point in a two-year development and evaluation exercise. Work to date has focused on construction of the conceptual interlingua resource, design and implementation of system modules, and first-round benchmark evaluation of the system using the TREC-7 data reported here. Over the next year, we plan to significantly improve system operating performance based on the lessons learned in this evaluation, continue to increase the coverage of our language resources, add a fourth language, and develop a dedicated module for the recognition and processing of proper nouns. Each of these lines of development will end with a benchmark evaluation to measure the improvements gained over the results presented here.

Within the context of the TREC-7 Cross-Language Retrieval track, we have evaluated the CINDOR system against both the English and French data. This constitutes only a subset of the primary track evaluation, which also included data in German and Italian. Although the primary evaluation objective of the track was to evaluate systems against all four of the track languages, the track definition from the beginning included provision for groups which were not interested in working with all four languages. Further, although we did submit merged runs with results from combined English and French document collections, our primary evaluation focus has to date been constrained to single language pairs. Although the CINDOR system will naturally support retrieval from multilingual document collections, given the language-independent nature of its retrieval algorithms, the individual English and French language subcollections of the TREC-7 data were indexed independently and separately in this round of evaluation. We feel that the objective of this, our first evaluation, should be to understand completely the performance of cross-language retrieval based on our conceptual interlingua approach. We can more easily gain insight into this performance by analyzing single language pair retrieval experiments rather than multiple language runs. This objective is reflected in the analysis of our TREC-7 performance which we report in Sections 4 and 5, followed by our conclusions in Section 6.

2. Conceptual Interlingua.

The use of a lexical resource such as WordNet for document retrieval has been extensively investigated, based on a belief that such a resource can provide language knowledge with the potential for increased retrieval performance. This potential has been identified by [Gonzalo *et al* 1998] as:

- the possibility to discriminate word senses in documents and queries. This would prevent matching *'spring'* in its *"metal device"* sense with documents mentioning *'spring'* in the sense of *"springtime"*, and then retrieval accuracy could be improved.

- WordNet provides the possibility of matching semantically related words. For instance, *'spring'*, *'fountain'*, *'outflow'*, *'outpouring'*, in the appropriate senses, can be identified as occurrences of the same concept, *'natural flow of ground water'*. And beyond synonymy, WordNet can be used to measure semantic distance between occurring terms to get more sophisticated ways of comparing documents and queries.

Much of the research on WordNet for information retrieval, has in fact focused on the second source of potential; query expansion and semantic relatedness. Investigation of the usefulness of WordNet for word sense disambiguation has taken place in the context of much research in the information retrieval community which has concluded that retrieval is in fact robust to a large degree of sense ambiguity, for example most recently by [Sanderson:97]. This situation is exacerbated when WordNet sense disambiguation experiments lead to the conclusion that, "missing correct matches because of incorrect sense resolution has a much more deleterious effect on retrieval performance than does making spurious matches" [Voorhees:93]. The obvious conclusion is that if WordNet is to be used for sense disambiguation in a retrieval application, then one must proceed carefully.

The conclusions to be drawn from research on the use of WordNet for query expansion suggest an equal amount of caution when pursuing this potential, especially when working in the TREC environment [Voorhees:94],[Voorhees:93]. While there was some suggestion of a query expansion benefit in small document collections, the experimental results from the TREC environment suggested that query expansion made little difference, presumably because the TREC topic statements already provided such a rich description of the information being sought. WordNet also contributed to a loss of retrieval effectiveness in experiments reported by [Richardson & Smeaton 1995]. In an application involving short documents and queries however, specifically with image captions, Smeaton & Quigley [1996] gained an improvement in retrieval performance through the use of a conceptual distance measure based on WordNet 1.4. One important aspect of this work though, was that both image captions and user queries were manually disambiguated with respect to WordNet senses.

The study reported in [Gonzalo *et al* 1998] set out to determine whether retrieval based on properly disambiguated WordNet synonym sets, *'synsets'*, would perform better than a standard system using word forms. The authors concluded that WordNet synsets provided a 29% increase in retrieval performance against the baseline when queries and documents are manually disambiguated. The results of this study are not, however, presented in a manner which would allow us to speculate as to the likelihood that this improvement would be observed in other test environments.

The use of a lexical/conceptual resource such as WordNet acquires a whole new dimension however, when one moves from single language retrieval applications, in which the above reported experiments were conducted, to an environment where multiple languages are involved. Research has indicated, for example, that cross-language retrieval performance is much more sensitive to ambiguity issues than monolingual retrieval [Ballesteros & Croft 1998]. In particular, the conceptual nature of WordNet synsets allows one, in a multilingual environment, to achieve cross-language matching at the conceptual rather than at the lexical level. Further, when synset labels or ids are used as the indexing vocabulary, the indexing language can be considered an *interlingua*; assuming the equivalent terms of each language are mapped to the corresponding synsets, one can match all combinations of languages, without relying on language-to-language translations in all directions between all possible language pairs.

The CINDOR project represents an attempt to build an information retrieval system specifically aimed at providing a language-independent model of retrieval at the conceptual level. The conceptual interlingua consists of the WordNet hierarchy of synset labels. Each synset label (concept) is linked in turn to a set of words or phrases which instantiate that concept in each of the languages supported. For example, the concept of *"elasticity:* the tendency of a body to return to its original shape after it has been stretched or compressed", which has the label *131186,* is instantiated in English and French as follows:

131186 spring, give, springiness
131186 elasticite, flexibilite, moelleux

This means that any document or query term which is identified as an instantiation of the concept of *'elasticity'* is indexed to the concept label *131186*. Whether the term occurs in an English, French, or Spanish document or query, the label will be the same and retrieval will be enabled.

On the other hand, the issues inherent in the use of WordNet which were identified in experiments in a single language, are if anything exacerbated when multiple languages are involved. Word sense ambiguity remains. For example, the term *'spring'* is indexed with the multiple labels:

spring 313842, 109405, 127451, 131186, 154459, 154460...

Some of the ambiguity can be resolved with fairly straightforward approaches, such as the use of part-of-speech analysis to eliminate verb senses when *'spring'* occurs in a role which is obviously a noun. In addition to the within-language ambiguities however, there is an additional layer of across-language ambiguity introduced, not just with respect to translation, but even with words which are spelled identically in different languages but which relate to different concepts (*'false cognates'*). We have discovered in TREC-7 for example, that *'Concorde'* in English should be related to supersonic jet, but in French belongs primarily to the concept of unity. Our participation in the TREC-7 evaluation has provided us with the first set of test data to use in teasing apart the issues of ambiguity, cross-language matching, concept identification, translation etc. which are inherent in our use of a conceptual interlingua for retrieval.

3. CINDOR Retrieval.

Although the CINDOR system is designed to be particularly effective in performing cross-language retrieval based on matching at the conceptual level, we are conscious of the fact that we must pay careful attention not only to performance based on cross-language versus equivalent monolingual queries, but also on the *absolute* performance achieved. This means that, while a great deal of the focus of our work is on research ensuring concept-level matching across languages, we must be acutely aware of the need to also solve within language issues of concept-based retrieval and word sense ambiguity in order to ensure an acceptable level of baseline performance. In this respect we are fortunate to be able to build upon a substantial body of work in cross-language retrieval which has already identified many of the issues involved [Oard & Diekema 1998].

The two issues of cross-language retrieval which we take to be the most important, apart of course from the obvious necessity for translation coverage, are ambiguity [Ballesteros 1998] and the treatment of phrases [Hull & Grefenstette 1996]. We believe the issue of ambiguity is particularly sensitive in our approach since the many different word senses are further expanded with lists of synonyms for each sense.

We explicitly tested the necessity for word sense disambiguation in the CINDOR system by running a test evaluation in which all document and query terms were indexed by all of the applicable concept labels and a straightforward vector space retrieval algorithm was used in matching. Apart from the likelihood of retrieving documents which match an incorrect sense of query terms, what transpired from this experiment was that the highest ranked documents were those that had the most occurrences of the query term with the most different senses! For example, the TREC-6 query *'automobile air pollution'*, retrieved documents about the *'Air Force'* in the highest rank positions. This is explained by the fact that the term *air* belongs to 36 different synsets in WordNet, while *automobile* and *pollution* each have only 3 different possible senses. When document and query indexing is based on synset labels without any disambiguation effort, the effect is to multiply the weight of each term by its degree of ambiguity so the most ambiguous terms receive the highest weighting. Though this effect could probably have been neutralized directly by a sense-based weight adjustment, we felt that the more appropriate approach lay in the direction of tackling the word sense disambiguation problem.

While a definite goal of our ongoing research, the CINDOR system does not currently include specific processing for word sense disambiguation. Seeing the need to address the sense problem described above, while cognizant of the fact that 'incorrect sense resolution has a much more deleterious effect on retrieval performance than does making spurious matches' [Voorhees 1993], we have therefore aimed for the middle ground. We retain all possible term senses in both document and query indexing, but we ensure that only one sense matches and contributes to the score for a retrieved document. Our queries are therefore evaluated as if they had a form similar to:

[automobile1 *OR* automobile2 *OR* automobile3] +
[air1 *OR* air2 *OR* air3 *OR* air4 *OR* ... *OR* air36] +
[pollution1 *OR* pollution2 *OR* pollution3] .

The second important issue of cross-language retrieval is in fact tackled in a similar way. The CINDOR system does not yet include the capability to recognize and process multi-word terms, though again this is a feature that is foreseen in the near future, particularly with respect to multi-word proper nouns. So although the Princeton WordNet contains many entries composed of multi-word entities, these are currently not directly utilized in CINDOR. Instead, indexing is as above, with each single document and query term being assigned a concept label for each synset to which it belongs, and phrase matches are then *scored* at retrieval time as if queries had a form similar to:

[automobile1 *OR* automobile2 *OR* automobile3] *AND*
[air1 *OR* air2 *OR* air3 *OR* air4 *OR* ... *OR* air36] *AND*
[pollution1 *OR* pollution2 *OR* pollution3] .

Other aspects of CINDOR retrieval are related to our experience in matching documents and queries in multiple languages using the conceptual interlingua, though the solutions again benefit from the experience of past research:

Document terms which are not found to instantiate any concepts in our current conceptual interlingua, essentially out-of-vocabulary words, are added to a table as a new concept with a corresponding new concept label. Since a large portion of these terms will relate to proper nouns not in WordNet, this table is language-independent. This has the effect that identical cognates across languages will match, even though not explicitly present in our conceptual interlingua.

Although we recognize and tag the language of documents, the conceptual interlingua is not consulted on a language-dependent basis. Concept labels are assigned to all document and query terms from all language terms. For example, whether the term *'concorde'* occurs in a French document or an English query, or anywhere else, it will be assigned the concept labels for the English concept related to jets, the French concept of unity, and any others in which the word is found. This ensures the broadest possible set of matches.

CINDOR applies morphological analysis to identify term stems in each language, and performs part-of-speech tagging to distinguish between the four categories used in WordNet; noun, verb, adjective and adverb. Retrieval is based on the Vector Space Model with retrieved documents ranked according to relevance score.

While endeavoring to ensure that we covered the issues of ambiguity, phrases, out-of-vocabulary words, and other cross-language problems while also processing the English and French document and queries by the TREC-7 deadline, it seems, as our analysis of the evaluation results will show, that we overlooked some of the more basic issues of this exercise, including the normalization of terms for case and the normalization of accented characters. While both of these processes are trivial on their own, we have seen that they can be of immense importance in ensuring that query terms and document terms are assigned the right concept labels for matching.

4. Analysis of TREC-7 Results.

Our participation in the TREC-7 Cross-Language Retrieval track was limited to the English and French document subcollections, which are as follows:

English	Associated Press	1988-1990	242,918 docs (760MB)
French	Schweizerischen Depeschen Agentur (SDA)	1988-1990	141,656 docs (250MB)

CINDOR indexing used WordNet version 1.5 augmented by semi-automatically assigned translations of the English terminology in French. The French data was initially derived from a large proprietary multilingual concept lexicon, which was then manually controlled for quality and further augmented by manual additions. Manual additions were independent of TREC queries but were driven by a frequency-sorted list of terms from the relevant TREC document collections, in order to strive for the greatest coverage of term occurrences. Our manual efforts were also focused on nouns over other parts of speech. Analysis of conceptual interlingua coverage shows that slightly less than 20% of the English WordNet has been translated into French so far, while our analysis shows that 70% of non-stopword term occurrences in a French corpus will be present in our conceptual interlingua.

	WordNet 1.5	French	% Coverage
Terms	168,135	72,473	
Synsets	91,591	16,340	17.8%

Four official runs were submitted to NIST for evaluation, all using only the topics' *'Title'* field. Given the fact that the central objective of the evaluation from our point of view was not an exact match with the track requirements (we did not cover all four languages), only one of our runs was in fact evaluated officially by NIST:

Run	Query Lang.	Doc. Lang	
TW1E2EF	English	English + French	*evaluated!*
TW2F2EF	French	English+French	
TW3E2F	English	French	
TW4F2E	French	English	

Our analysis of the official evaluation of run *TW1E2EF* indicates that it did not perform well compared to other groups who submitted equivalent cross-language retrieval runs with this combination of English queries and both English and French documents. Having analyzed the results on a query-by-query basis however, we have discovered that the formula used in merging results from different document subcollections, was inconsistent with the formula used in computing CINDOR retrieval scores to take account of word sense selection and phrase matching. Merging was used in this run because the English and French document collections were indexed separately, so the run was compiled by a combination of results from an English-English run and an English-French run. Fixing the inconsistency in merging results increased average precision for this combination from 0.1570 to 0.1967, an increase of 25%, as shown in Figure 1 below.

The officially evaluated English against English and French run also suffered further from the simple implementation issues referred to at the end of Section 3. We believe it will be clearer however if we focus on the single language pair runs between English and French to illustrate the affect of these errors on the performance of our system, rather than trying to disentangle the issues related to individual languages and issues of merging individual runs.

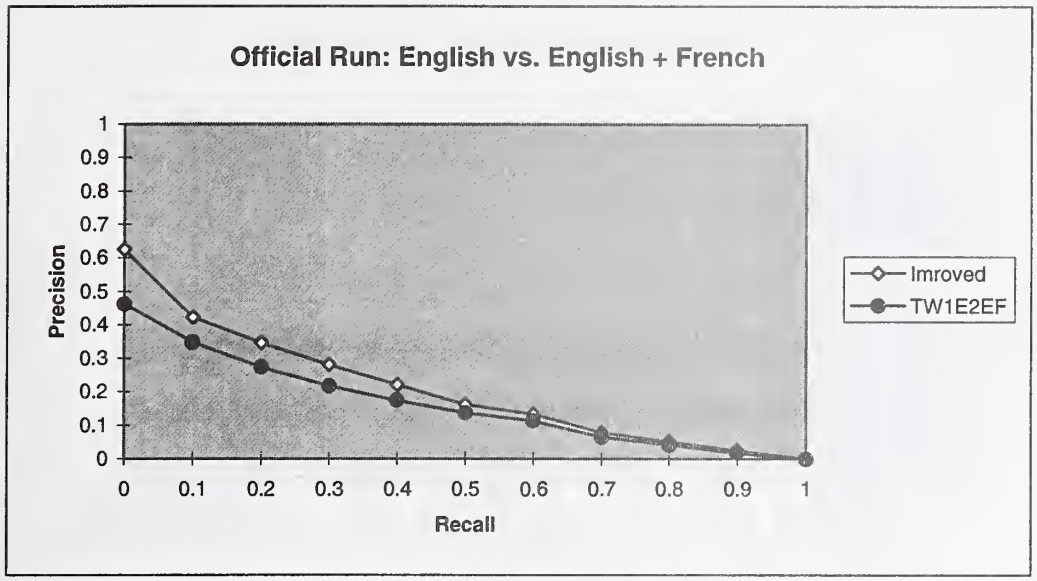


Figure 1
Official run, plus improved merging run

5. Analysis of English and French runs.

Apart from the English against French and French against English runs which we officially submitted to NIST, we have also run monolingual English and monolingual French runs to act as a baseline for our cross-language evaluation.

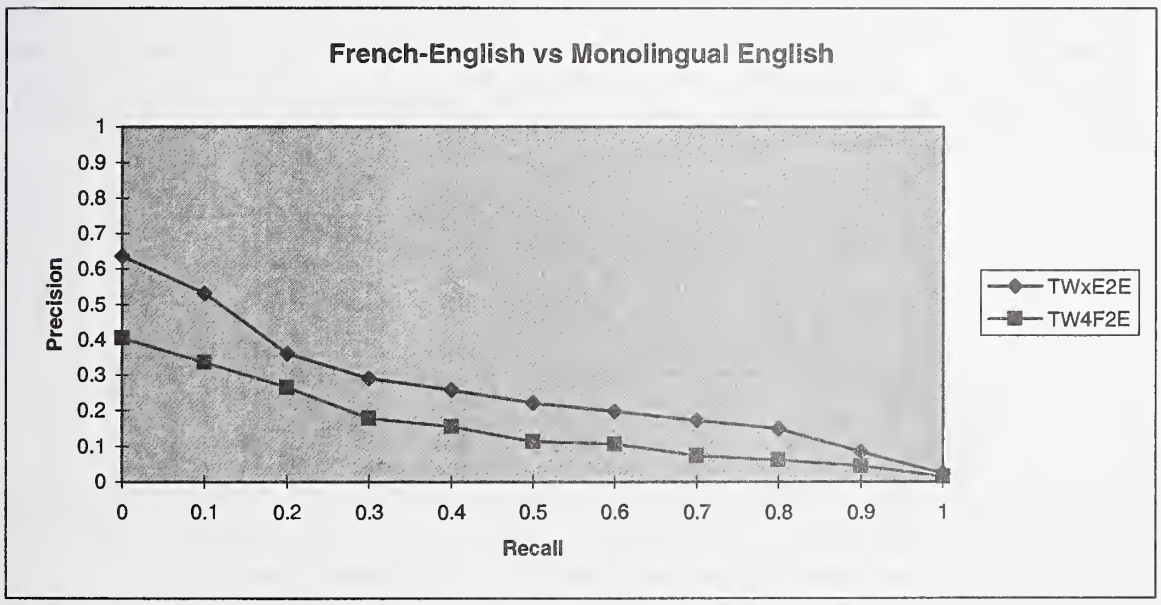


Figure 2.
Comparing French-English Cross-Language to English-only

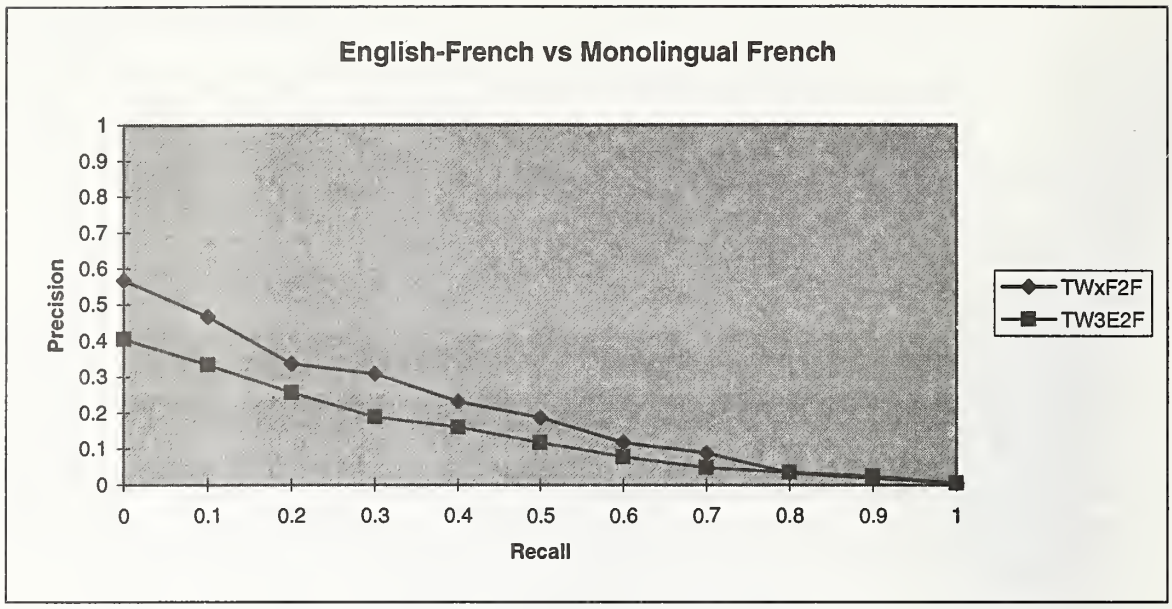


Figure 3.
Comparing English-French Cross-Language to French-only

Runs *TW3E2F* and *TW4F2E*, as presented in Figures 2 and 3, were completed before the official TREC deadline and submitted as official runs to NIST. The monolingual runs, *TWxE2E* and *TWxF2F*, were not submitted because of a limit of four submissions to the cross-language track.

While it is clear from the precision-recall graphs of Figures 3 and 4 that CINDOR cross-language retrieval in each direction is less effective than the equivalent monolingual queries (see Table 1 below), it is instructive to examine performance on a query-by-query basis in order to determine exactly the reasons for this behavior. At an aggregate view, French-English retrieval performs at 58% of monolingual English on the measure of average precision, while English-French achieves 69% of the average precision score of monolingual French.

Query \ Doc	English	French
English	0.2511	0.1377
French	0.1461	0.1992

Table 1
Average Precision: Cross-Language versus Monolingual

A query-by-query analysis of CINDOR performance is facilitated by Figures 5 and 6 below, which show side-by-side analysis of average precision for cross-language versus monolingual retrieval for each of the 28 test topics over both English (Figure 5) and French (Figure 6) document collections.

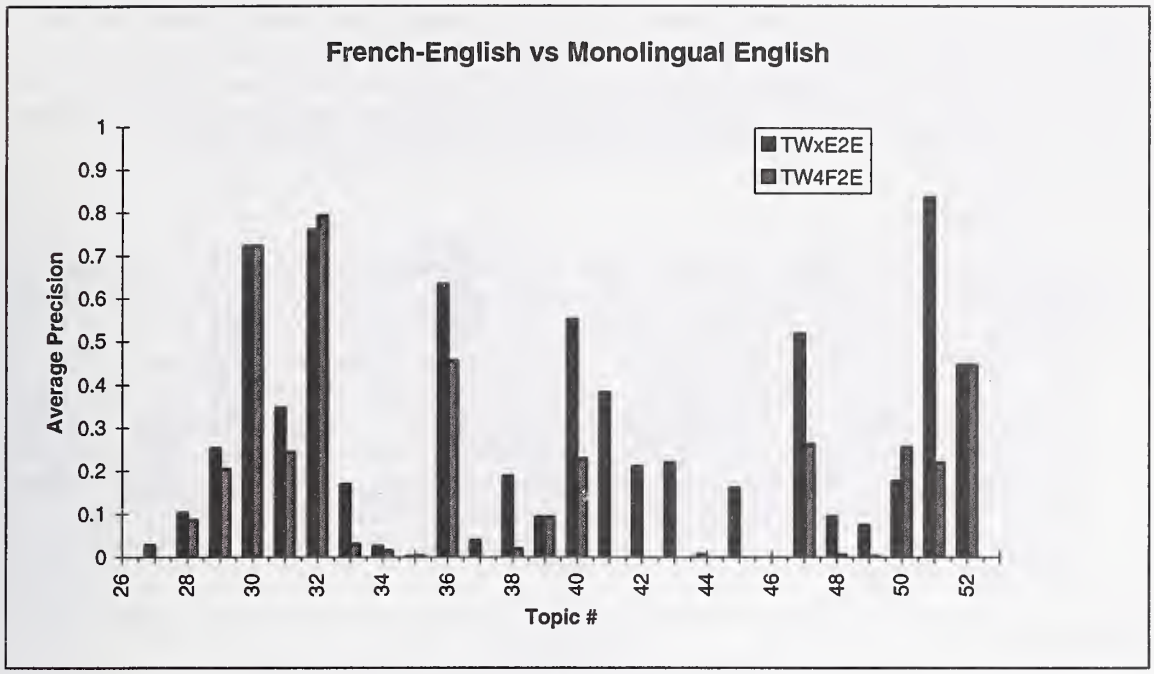


Figure 5
Query-by-query analysis against English documents

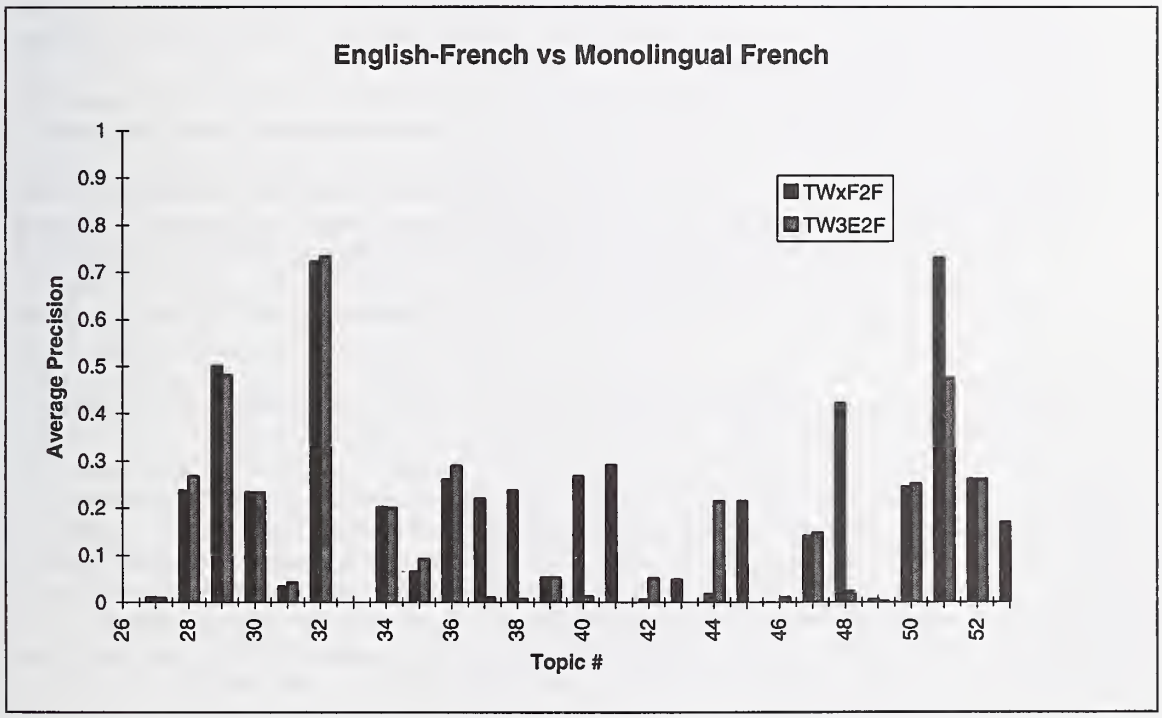


Figure 6.
Query-by-query analysis against French documents

While the initial query-by-query views presented in Figures 5 and 6 indicate specific queries over which the CINDOR system has performed poorly, a detailed analysis of individual query performance serves to illustrate certain classes of problem which have negatively impacted our overall performance on precision and recall. Many of these problems are simple items which one would, but should not, assume would be accounted for as a matter of course in building a cross-language retrieval system:

Implementation issues:

accents: The normalization of accented words was not included as part of CINDOR processing. This meant that words which are accented in one language but not in another, did not match. Words like *'Lótschberg'*, which is not in our current conceptual interlingua but would have matched with *'Lotschberg'* or *'Lötschberg'* had we normalized the accents, were lost.
caps: Although the normalization of capitalized words was included as a step in query processing, this step was mistakenly omitted in processing the French conceptual interlingua resources. Query terms such as *'Pologne'* and *'Allemagne'* were therefore reduced to *'pologne'* and *'allemagne'* during query processing, but did not receive the correct concept label because the conceptual interlingua contained the terms with their uppercase initial character intact.
hyphen: Our lexical analysis module did not segment hyphenated words so *'Franco-Allemande'* was treated as a unit.

General issues:

gap / pnoun: Although listed as two classes in the table above, both refer to particular gaps in our conceptual interlingua coverage. This is not an error, but rather is the nature of our approach (indeed every approach) that there will always be gaps in the resources used.
stemming: When the term *'kidnapping'* was analyzed from a French query using French analysis, it was returned in the form *'kidnapping'*. English analysis would however have given *'kidnap'*, so the match was missed.
phrase: In several cases, such as *'tremblement de terre'* and *'swiss confederation'*, phrases were actually present in our conceptual interlingua, but were not recognized as phrases by our query processing.
query: In some queries, particularly English topic 53, the topic title underspecifies the information need. This is indeed normal in many TREC topic descriptions, but here there is an inconsistency across languages. French topic 53 specifies that *'Europe'* is of interest, while the English version does not. A more general example of underspecification is query 33, *'genetic engineering'*, which the description elaborates to genetic engineering only in the agricultural field.

On the other hand, when the queries with these specific errors are taken out of consideration, the picture becomes altogether more positive. Ideally we would like to be able to fix these problems, re-run our tests and report the improved results directly. In the absence of the time necessary to do this however, we are limited to examining the query sub-set which has escaped unaffected by the issues outlined above. Given this limitation, we believe a fair picture of CINDOR's potential strength is presented in Figure 7 below, which presents this query subset in the direction of English-to-French versus French monolingual. This figure illustrates that CINDOR achieves cross-language retrieval performance equivalent to the monolingual level, without the loss of precision normally expected when querying across languages. This is where we believe, based on our notion of a conceptual interlingua, that CINDOR's particular strength lies. It is encouraging that this data supports our belief, though we acknowledge that the full query set must be re-run for verification.

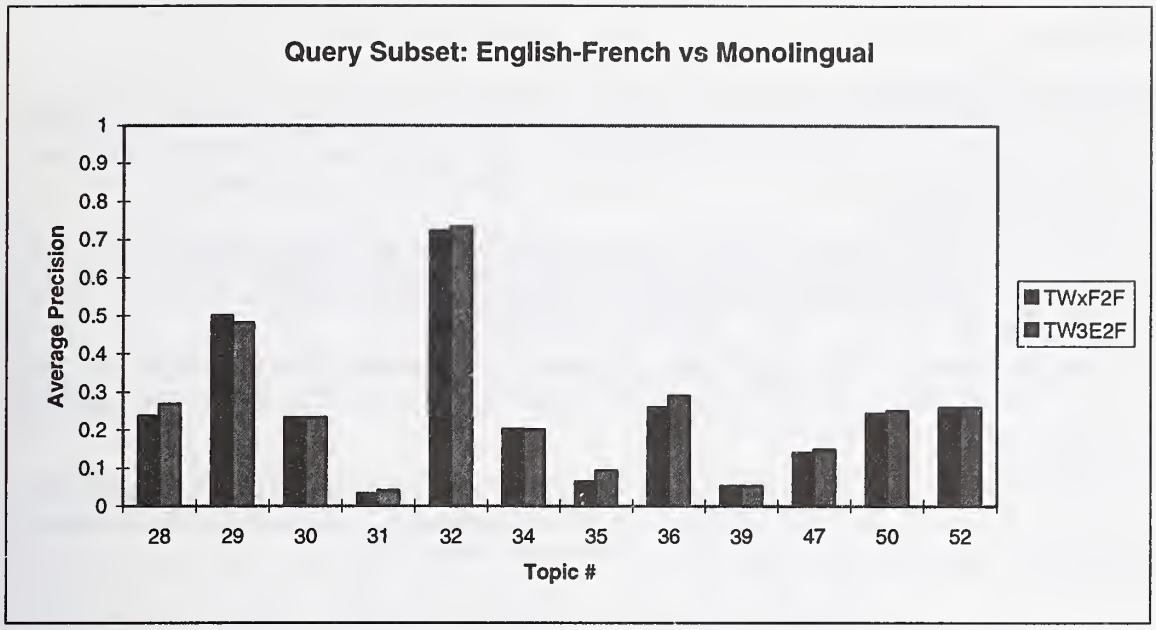


Figure 7.
Performance over queries unaffected by specific errors.

6. Conclusion.

Our query-by-query analysis of CINDOR's cross-language performance in both directions between English and French, outside of specific implementation errors, shows a remarkable number of queries for which there is either no loss of precision, or even a slight gain, when comparing cross-language performance against equivalent monolingual queries. When we examine these queries with respect to both the precision and recall achieved we see that in these cases CINDOR retrieves very often the same number of relevant documents with the same level of precision. Though we have not yet checked if exactly the same actual documents are being retrieved in these cases, it would seem to be that this is so. In several queries we also see a slight re-shuffling of the rank positions of documents returned from cross-language searches, that slightly increases the overall precision of the results.

It is therefore all the more unfortunate that our overall performance was so negatively affected by such simple implementation issues as non-matching capitalized versus non-capitalized words and accented versus unaccented characters. Such is the nature of the task of building a new retrieval system and evaluating it in the TREC environment, all under time pressure, that these kinds of mistakes are often made.

While we would like to present to the TREC community the conclusion that our conceptual interlingua approach to cross-language retrieval is obviously effective, we believe the data, clouded by our errors as it is, only suggests that this is the case. The fact that only 15 of 28 queries were left unaffected by errors (in the direction English-to-French) does not leave enough data for any solid conclusions. It is the fact that on 12 of those 15 queries we have achieved cross-language retrieval performance which closely matches the equivalent monolingual runs, which we believe is more indicative of the true potential of our approach.

While we have examined here the relative performance of cross-language retrieval compared to equivalent monolingual experiments, we have refrained from commenting on the absolute levels of performance achieved. We believe, in the absence of information about the performance of other systems which have submitted runs based on this test collection, we do not have sufficient data to comment competently on this aspect of our results. We do see however, that many of the implementation issues addressed in the previous sections will have negatively impacted our absolute level of performance compared to other groups, and this must be borne in mind when the comparison is finally presented.

References:

[Ballesteros & Croft:98]
L. Ballesteros and W. B. Croft, "Resolving Ambiguity for Cross-Language Retrieval", Proceedings of the 21st Annual Conference on Research and Development in Information Retrieval, SIGIR'98, Melbourne, Australia, 1998.

[Ballesteros & Croft:97]
L. Ballesteros and W. B. Croft, "Phrasal Translation and Query Expansion Techniques for Cross-Language Information Retrieval", Proceedings of the 20th Annual Conference on Research and Development in Information Retrieval, SIGIR'97, Philadelphia, PA, pp84-91, 1997.

[Gonzalo *et al*:98]
J. Gonzalo, F. Verdejo, I. Chugur, and J Cigarrán, "Indexing with WordNet Synsets can Improve Text Retrieval", Proceedings of the COLING/ACL Workshop on Using WordNet for Natural Language Processing, Montreal, Canada, 1998.

[Hull & Grefenstette:96]
D. Hull and G. Grefenstette, "Querying across Languages: A Dictionary-based Approach to Multilingual Information Retrieval", In proceedings of the 19th International Conference on Research and Development in Information Retrieval, SIGIR'96, Zurich, Switzerland, 1996.

[Miller:90]
G. Miller, "Wordnet: An On-line Lexical Database", International Journal of Lexicography, special issue, Vol. 3, No. 4, 1990.

[Oard & Diekema:98]
D. W. Oard and A. R. Diekema, "Cross-Language Information Retrieval", To appear in: Annual Review of Information Science and Technology (ARIST), Vol. 33, Martha Williams (Ed.), Information Today Inc., Medford, New Jersey, 1998.

[Richardson & Smeaton 1995]
R. Richardson and A. F. Smeaton, "Using WordNet in a Knowledge-Based Approach to Information Retrieval", In proceedings of the BCD-IRSG Colloquium, Crewe, 1995.

[Sanderson:97]
M. Sanderson, "Word Sense Disambiguation and Information Retrieval", PhD Thesis, Dept. of Computer Science, University of Glasgow, Scotland, 1997.

[Smeaton & Quiqley:96]
A. F. Smeaton and A. Quigley, "Experiments on Using Semantic Distances between Words in Image Caption Retrieval", In proceedings of the 19th International Conference on Research and Development in Information Retrieval, SIGIR'96, Zurich, Switzerland, 1996.

[Voorhees:93]
E. Voorhees, "Using WordNet to Disambiguate Word Senses for Text Retrieval", Proceedings of the 16th Annual Conference on Research and Development in Information Retrieval, SIGIR'93, Pittsburgh, PA, 1993.

[Voorhees:93]
E. Voorhees, "Vector Expansion in a Large Collection", In D. K. Harman, editor, Proceedings of the First Text Retrieval Conference (TREC-1), pages 343-351. NIST Special Publication 500-207, March 1993.

[Voorhees:94]
E. Voorhees, "On Expanding Query Vectors with Lexically Related Words", In D. K. Harman, editor, Proceedings of the Second Text Retrieval Conference (TREC-2), 1994.

Acknowledgments:
The authors wish to acknowledge the important contributions made to many aspects of this work by our colleagues, including; Christophe Amice, Kostya Bogatyrev, Dave Gallaro, Delphine Khanna, Severine Meyer, Woojin Paik, Patricia Patché, Christian Sarmoria, and Edmund Yu.

RETRIEVAL OF BROADCAST NEWS DOCUMENTS WITH THE THISL SYSTEM

Dave Abberley (1), Steve Renals (1) Gary Cook (2) and Tony Robinson (2,3)

(1) Department of Computer Science, University of Sheffield, UK
(2) Department of Engineering, University of Cambridge, UK
(3) SoftSound, UK
Email: {s.renals,d.abberley}@dcs.shef.ac.uk , {ajr,gdc}@eng.cam.ac.uk

ABSTRACT

This paper describes the THISL system that participated in the TREC-7 evaluation, Spoken Document Retrieval (SDR) Track, and presents the results obtained, together with some analysis. The THISL system is based on the ABBOT speech recognition system and the thislIR text retrieval system. In this evaluation we were concerned with investigating the suitability for SDR of a recognizer running at less than ten times realtime, the use of multiple transcriptions and word graphs, the effect of simple query expansion algorithms and the effect of varying standard IR parameters.

1. INTRODUCTION

THISL is an ESPRIT Long Term Research project that is investigating the development of a news-on-demand system using speech recognition, natural language processing and text retrieval. The main goal of the project is to develop a system, directed mainly toward UK English speech, for a BBC newsroom application; the TREC/SDR evaluation gives us a good opportunity to evaluate our current system on a closely related task.

The THISL spoken document retrieval system is based on the ABBOT large vocabulary continuous speech recognizer [1] and a probabilistic ranked text retrieval system. The large vocabulary speech recognizer is used to transcribe the broadcast audio, thus transforming the problem into one of text retrieval.

In this evaluation we were concerned with the following questions:

- Is a recognizer running in substantially less than ten times real-time suitable for spoken document retrieval?
- Can the use of multiple transcriptions or word graphs of documents be used to increase robustness and decrease the effect of recognition errors?
- Can query expansion be used to improve recall and precision?
- What is the effect of differing stop lists and application of stemming?

The system we used in this year's evaluation differs somewhat from the system we used in the TREC-6 SDR track [2]. In particular, we have replaced the PRISE text retrieval system with a locally implemented probabilistic system. We no longer use wordspotting to deal with out-of-vocabulary terms in queries, since experience has indicated that this is not a serious problem with speech recognizers that use a vocabulary of around 65,000 words (in this evaluation it turned out that there were three query words that were out-of-vocabulary with respect to our recognizer).

2. SPEECH RECOGNITION

2.1. ABBOT

ABBOT is a hybrid connectionist/HMM system [3] that differs from traditional HMMs in that the posterior probability of each phone given the acoustic data is directly estimated at each frame, rather than the likelihood of a phone (or state) model generating the data. Posterior probability estimation is performed by a connectionist network (or set of networks) trained to classify phones. In the ABBOT system, a recurrent network [4] is used. Direct estimation of the posterior probability distribution using a connectionist network is attractive since fewer parameters are required for the connectionist model (the posterior distribution is typically less complex than the likelihood) and connectionist architectures make very few assumptions on the form of the distribution. Additionally, this approach enables the use of an efficient search algorithm that uses a posterior probability-based pruning [5] and is able to provide useful acoustic confidence measures [6].

The speech recognition system used by the THISL group in the TREC-7 SDR track was a version of that used by the

This work was supported by ESPRIT Long Term Research Projects THISL (23495) and SPRACH (20077).

related CU-CON group in the 1997 ARPA CSR Hub 4 evaluation [7].

2.2. TRAINING

2.2.1. ACOUSTIC MODEL TRAINING

The acoustic model used in the THISL system consisted of two recurrent networks with 53 context-independent phone classes (plus silence). One network estimated the phone posterior probability distribution for each frame given a sequence of 12th order perceptual linear prediction features [8]. The other network performed the same distribution estimation with features presented in reverse order (since recurrent networks are time-asymmetric) and the two probability estimates were averaged in the log domain. Each network contained 384 state units, resulting in a total of about 350 000 acoustic model parameters, trained on the SDR acoustic training data. About 76 hours of the 100 hours of SDR acoustic training data is transcribed. After computing the average log likelihood per frame during a Viterbi alignment, a further 16 hours of this data was discarded as being below an empirically chosen log likelihood threshold, resulting in a transcribed set of acoustic training data of about 60 hours duration.

The final system used 697 context-dependent phone models, the acoustic context classes being arrived at via a decision tree algorithm. A context class network was used for each context-independent phone class, which (when combined with the context-independent phone probabilities) produced a context-dependent phone probability [9].

2.2.2. LANGUAGE MODEL TRAINING

A backed-off trigram language model was estimated from the following text sources:

- 1997 Hub4 LM text data (broadcast news transcriptions to 1996) (132M words);
- 1995 Hub4 non-financial newswire texts (108M words);
- 1995 Hub3 financial newswire texts (45M words);
- The transcripts of the SDR acoustic training data (0.8M words);
- 1995 Marketplace acoustic transcripts (0.05M words).

The 65,532 word vocabulary used all the words from the transcription of the acoustic training data, plus the most frequent remaining words extracted from the broadcast news text corpus (ignoring common misspellings and obvious text processing errors). The resultant language model contained 7.1 million bigrams and 24.0 million trigrams. We did not use the more recent SDR LM data for language modelling, although some of this data was used for query expansion (section 7).

2.3. RESULTS

Using the noway start-synchronous decoder [5] the system ran in about seven times real time on an Ultra-1/167MHz (512-1024 Mb RAM), with the computation split approximately equally between the recurrent network-based acoustic model and the LVCSR search algorithm. Running at this speed required more pruning than would be employed in a CSR evaluation, and the estimated relative search error resulting from incorrect pruning of the search space was 10–20%.

The overall average word error rate (WER) of the THISL speech recognition system in this evaluation was 35.9%. We can also use an error metric conditioned on the text retrieval system, the *term error rate* (TER) [10], which is given by the following formula:

$$TER = \frac{\sum_{t \in T} | R(t) - H(t) |}{|T|} \times 100\% \qquad (1)$$

where $R(t)$ and $H(t)$ represent the number of occurrences of *term* t in the reference and hypothesised transcripts respectively. The set of terms T is calculated after the transcripts have been stopped and stemmed but without taking account of term order. Thus TER gives a more accurate measure than WER of the erroneous terms which will be processed during IR. Additionally, calculating WER is meaningless for merged transcripts (section 5), but TER still provides some information about transcript quality. In conjunction with our submitted system, using a 379 word stop list and Porter stemming the THISL speech recognition system returned a TER of 52.2%.

3. TEXT RETRIEVAL

In last year's SDR evaluation we used the PRISE text retrieval system developed by NIST. This year we used a locally implemented system. This was essentially a "textbook TREC system", using a stop list, the Porter stemming algorithm and the Okapi term weighting function. Specifically we used the term weighting function $CW(t,d)$ for a term t and a document d given in [11]:

$$CW(t,d) = \frac{CFW(t) * TF(t,d) * (K+1)}{K((1-b) + b * NDL(d)) + TF(t,d)}. \qquad (2)$$

$TF(t,d)$ is the frequency of term t in document d, $NDL(d)$ is the normalized document length of d:

$$NDL(d) = \frac{DL(d)}{\overline{DL}}, \qquad (3)$$

where $DL(d)$ is the length of document d (ie the number of unstopped terms in d). $CFW(t)$ is the collection frequency weight of term t and is defined as:

$$CFW(t) = \log\left(\frac{N}{N(t)}\right) \qquad (4)$$

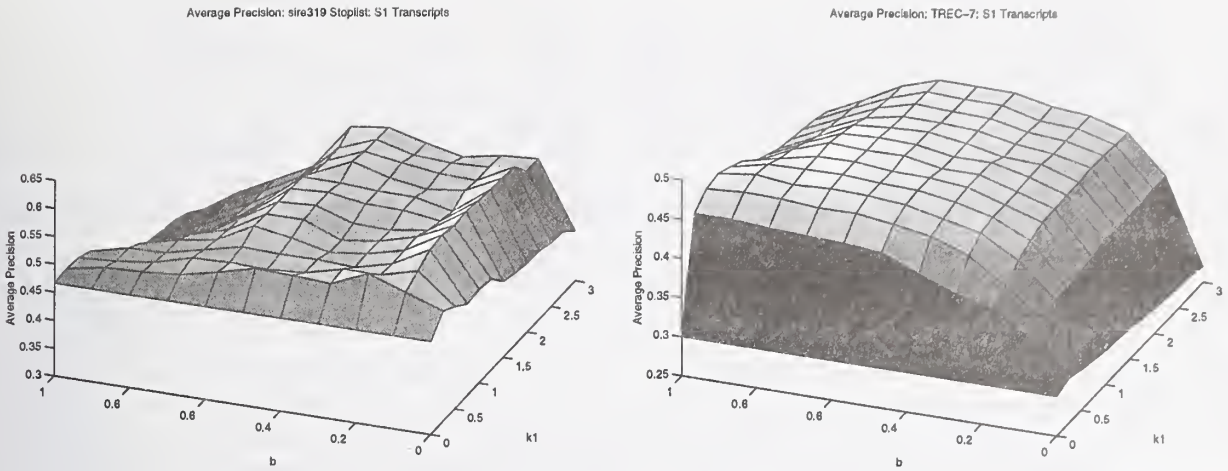


Figure 1: Plot of average precision against term weighting parameters b and K for TREC-7/SDR local development queries (left), and TREC-7/SDR evaluation queries (right).

where N is the number of documents in the collection and $N(t)$ is the number of documents containing term t. The parameters b and K in (2) control the effect of document length and term frequency as usual.

Prior to the evaluation, we conducted a variety of experiments, using a development set of 16 queries devised and judged for relevance locally. These experiments were designed to investigate:

- the effect of varying term weighting parameters, stop lists and stemming (section 4)
- the use of multiple transcriptions arising from the component networks of the ABBOT acoustic model (section 5);
- the use of word graph representations of spoken documents (section 6);
- the behaviour of query expansion (section 7).

We note that these development queries were more similar to the TREC-6/SDR queries than the TREC-7/SDR queries, having an average of 4.5 relevant documents per query (found by manual operation of PRISE). This compares with the evaluation queries which had an average of 17 relevant documents per query.

4. TEXT RETRIEVAL PARAMETERS

4.1. TERM WEIGHTING PARAMETERS

Since the document collection and queries are a little different to the TREC ad-hoc task, we decided to investigate the effect of varying the parameters b and K in the term weighting function (2). The results for the development set are shown on the left of figure 1. After the evaluation we produced a similar graph for the TREC-7 SDR evaluation queries (figure 1, right). We note that for the development queries there is a ridge of high average precision along $K = 0.25$, which corresponds to a decrease in the significance of TF compared with CFW. This is not present in the evaluation queries. There is another a maximum around $(b, K) = (0.5, 1.0)$, for both sets of queries, which (fortunately) were the parameter settings used for all our submitted runs.

The reason for the different behaviour of the two query sets is not clear. Although it may be due to the relatively small task size (around 3000 spoken documents), we also note that our local development queries had many fewer relevant documents per query compared with the evaluation queries (4.5 vs. 17). Support for the latter hypothesis is given by the fact that the parameter landscape for the known-item TREC-6/SDR queries (ie 1 relevant document per query) is most similar to the development set.

4.2. STOP LISTS

We conducted experiments using hand constructed stop lists including the 23 word stop list that is standard with PRISE, the 319 word stop list used by the University of Glasgow, the 429 word stop list in [12], and a locally developed 379 word stop list based on the Glasgow stop list with extra words added following analysis of previous TREC queries. As control experiments we used stop lists comprising the most frequent n words, and also no stop list. Results on our development set of queries are shown in figure 2, and a graph of term error rate vs. stop-list size is shown in fig-

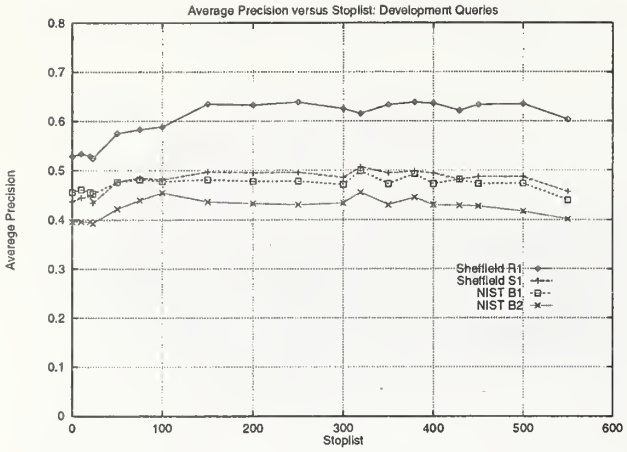


Figure 2: Effect of stop list on average precision using local TREC-7 development queries for R1, S1, B1 and B2 conditions, using Porter stemming. Stop lists of size 23, 319, 379 and 429 were hand-constructed the others comprised the most frequent n words.

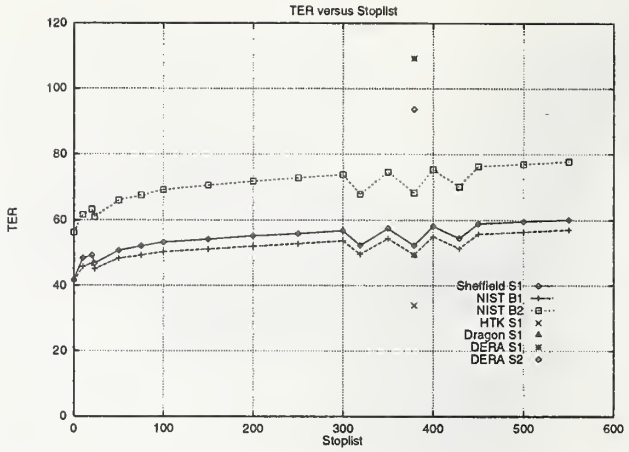


Figure 3: Effect of stop list on term error rate for S1, B1 and B2 recognizers, using Porter stemming. The hand constructed stop-lists of 319, 379 and 429 words can be clearly identified. TER for Dragon, CUHTK and DERASRU recognizers with the 379 word stop list are also shown.

ure 3. We note that hand-constructed stop lists perform a little better than the similarly sized "top-n" stop lists.

4.3. STEMMING

We also evaluated the effect of stemming by running after the evaluation with and without the Porter stemming algorithm. These results are shown in table 1.

	Average Precision	
	With	Without
System	Stemming	Stemming
R1	0.4886	0.4179
S1	0.4599	0.3774
B1	0.4355	0.3570
B2	0.3529	0.2570

Table 1: Effect of stemming (Porter algorithm) on average precision to TREC-7 SDR queries (post-evaluation experiment). Experiments used a 379 word stop list and query expansion.

5. MULTIPLE TRANSCRIPTIONS

A number of participants at the TREC-6 SDR track (eg, [13, 14]) took advantage of the availability of multiple sets of speech recognition transcriptions and merged them to produce improved information retrieval performance. This method was successful because although speech recognizers make errors, different speech recognizers are likely to make different errors. Thus if an important query word has been missed by one recognizer, another one might recognize it correctly so that it does not get omitted from the index.

As mentioned in section 2, the ABBOT acoustic model is based on multiple recurrent networks, which are averaged together at the acoustic frame level. However, it is possible to run separate decodings based on the individual recurrent networks and to merge them together at the transcription level. Experiments were run on the TREC-6 known-item retrieval task using the 379 word stop list but no query expansion. Table 2 shows the results in terms of word error rate (WER), term error rate (TER) and the various TREC-6 IR performance measures.

The table indicates that merging the RNNs at the acoustic probability level (S1) produces better WER/TER and IR performance than either of the individual networks. Despite the inevitably higher TER, merging multiple transcripts seems to produce slightly better IR results than taking their union. The detrimental effects of merging may be partially offset by term frequency weighting. In these experiments, neither merging technique produced clearly better IR performance than the single best set of transcripts (S1), except for the percentage of queries for which the answer was not found.

The results from these experiments are somewhat inconclusive: it is possible that multiple transcripts could be used to enhance retrieval performance but these benefits have yet to be demonstrated unequivocally, and must be offset against the considerable extra resources required to produce the multiple transcriptions (which is why the experiments were not repeated on TREC-7 data).

Transcripts	WER	TER	Mean Rank	Mean Reciprocal	Percentage at Rank 1	Percentage Not Found
R1	–	–	5.85	0.8509	78.7%	0.0%
S1	38.8%	55.4%	11.72	0.7776	74.5%	2.1%
Forward net	43.2%	63.3%	14.33	0.6996	61.7%	2.1%
Backward net	41.7%	61.4%	17.96	0.7091	63.8%	4.3%
Merged fwd+bwd	–	135.9%	14.51	0.7414	68.1%	0.0%
Union fwd+bwd	–	90.3%	18.45	0.7477	68.1%	0.0%
Merged S1+fwd+bwd	–	228.5%	14.40	0.7793	72.3%	0.0%
Union S1+fwd+bwd	–	95.9%	19.77	0.7434	68.1%	0.0%

Table 2: Use of multiple transcriptions derived from ABBOT on the TREC-6 known-item retrieval task. R1 are the reference transcripts, S1 are the transcripts produced by ABBOT using frame-level merging. Forward and backward are the decodings produced by the nets in isolation. The term 'merged' implies the concatenation of two or more sets of transcripts whereas the term 'union' implies the union of sets of transcripts — multiple occurrences of the same term are discarded.

6. WORD GRAPHS

As a side effect of large vocabulary decoding, it is possible to produce word graphs (lattices). A word graph consists of set of nodes, each labelled with a reference time, and a set of links. Each link connects two nodes and is labelled with a word, together with information such as the acoustic score of that word accounting for the acoustic data that covers the time span between the nodes. Each link in the word graph corresponds to a word that was hypothesised during the search that could contribute to a complete word path within the graph. Thus a word graph efficiently represents the entire valid search space considered by the speech recognition decoder. On average, the word graphs produced by ABBOT contain about twenty times as many words as in the most probable transcription.

We treated a word graph as we would a single transcription in text retrieval, representing a document as a bag of word graph links. However since word graphs tend to be bushier in regions of acoustic confusion, the contribution of link i to the corresponding term frequency was based on the reciprocal of the graph density $(1/GD_i)$ for i. The graph density GD_i of graph link i is defined as the average number of links in the graph that account for each frame covered by link i.

The term frequencies that arise from representing documents as bags of graph links are less sharp than those that arise from 1-best transcriptions, since more terms are present in the document. Two ways to sharpen the term frequencies arising from graphs are by merging with the most probable transcription or by weighting the lattice links by an acoustic score. In this paper we have only tried the former.

We ran a number of experiments using word graphs on our development queries, using the Glasgow 319 word stop list. Results indicated that best performance resulted from parameter values $b = 0.6$ and $K = 2.0$. Recall and precision curves are shown in figure 4 (left). These results did not indicate that word graphs gave improvements in recall and precision, so we did not use them in our submitted evaluation system.

After the evaluation we ran the SDR evaluation queries against indexed word lattices, using the same parameters as before. Recall and precision curves are shown in figure 4 (right). The performance of the word graph representation is substantially worse than the one-best transcriptions. Since the evaluation queries were quite different to our development queries we reran a search in (b, K) space, but no other settings of these parameters were significantly better.

7. QUERY EXPANSION

If a relevant document does not contain the terms that are in the query, then that document will not be retrieved. The aim of query expansion is to reduce this query/document mismatch by expanding the query using words or phrases with a similar meaning or some other statistical relation to the set of relevant documents. Such a process may have increased importance in spoken document retrieval, since the word mismatch problem is heightened by the presence of errors in the automatic transcription of spoken documents.

An obvious danger in using relevant documents retrieved from a database of automatically transcribed spoken documents is that the query expansion may include recognition errors. This was an experience reported by the INQUERY group in the TREC-6 SDR evaluation [15]. To avoid this problem we retrieved relevant documents from another collection of newswire text. The query expansion algorithm was then applied to the top n documents retrieved from that collection. The resulting expanded query was then applied to the collection of spoken documents.

We used an algorithm based on the local context analysis algorithm of Xu and Croft [16]. The initial query Q is

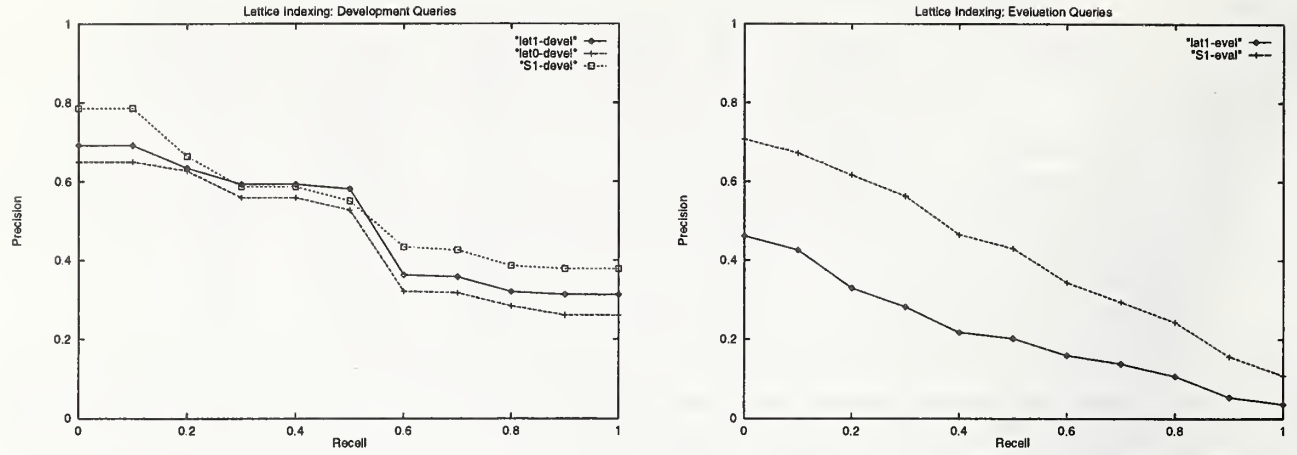


Figure 4: Recall-precision plot of SDR development queries (left) and evaluation queries (right) with documents represented as word graphs (lat0), most probable transcriptions (S1) and merged word graphs and transcriptions (lat1).

applied to the secondary query expansion collection. The nr top ranked documents are regarded as relevant; the algorithm is not discriminative so no non-relevant documents are required. A query expansion weight, $QEW(Q,e)$ is defined as follows:

$$QEW(Q,e) = \sum_{t \in Q} \log\left(\frac{\log(AF(e,t)) * CFW(e)}{\log(nr)} + \delta\right) * CFW(t). \quad (5)$$

The potential query expansion terms e are simply those terms in the relevant documents. The term $AF(e,t)$ measures the term frequency correlation of two terms e and t across collection of documents d_i:

$$AF(e,t) = \sum_{i=1}^{nr} TF(e,d_i) * TF(t,d_i). \quad (6)$$

The nt possible expansion terms with the largest weights are then added to the original query, weighted as $1/rank$.

In practice the values of nr and nt are maximum limits, since we threshold so that only those documents with a score greater than 0.8 times the score of the top-ranked document are considered, and only those terms with $QEW(Q,e)$ greater than an empirically-determined threshold are added.

In the SDR evaluation we used the June 1997–February 1998 LA Times/Washington Post portion of the SDR LM text corpus as the query expansion database. This corpus contains about 13 million words and about 22,000 documents. The parameters nr and nt are clearly dependent on the size of the query expansion collection. Experiments to investigate the dependence on these parameters were carried out on our local development queries, and the results are shown in figure 5. From this we chose parameter values $(nr,nt) = (8,10)$. Figure 6 shows the performance of

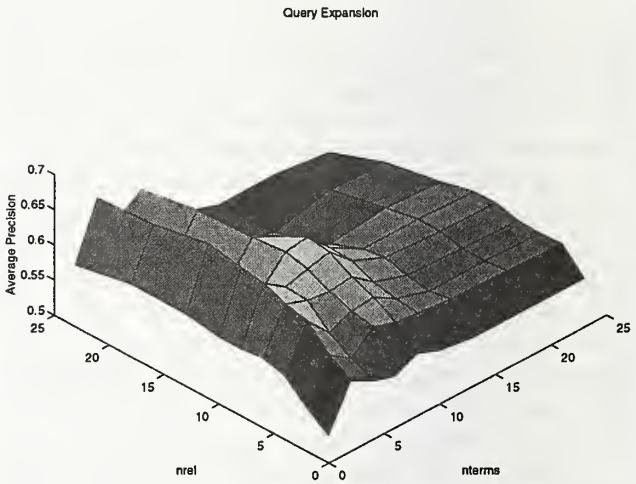


Figure 5: Effect of the query expansion parameters nr (maximum number of relevant documents to consider) and nt (maximum number of terms to add) on the average precision for S1, using 379 word stop list applied to our local development queries. The LA Times/Washington Post portion of the SDR language model corpus was used as the query expansion collection.

query expansion using a newswire corpus versus expanding on the target recognizer transcripts. Note that expanding on the recognizer transcripts is worse than no query expansion.

8. EVALUATION RESULTS

The text queries were preprocessed before being input to the system, to remove punctuation, convert to lower case and to

Condition	WER	TER	Retrieved	Relevant	Rel. Retrieved	AveP	R-P
R1	–	–	17613	390	364	0.4886	0.4583
S1	35.9%	52.2%	18312	390	360	0.4599	0.4485
B1	35.2%	49.5%	18093	390	355	0.4355	0.4562
B2	47.8%	68.3%	18671	390	354	0.3529	0.3347
CR-CUHTK	24.8%	34.0%	18105	390	365	0.4711	0.4469
CR-DERASRU-S1	66.2%	109.3%	17844	390	334	0.3780	0.4164
CR-DERASRU-S2	61.5%	93.7%	17973	390	344	0.4047	0.4016
CR-DRAGON-S1	29.8%	49.2%	18252	390	361	0.4613	0.4372

Table 4: Summary of results in different conditions. WER is word error rate, TER is term error rate (defined in section 2), AveP is the average precision and R-P is the R-precision.

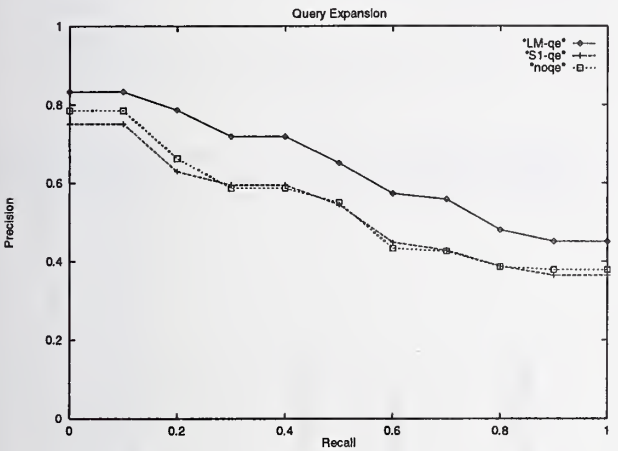


Figure 6: Effect of query expansion using newswire text (LM-qe) and recognizer transcripts (S1-qe) compared with no query expansion (noqe) on development queries.

expand abbreviation words to cover alternative transcription possibilities (eg, "AIDS" was expanded to "aids" and "a. i. d. s.").

No multiwords or phrases were used in the recognition or retrieval process. There were three OOV query words: *Montserrat*, *Trie* and *vs.* (versus). Our TREC-6 word spotting system [2] for OOV word restoration was not used in the TREC-7 system. Hopefully query expansion partially offset some of the problems caused by the OOV words.

Our submitted system used the 1-best transcriptions together with the query expansion algorithm outlined above. The 379 word stop list was used, together with the Porter stemming algorithm. The tunable parameters (set according to local development data) are given in table 3.

We ran using the following different transcripts:

R1 Reference transcripts (low WER)

B1 Medium error baseline transcripts (NIST running CMU Sphinx) (35% WER)

Parameter	Value
b	0.5
K	1.0
QE-b	0.5
QE-K	0.25
QE-nt	10
QE-nr	8

Table 3: Parameter settings for TREC/SDR submitted runs.

B2 High error baseline transcripts (NIST running CMU Sphinx) (49% WER)

S1 THISL speech recognition (36% WER)

CR-CUHTK Cambridge University (HTK) speech recognition (25% WER)

CR-DERASRU-S1 DERA/SRU speech recognition (66% WER)

CR-DERASRU-S2 DERA/SRU speech recognition (61% WER)

CR-DRAGON-S1 Dragon speech recognition (30% WER)

The results are summarized in table 4. The recall-precision curves resulting from these runs are shown in figure 7. Figure 8 shows the effect of query expansion on recall and precision for the R1 and S1 conditions. Results for the other speech recognizers are not shown to avoid cluttering the graph, but the effect of query expansion follows a similar trend for those.

Figure 9 shows the relative change due to query expansion for each of the twenty-three queries. As can be seen, query expansion resulted in an improvement or no significant change in average precision for most queries. An example of a query for which the query expansion algorithm proved effective:

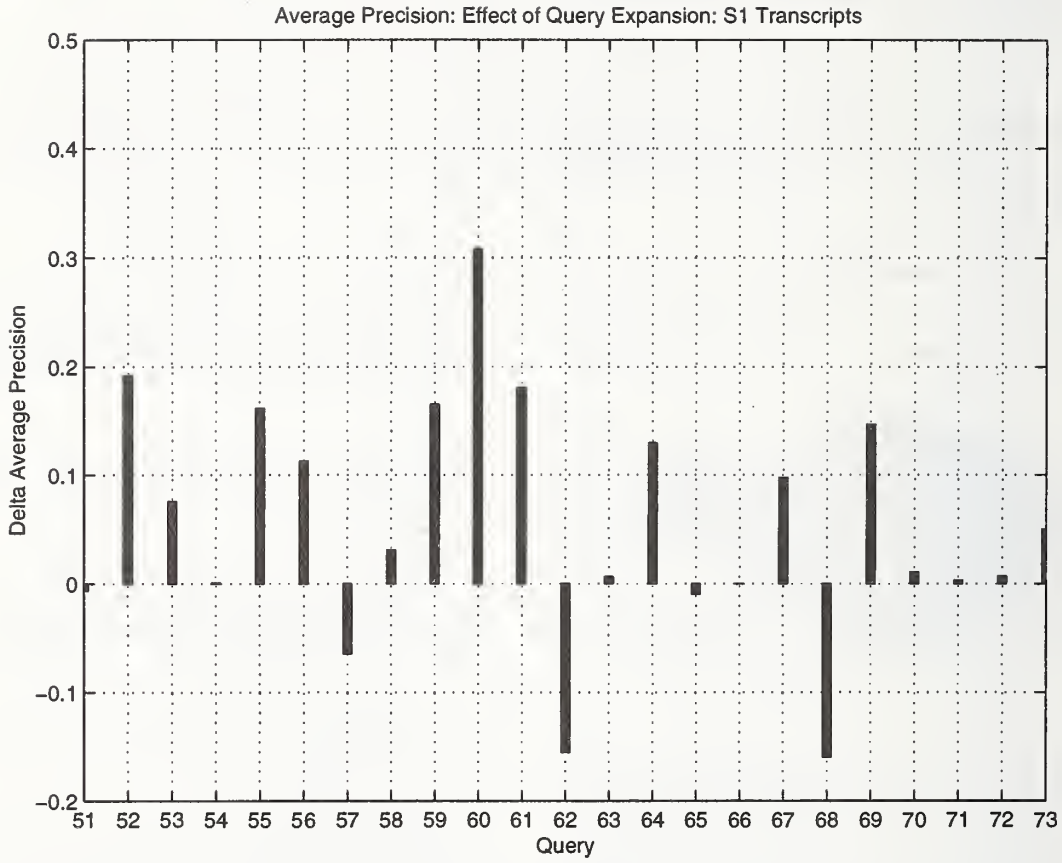


Figure 9: Query-by-query effect of query expansion in terms of change in average precision compared with no query expansion.

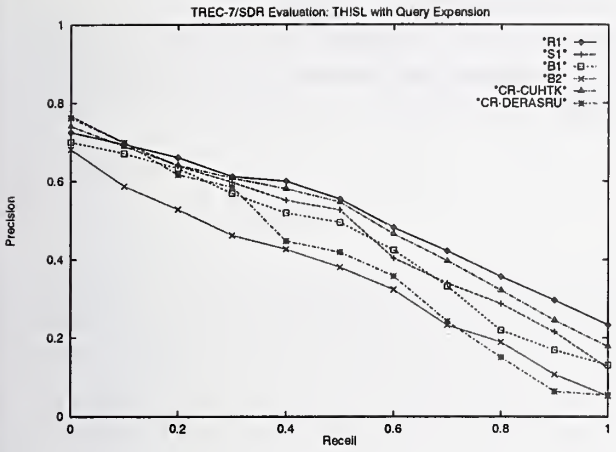


Figure 7: Recall-precision curves of the THISL system running on various transcripts submitted for TREC-7/SDR.

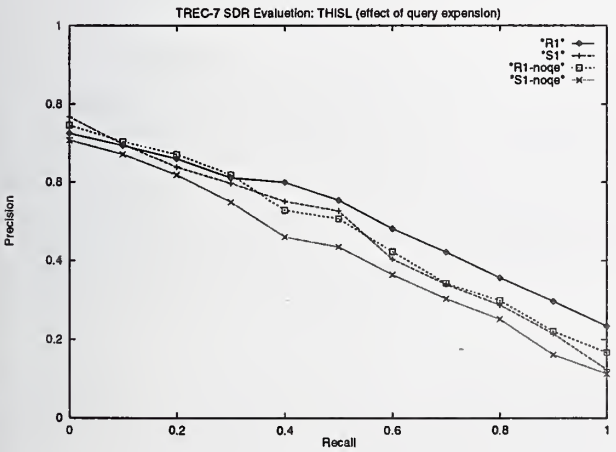


Figure 8: Effect of query expansion on recall-precision for evaluation R1 and S1 conditions (post-evaluation experiment).

60: What information is available on the activities and motivation of intrusive photographers, i.e., the so-called paparazzi?

Original Query: activ avail paparazzi photograph intrus motiv call (AveP = 0.5630)

Expansion Terms: spencer ritz gambino merced editor trespass tabloid (AveP = 0.8589)

A query for which query expansion failed was the following:

62: Find reports of fatal air crashes.

Original Query: air fatal crash (AveP = 0.3520)

Expansion Terms: auto aviat safeti vehicl occup bag jour util (AveP = 0.1893)

9. CONCLUSIONS

The major conclusions that we have drawn from these experiments are:

- Query expansion using a secondary collection derived from newswire data from a similar time period gives a consistent relative improvement in average precision of around 10%.
- Although speech recognizer word error rate does have an effect on recall and precision of the retrieval performance, there is not a clear linear relationship. It seems to be the case that varying retrieval strategy has a much greater effect than improving the recognizer.
- Our first attempts at including word graph and multiple transcription information have not resulted in improvements in recall and precision.
- Using a 100 hour audio archive, spoken document retrieval using a relatively high WER speech recognizer has around 5% lower average precision compared with the reference transcriptions.

10. REFERENCES

[1] T. Robinson, M. Hochberg, and S. Renals, "The use of recurrent networks in continuous speech recognition," in *Automatic Speech and Speaker Recognition – Advanced Topics* (C. H. Lee, K. K. Paliwal, and F. K. Soong, eds.), ch. 10, pp. 233–258, Kluwer Academic Publishers, 1996.

[2] D. Abberley, S. Renals, G. Cook, and T. Robinson, "The 1997 THISL spoken document retrieval system," in *Proc. Sixth Text Retrieval Conference (TREC–6)*, pp. 747–752, 1998.

[3] H. Bourlard and N. Morgan, *Connectionist Speech Recognition—A Hybrid Approach*. Kluwer Academic, 1994.

[4] A. J. Robinson, "The application of recurrent nets to phone probability estimation," *IEEE Trans. Neural Networks*, vol. 5, pp. 298–305, 1994.

[5] S. Renals and M. Hochberg, "Start-synchronous search for large vocabulary continuous speech recognition," *IEEE Trans. Speech and Audio Processing*, in press.

[6] G. Williams and S. Renals, "Confidence measures derived from an Acceptor HMM," in *Proc. Int. Conf. Spoken Language Processing*, 1998.

[7] G. D. Cook and A. J. Robinson, "The 1997 Abbot system for the transcription of broadcast news," in *Proceedings of the 1998 Broadcast News Transcription and Understanding Workshop*, 1998.

[8] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *J. Acoust. Soc. Amer.*, vol. 87, pp. 1738–1752, 1990.

[9] D. J. Kershaw, M. M. Hochberg, and A. J. Robinson, "Context-dependent classes in a hybrid recurrent network-HMM speech recognition system," in *Advances in Neural Information Processing Systems*, vol. 8, MIT Press, 1996.

[10] S. E. Johnson, P. Jourlin, G. L. Moore, K. Sparck Jones, and P. C. Woodland, "The Cambridge University Spoken Document Retrieval System," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 1999. to appear.

[11] S. E. Robertson and K. Sparck Jones, "Simple proven approaches to text retrieval," Tech. Rep. TR356, Cambridge University Computer Laboratory, 1997.

[12] C. Fox, "Lexical analysis and stoplists," in *Information Retrieval: Data Structures and Algorithms* (W. B. Frakes and R. Baeza-Yates, eds.), ch. 7, pp. 102–130, Prentice Hall, 1992.

[13] F. Crestani, M. Sanderson, M. Theophylactou, and M. Lalmas, "Short queries, natural language and spoken document retrieval: Experiments at Glasgow University," in *Proc. Sixth Text Retrieval Conference (TREC-6)*, pp. 667–686, 1998.

[14] A. Singal, J. Choi, D. Hindle, and F. Pereira, "AT&T at TREC-6: SDR track," in *Proc. Sixth Text Retrieval Conference (TREC-6)*, pp. 227–232, 1998.

[15] J. Allan, J. Callan, W. B. Croft, L. Ballesteros, D. Byrd, R. Swan, and J. Xu, "INQUERY does battle with TREC-6," in *Proc. Sixth Text Retrieval Conference (TREC-6)*, pp. 169–206, 1998.

[16] J. Xu and W. B. Croft, "Query expansion using local and global document analysis," in *Proc. ACM SIGIR*, 1996.

SPOKEN DOCUMENT RETRIEVAL FOR TREC-7 AT CAMBRIDGE UNIVERSITY

S.E. Johnson†, P. Jourlin‡, G.L. Moore†, K. Spärck Jones‡ & P.C. Woodland†

†Cambridge University Engineering Department,Trumpington Street, Cambridge CB2 1PZ, UK.
Email: {sej28,glm20,pcw}@eng.cam.ac.uk
‡Cambridge University Computer Laboratory, Pembroke Street, Cambridge, CB2 3QG, UK.
Email: {pj207,ksj}@cl.cam.ac.uk

ABSTRACT

This paper presents work done at Cambridge University, on the TREC-7 Spoken Document Retrieval (SDR) Track. The broadcast news audio was transcribed using a 2-pass gender-dependent HTK speech recogniser which ran at 50 times real time and gave an overall word error rate of 24.8%, the lowest in the track. The Okapi-based retrieval engine used in TREC-6 by the City/Cambridge University collaboration was supplemented by improving the stop-list, adding a bad-spelling mapper and stemmer exceptions list, adding word-pair information, integrating part-of-speech weighting on query terms and including some pre-search statistical expansion. The final system gave an average precision of 0.4817 on the reference and 0.4509 on the automatic transcription, with the R-precision being 0.4603 and 0.4330 respectively.

The paper also presents results on a new set of 60 queries with assessments for the TREC-6 test document data used for development purposes, and analyses the relationship between recognition accuracy, as defined by a pre-processed term error rate, and retrieval performance for both sets of data.

1. INTRODUCTION

Spoken Document Retrieval (SDR) combines state of the art technology from the fields of speech recognition and information retrieval. We combine the high performance HTK speech recogniser with the tried and tested Okapi-based retrieval engine to produce a good SDR system, then develop some extensions to improve the system further. We evaluated performance during development on the TREC-6 SDR document data using a set of 60 queries developed in-house (CU60), and applied our final system in the TREC-7 SDR track.

This paper firstly describes the TREC SDR task and the data used in both development and evaluation of our SDR system. The speech recogniser is described in detail in section 2, where the performance of all the sites participating in the cross-recogniser runs is given. The retrieval engine is then described in section 3 emphasising the innovations introduced for the TREC-7 evaluation and giving results based on both the CU60 development set and the TREC-7 evaluation set. A summary of these results is presented in section 4. The relationship between the output of the speech recogniser and the input of the retriever is discussed in section 5, leading to the introduction of a processed Term Error Rate (TER) to represent the recognition accuracy for SDR systems. Section 6 presents the relationship between this TER and retrieval performance for different speech recognisers and shows the degradation of retrieval performance with increased TER. Finally, conclusions are offered in section 7.

1.1. Description of TREC SDR Task

For the TREC-7 SDR track, audio from American broadcast radio and TV news programs is presented along with a list of manually-generated *document*-boundaries. Natural language text queries, such as "Have their been any volcanic eruptions in Montserrat recently?" are then provided. The participating sites must generate a transcription of the audio automatically and run an IR engine on this transcription to provide a ranked list of potentially relevant documents.

Real relevance assessments generated by humans are then used to evaluate the ranked list in terms of the standard IR measures of precision and recall. Sites may also run their retrieval system on a manually-generated reference transcription, baseline transcription(s) provided by NIST and cross-recogniser transcriptions generated by other participating sites.

1.2. Description of data

There are two main considerations when describing the data for SDR. Firstly the audio data used for transcription, and secondly the query/relevance set used during retrieval. Table 1 describes the main properties of the former, whilst Table 2 describes the latter, for the *development* and *evaluation* data sets.

	Development	Evaluation
Name of Data	TREC-6 Test	TREC-7 Test
Nominal Length of Audio	50 hours	100 hours
Number of Documents	1451	2866
Number of Different Shows	12	8
Approx. Number of Words	410,000	770,000
Average Doc length	283 words	269 words

Table 1: Description of data used

	Development	Evaluation
Name of Query Set	CU60	TREC-7 Test
Number of Queries	60	23
Average Length of Query	7.1 words	14.7 words
Number of Relevant Docs	549	390
Mean # Rel Docs per Query	9.2 docs	17.0 docs

Table 2: Description of query and relevance sets used

2. THE HTK BROADCAST NEWS TRANSCRIPTION SYSTEM

The input data is presented to our HTK transcription system as complete episodes of broadcast news shows and these are first converted to a set of segments for further processing. The segmentation uses Gaussian mixture models to divide the audio into narrow and wideband audio and also to discard parts of the audio stream that contains no speech (typically pure music). The output of a phone recogniser is used to determine the final segments which are intended to be acoustically homogeneous. Further details of the segmenter are given in [5].

Each frame of input speech to be transcribed is represented by a 39 dimensional feature vector that consists of 13 (including c_0) cepstral parameters and their first and second differentials. Cepstral mean normalisation (CMN) is applied over a segment.

Our system uses the LIMSI 1993 WSJ pronunciation dictionary augmented by pronunciations from a TTS system and hand generated corrections. Cross-word context dependent decision tree state clustered mixture Gaussian HMMs are used with a 65k word vocabulary. The full HTK system [12] operates in multiple passes and incorporates unsupervised maximum likelihood linear regression (MLLR) based adaptation and uses complex language models via lattice rescoring and quinphone HMMs. This system gave a word error rate of 16.2% in the 1997 DARPA Hub4 broadcast news evaluation.

The TREC-7 HTK SDR system uses the first two passes of the 1997 HTK Broadcast News System [12] in a modified form for reduced computational requirement. The first pass uses gender independent, bandwidth dependent cross-word triphone models with a trigram language model to produce an initial transcription. The output of the first pass is used along with a top-down covariance-based segment clustering algorithm [6] to group segments within each show to perform unsupervised test-set adaptation using maximum likelihood linear regression based model adaptation [7, 3]. A second recognition pass through the data is then performed using a bigram language model to generate word lattices using adapted gender and bandwidth specific HMMs. These bigram lattices were expanded using a 4-gram language model and the best path through these lattices gives the final output. This system runs in about 50 times real-time on a Sun Ultra2 and achieves an error rate of 17.4% on the 1997 Hub4 evaluation data. It should be noted that the error rates on Hub4 data and TREC data are not strictly comparable in part due to the differences in quality of the reference transcriptions.

The HMMs for TREC-7 used HMMs trained on 70 hours of acoustic data and the language model was trained on broadcast news transcriptions ranging in date from 1992 to May 1997 supplied by the LDC and Primary Source Media (about 152 million words in total). The language model training texts also included the acoustic training data (about 700k words). These data were supplemented by 22 million words of texts from the Los Angeles Times and Washington Post covering the span of the evaluation period (June 1997 to April 1998 inclusive). Using all these sources a 65k wordlist was chosen from the combined word frequency list whilst ensuring that the number of new pronunciations which had to be created was manageable. The final wordlist had an OOV rate of 0.3% on the TREC-7 data.

Development work on the TREC-6 test corpus was done using two HTK based systems. The two pass system (HTK-2) was similar in design to the final TREC-7 system but used HMMs trained on only the allowable 35 hours of acoustic training data and used a reduced set of texts for language model training data and only data that was allowable for the TREC-6 tests. This system gave a word error rate of 24.1% on the TREC-6 test data. The other system used in TREC-6 development was a single pass system (HTK-1) and ran in about 45 times real time. This was similar to the first pass of the two pass system but used more pruning and gave a word error rate 28.6% on the TREC-6 SDR test data.

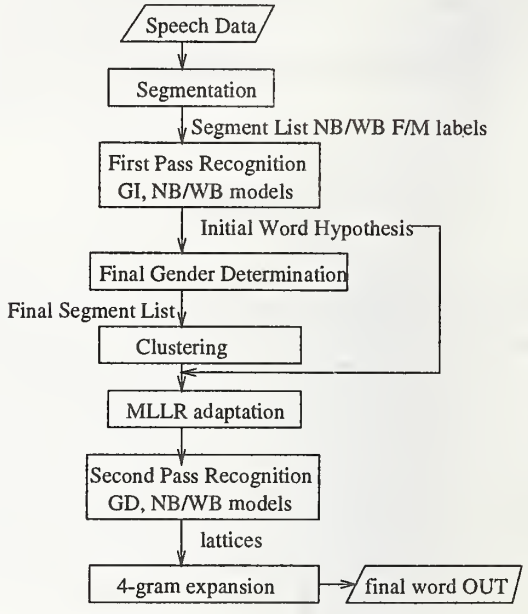


Figure 1: Processing for SDR Speech Recognition

2.1. WER results from Cross Recogniser Runs

We have also used alternative automatic transcriptions to assess the effect of error rate on retrieval performance, namely, for TREC-6 the baseline supplied by NIST (computed by IBM) and the transcription obtained by Sheffield University [1]. For TREC-7 there are the 2 NIST-supplied baselines generated from the CMU recogniser, and cross recogniser runs from Dragon, ATT, Sheffield and DERA. The full set of comparisons with other SDR sites is given in Table 3. [1]

TREC-6 TEST DATA	Corr.	Sub	Del	Ins	Err
NIST/IBM Baseline	59.1	33.6	7.3	9.1	50.0
Sheffield	66.0	25.3	8.7	5.8	39.8
HTK-1	77.3	17.5	5.2	5.9	28.6
HTK-2	80.8	14.6	4.6	4.9	24.1

TREC-7 TEST DATA	Corr.	Sub	Del	Ins	Err
CUHTK	79.5	15.6	4.8	4.3	24.8
Dragon	74.6	18.6	6.8	4.3	29.8
ATT	73.7	20.4	5.9	4.8	31.0
NIST/CMU base1	72.1	22.6	5.3	6.7	34.6
Sheffield	69.5	23.7	6.8	5.4	35.8
NIST/CMU base2	65.8	30.1	4.1	12.9	47.1
DERA run 2	47.3	44.8	7.9	8.8	61.5
DERA run 1	39.7	47.7	12.6	5.9	66.2

Table 3: WER results for development and evaluation

[1] NB: development WERs were found on a document (story) basis, but evaluation WERs were on an episode basis.

3. IR SYSTEM DEVELOPMENT

3.1. Benchmark System

Our benchmark retriever was the Okapi-based system used by the City/ Cambridge University collaboration for quasi-spoken document retrieval in the TREC-6 evaluation [11]. The overall SDR system architecture is illustrated in Figure 2.

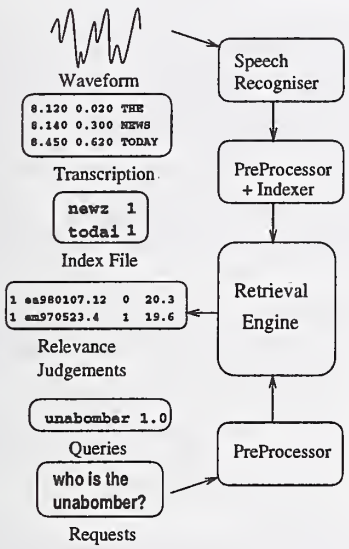


Figure 2: The overall SDR system architecture

The IR system is split into two stages. Firstly a preprocessor stops and stems the words in all the documents using a Porter stemmer [8] and an inverted index file is generated which contains the number of documents in the collection N, the length of each document, $dl(j)$, the number of documents containing each query term, $n(i)$, and the number of times the term occurs in the given document, $tf(i,j)$.

Following [10] and [9] the main retrieval engine generates a score for each document j for each query by summing the combined weights, $cw(i,j)$ for each query term i produced from the formula:

$$cw(i,j) = \frac{(\log N - \log n(i))tf(i,j)(K+1)}{K(1-b+bndl(j)) + tf(i,j)}$$

where $ndl(j)$ is the length of document j normalised by the average dl and K and b are tuning constants. The final ranked list of documents is thus produced for each query by sorting the returned weights in descending order.

We used our two data sets to explore various refinements to this benchmark system, for the moment disregarding whether they can be respectably motivated within the probabilistic model. Thus for example, as there is some demonstrated retrieval value in simple phrases, we experimented with this. Though it is impossible with such a small data set to assess how useful various retrieval devices are as means of offsetting speech recognition errors, the fact that with larger collections error compensation may be more important suggested that it was worth undertaking some initial work on devices that not only seem to have some general utility (as shown in past TRECs), but also may have some particular value in the spoken document context.

The next sections report our comparative experiments using the CU60 queries/assessments for development, and the TREC-7 data for evaluation, with results presented using a subset of the full TREC measures.

3.2. Improvements ? - Stopping

The first preprocessing stage is to remove *stop* words, so IR performance may thus be affected by which words are defined as stop words.

Work was done to stabilise our existing standard stoplist. Initially extra query-specific words, such as *find* and *documents*, were added to the stoplist for queries only. This meant two stop lists were used, one for the documents and one for the queries. This was useful, but we went further and developed a new pair of stoplists specifically for the broadcast news data. Thus 'words' occurring in broadcast news which represent hesitations in speech, such as *uh-huh* or *hmmm* were defined as stop words and finally some common function words which appeared to have been overlooked such as *am* were also added.

The improvement in performance on the development data is given in Table 4. The gain in average precision is 0.7% on the reference and 1.9% on the automatic transcriptions. The greater improvement on the latter is due to the introduction of recogniser-specific hesitations into the stopword list.

	AveP ref	R-P ref	@5docs ref	AveP HTK	R-P HTK	@5docs HTK
T6 Stop	0.6687	0.5931	0.5600	0.6287	0.5583	0.5267
New Stop	0.6758	0.5928	0.5733	0.6478	0.5834	0.5433

Table 4: Effect of using new stop lists for the CU60 data set

The corresponding performance from introducing these new stoplists on the TREC-7 evaluation is given in Table 5. The average precision on the reference increases by 0.3% whilst for the automatic transcriptions, the increase is 1.7% These results confirm the benefit of using the new stopword lists.

	AveP ref	R-P ref	@5docs ref	AveP HTK	R-P HTK	@5docs HTK
T6 Stop	0.4661	0.4481	0.5304	0.4345	0.4242	0.5478
New Stop	0.4689	0.4617	0.5565	0.4512	0.4385	0.5826

Table 5: Effect of using new stop lists for the TREC-7 data set

3.3. Improvements ? - Mapping

We tried adding a mapping list for word variants. Some of these mappings only affect the reference transcriptions, but were included to allow proper reference/speech comparison. Others might be important for the automatically transcribed spoken documents.

Reference transcriptions from previous broadcast news evaluations were used to generate a list of commonly misspelt words. These are mostly names, such as *Chechnia/Chechnya* or *Zuganauf/Zuganov/ Zyuganov*, but the list also include some words or phrases in common usage which are often misspelt, such as *all right/alright* and *baby sit/baby-sit/babysit*. This list was then used to correct the misspelt words by mapping the transcriptions accordingly.

A few synonyms were also added to the mappings to allow words like *United States/U.S.* to be made equivalent. A stemming exceptions list was also made to compensate for known problems with the Porter stemmer, such as equating *news* and *new*, but not *government* and *governmental*. For ease of implementation this was also included in the mapping step.

The effect of adding this mapping stage to the preprocessing when using the new stoplists is given in Table 6 for the development data and Table 7 on the evaluation data.

	AveP ref	R-P ref	@5docs ref	AveP HTK	R-P HTK	@5docs HTK
New Stop	0.6758	0.5928	0.5733	0.6478	0.5834	0.5433
+ Mapping	0.6960	0.6191	0.5867	0.6746	0.6217	0.5533

Table 6: Effect of adding mapping for the CU60 data set

	AveP ref	R-P ref	@5docs ref	AveP HTK	R-P HTK	@5docs HTK
New Stop	0.4689	0.4617	0.5565	0.4512	0.4385	0.5826
+ Mapping	0.4769	0.4694	0.5565	0.4422	0.4344	0.5565

Table 7: Effect of adding mapping for the TREC-7 data set

The mapping increased average precision by 2.0% on the reference and 2.7% on the automatic transcriptions for the development data. The corresponding effect on the TREC-7 evaluation data was disappointing, with an increase of 0.8% on the reference, but a *decrease* of 0.9% on the automatic transcriptions. When the systems were analysed in more detail, it was found that the average precision on the HTK transcriptions remained unaltered for 5 queries, increased for 11 queries, and decreased for 7 queries. The largest decrease occurred on query 70:
What are the latest developments in gun control in the U.S.? In particular, what measures are being taken to protect children from guns?
After stopping this became:
latest developments gun control us| particular measures protect children guns

Since the training transcriptions did not always consistently write the word *gunshot* as either one or two words, the mapping file contained the map `[gun+shot → gunshot]`. Unfortunately, although this would have helped had the query contained the word *gunshot*, it actually degraded performance in this case, as an instance of the word *gun* had effectively disappeared from two relevant documents.

Difficulties with words like this exist whether or not mapping is carried out. For example, suppose the word *gunshot* had been in the query and no mapping had been implemented: our system would not have found stories transcribed as *gun shot*. Therefore, whether a given mapping is beneficial or not, may depend on exactly what the query terms are. This is an inherent difficulty with this type of system and requires either some expansion, or a way of allowing words such as *gunshot* to match both *gunshot* and *gun + shot* during the scoring to solve it.

A similar problem arises with query 62:
Find reports of fatal air crashes.

If the mapping `[air+force → airforce]` is implemented, then the score for the relevant document `ee970703.22` which contains two instances of the word-pair `[air+force]` decreases from rank 4 to rank 47 due to the two occurrences of the word *air*, which can no longer be found.

The conclusion is therefore that mapping in order to correct bad spellings and allow exceptions to the stemming algorithm is a good thing and will improve retrieval performance. However, mappings which convert two words into one when it is not always clear whether the word should exist as one word, a hyphenated word, or two separate words, should be used sparingly, at the system's peril.

3.4. Improvements ? - Word-Pair Modelling

Past TREC tests have shown that there is some value in the use of phrasal terms, though these need not be linguistically, as opposed to statistically, defined. The most common method is to use a file-based phrasal vocabulary. However linguistically-motivated phrasal terms drawn from the request topic have also been used e.g. with INQUERY, and we decided to try this.

Each query was tagged using a Brill tagger [2] and pairs of adjacent words with no interceding punctuation, which followed the sequence *N/N* or *J/N*, where *N* is a noun or name and *J* is an adjective, were marked as word-pairs. These word-pairs were then weighted and added to the query terms. The indexing procedure was refined to allow Term Position Indexes (TPIs) to be stored in an augmented inverted file. The normal combined weight measure was applied to the word-pairs, after using the TPIs to find which documents the word-pairs occurred in.

An example of the word-pairs added in this way is:

CU60 : *How many people have been murdered by the IRA in Northern Ireland*
north_ireland

TREC-7: *What are the latest developments in gun control in the U.S.? In particular, what measures are being taken to protect children from guns?*
gun_control

The effect of applying this word-pair modelling on the original TREC-6 system and the system after the new stoplists and mapping had been applied are given in Tables 8 and 9 for the CU60 and TREC-7 tasks respectively. The lines labelled 'a'(lone) show the impact of this device alone, the lines labelled 'c'(ombined) show the impact of the word-pair device when *added* to the previous stopping and mapping.

		AveP ref	R-P ref	@5docs ref	AveP HTK	R-P HTK	@5docs HTK
a		0.6687	0.5931	0.5600	0.6287	0.5583	0.5267
a	+wp	0.6690	0.5898	0.5633	0.6371	0.5709	0.5267
c		0.6960	0.6191	0.5867	0.6746	0.6217	0.5533
c	+wp	0.7015	0.6288	0.5867	0.6760	0.6216	0.5500

Table 8: Effect of adding word-pair weights for the CU60 data set

		AveP ref	R-P ref	@5docs ref	AveP HTK	R-P HTK	@5docs HTK
a		0.4661	0.4481	0.5304	0.4345	0.4242	0.5478
a	+wp	0.4597	0.4246	0.5130	0.4287	0.4097	0.5391
c		0.4769	0.4694	0.5565	0.4422	0.4344	0.5565
c	+wp	0.4714	0.4549	0.5652	0.4423	0.4199	0.5565

Table 9: Effect of adding word-pair weights for the TREC-7 data set

The addition of word-pair information in development increased the average precision of the combined system by 0.6% on the reference and 0.1% on the automatic transcriptions. The device was therefore included in the evaluation system. Unfortunately it had no effect on average precision for the automatically generated TREC-7 transcriptions and actually worsened performance on the reference. This may be influenced by the number of (stopped) query terms (3.417 per query for CU60, 7.13 for TREC-7) and the number of word-pairs added (1.18 per query for CU60, 1.61 for TREC-7), or may be a result of the different properties of the document sets.

3.5. Improvements ? - Part-of-Speech Weighting

Work in the past within the Okapi framework has not significantly investigated the use of explicit, as opposed to implicit, linguistic term characterisation within the probabilistic model (though the model does not constrain linguistic criteria for the initial choice of base terms). We nevertheless decided to study the use of linguistic information, admittedly in an fairly ad-hoc way, but with a view to possibly exploring it more rigorously later.

It seems that certain classes of words convey more information than others. For example, proper names are generally more helpful in finding specific information than commonly used verbs. To exploit this fact different weights were given to the query terms depending on their part-of-speech.

The query terms were tagged using the Brill tagger and then subsequently divided into one of four groups: Proper Noun (PN), Common Noun (CN), Adjective or Adverb (AA) and the rest, mainly consisting of verbs and hence denoted VB. The weights which gave the greatest increase in average precision on the development data and were therefore incorporated into the retrieval system proved to be:

Proper Noun (names)	1.2
Common Noun	1.1
Adjective & Adverbs	1.0
Verbs and the rest	0.9

confirming the belief that names generally hold more specific indications than common nouns, which in turn are better then adjectives, adverbs and verbs.

The effect of applying this POS weighting on the original TREC-6 system and the system after the new stoplists, mapping and word-pairs had been applied are given in Tables 10 and 11 for the CU60 and TREC-7 tasks respectively. [2] Also included in Table 11 are the results for the case of `[PN=1.3 CN=1.2 AA=1.0 VB=0.8]` labelled as *opt* as they provided an optimal set for the improvement in average precision for the HTK transcriptions on the TREC-7 evaluation data. The results without including word-pairs on the TREC-7 data are given as a comparison and labelled as *c-wp*.

		AveP ref	R-Prec ref	@5docs ref	AveP HTK	R-Prec HTK	@5docs HTK
a		0.6687	0.5931	0.5600	0.6287	0.5583	0.5267
a	+ POS	0.6750	0.6013	0.5633	0.6309	0.5552	0.5333
c		0.7015	0.6288	0.5867	0.6760	0.6216	0.5500
c	+ POS	0.7109	0.6402	0.5867	0.6802	0.6216	0.5533

Table 10: Effect of adding POS weighting for the CU60 data set

Implementing the chosen POS weights gave an increase in average precision of 0.94% on the reference and 0.42% on the HTK transcriptions for the CU60 development set and 0.94% on the reference and 0.76% on the HTK transcriptions for the TREC-7 evaluation set. Increasing the relative weights of nouns further for the TREC-7 task can be seen to increase average precision on the HTK-recogniser run, whilst leaving the reference unaffected. An additional increase can be gained by removing the word-pair device.

[2] A small bug in the integration of POS weighting and mapping resulted in the gain in the submitted evaluation run being slightly lower than that quoted here.

		AveP ref	R-Prec ref	@5docs ref	AveP HTK	R-Prec HTK	@5docs HTK
a		0.4661	0.4481	0.5304	0.4345	0.4242	0.5478
a	+POS	0.4695	0.4328	0.5652	0.4446	0.4310	0.5565
a	+opt	0.4737	0.4420	0.5739	0.4460	0.4394	0.5739
c		0.4714	0.4549	0.5652	0.4423	0.4199	0.5565
c	+POS	0.4808	0.4636	0.5913	0.4499	0.4372	0.5652
c	+opt	0.4807	0.4636	0.5913	0.4524	0.4439	0.5913
c-wp		0.4769	0.4694	0.5565	0.4422	0.4344	0.5565
c-wp	+POS	0.4869	0.4818	0.5913	0.4499	0.4408	0.5739
c-wp	+opt	0.4852	0.4673	0.6000	0.4544	0.4588	0.6000

Table 11: Effect of adding POS weighting for the TREC-7 data set

Care should be taken when increasing the difference between the query POS weights since this naturally interacts with the stemming because the information about part-of-speech in the documents is generally lost during the stemming procedure.

3.6. Improvements ? - Statistical Pre-Search Expansion

Following widespread TREC practice we decided to try pre-search expansion using statistical term co-occurrences, [10, 13] both as a device for including terms related to the query terms and to compensate for inadequate stemming. In principle this should be based on actual text data, for example, by using a parallel text corpus, but we used the transcribed document set. Note this is not ideal due to the document set being small and the presence of transcription errors.

Our expansion system, e(xpansion), adds a new stem E based on an original stem O when the probability of the original term being present in a document given that the expanded term is present, $P(O|E)$, is greater than a half. This is equivalent to saying the expanded stem is more likely to occur when the original stem is present than when it is absent. Only stems which occur in more documents than the original were added and terms which occur in over 2% of the documents were not expanded to reduce over-expansion problems.

An additional development , r(oots), was also included to enhance the stemming process. This involved adding stems which have a common root of at least five letters with the original stem.

The basic weight assigned to an expanded term is $P(O|E)$, namely the probability of the original term occurring given the expanded term. The weights for the original source and expanded terms are both normalised to give a total equal to the original term weight. Thus for a term weight, T, the weights for the original and expanded terms are:

$$O = \frac{T}{T + \sum_i P(O|E_i)} \qquad E = \frac{P(O|E)}{T + \sum_i P(O|E_i)}$$

The results are given in Tables 12 and 13 for the CU60 and TREC-7 data respectively.

		AveP ref	R-P ref	@5docs ref	AveP HTK	R-P HTK	@5docs HTK
a		0.6687	0.5931	0.5600	0.6287	0.5583	0.5267
a	+e	0.6695	0.5951	0.5633	0.6286	0.5592	0.5267
a	+r	0.6816	0.6194	0.5600	0.6381	0.5737	0.5233
c		0.7109	0.6402	0.5867	0.6802	0.6216	0.5533
c	+e	0.7121	0.6411	0.5900	0.6804	0.6225	0.5533
c	+r	0.7060	0.6352	0.5833	0.6797	0.6204	0.5533

Table 12: Effect of adding expansion for the CU60 data set

		AveP ref	R-P ref	@5docs ref	AveP HTK	R-P HTK	@5docs HTK
a		0.4661	0.4481	0.5304	0.4345	0.4242	0.5478
a	+e	0.4668	0.4481	0.5304	0.4373	0.4242	0.5478
a	+r	0.4687	0.4425	0.5478	0.4376	0.4192	0.5478
c		0.4808	0.4636	0.5913	0.4499	0.4372	0.5652
c	+e	0.4868	0.4673	0.5913	0.4565	0.4408	0.5739
c	+r	0.4953	0.4624	0.6000	0.4533	0.4438	0.5739
c-wp		0.4869	0.4818	0.5913	0.4499	0.4408	0.5739
c-wp	+e	0.4868	0.4673	0.5913	0.4556	0.4408	0.5739
c-wp	+r	0.4935	0.4624	0.6000	0.4521	0.4438	0.5739

Table 13: Effect of adding expansion for the TREC-7 data set

These results[3] show that the basic expansion, exp1, increases average precision on both the benchmark and combined system for CU60 queries. The addition of roots to the expansion process increases average precision on the benchmark CU60 system, but does not for the combined system. This is probably because the additional terms are added as a consequence of bad performance by the stemmer, for example *Californian* → *California*, *teaching* → *teacher*. Hence, when the expansion was added to the system which already included the stemming exceptions list, the performance no longer increased. The system used in the evaluation, therefore used the basic expansion system, exp1, but not the roots expansion as it was thought the stemmer-exceptions list offered better compensation for the problems with stemming.

The expansion device increased average precision on the combined TREC-7 system by 0.6% on the reference and 0.7% on the automatic transcriptions. The roots expansion decreased average precision on the automatic transcription as predicted, but actually increased average precision on the reference system. This may be partially due to the fact that the stemmer-exceptions list was manually generated from just the TREC-6 test data.

It is important to note that only a few words in a few queries have been expanded. The effect of this kind of query expansion could therefore be very different with a larger set of queries and documents. In addition, term co-occurrences may be better estimated using a larger distinct but similar collection of documents (e.g previous broadcast news stories).

3.7. Improvements ? - Tuning Model Parameters

Finally the model parameters b and K were tuned to give maximum average precision on the HTK transcriptions for the combined system on the CU60 data set. The results for this are given in Table 14 for the CU60 data set and 15 for the TREC-7 data set.

		AveP ref	R-P ref	@5docs ref	AveP HTK	R-P HTK	@5docs HTK
a		0.6687	0.5931	0.5600	0.6287	0.5583	0.5267
a	+tune	0.6686	0.5863	0.5633	0.6327	0.5603	0.5333
c		0.7121	0.6411	0.5900	0.6804	0.6225	0.5533
c	+tune	0.7082	0.6352	0.5900	0.6832	0.6305	0.5567

Table 14: Effect of tuning model parameters on the CU60 data set

[3]These results are better than those submitted due to a bug in the expansion code.

		AveP ref	R-P ref	@5docs ref	AveP HTK	R-P HTK	@5docs HTK
a		0.4661	0.4481	0.5304	0.4345	0.4242	0.5478
a	+tune	0.4696	0.4438	0.5391	0.4361	0.4221	0.5565
a	+opt	0.4643	0.4365	0.5391	0.4509	0.4215	0.5652
c		0.4868	0.4673	0.5913	0.4565	0.4408	0.5739
c	+tune	0.4935	0.4639	0.6000	0.4572	0.4493	0.5652
c	+opt	0.4928	0.4659	0.5826	0.4834	0.4447	0.5739
c-wp		0.4868	0.4673	0.5913	0.4556	0.4408	0.5739
c-wp	+tune	0.4903	0.4639	0.6000	0.4567	0.4493	0.5652
c-wp	+opt	0.4854	0.4722	0.5913	0.4686	0.4438	0.5478

Table 15: Effect of tuning model parameters on the TREC-7 data set

4. SUMMARY OF IR RESULTS

4.1. Summary of Results on CU60 data

To summarise the foregoing results we show two Tables 16 and 17 showing respectively the separate contributions of the various devices detailed above, and their combined effects. The overall improvement in average precision on the HTK transcriptions from adding all these new features is 5.5%.

The equivalent results for TREC-7 are given in section 4.2.

	AveP ref	R-P ref	@5docs ref	AveP HTK	R-P HTK	@5docs HTK
orig	0.6687	0.5931	0.5600	0.6287	0.5583	0.5267
stop	0.6758	0.5928	0.5733	0.6478	0.5834	0.5433
"+map	0.6960	0.6191	0.5867	0.6746	0.6217	0.5533
wp	0.6690	0.5898	0.5633	0.6371	0.5709	0.5267
POS	0.6750	0.6013	0.5633	0.6309	0.5552	0.5333
exp	0.6695	0.5951	0.5633	0.6286	0.5592	0.5267
tune	0.6686	0.5863	0.5633	0.6327	0.5603	0.5333

Table 16: Effect of devices applied separately on the CU60 data set

	AveP ref	R-P ref	@5docs ref	AveP HTK	R-P HTK	@5docs HTK
orig	0.6687	0.5931	0.5600	0.6287	0.5583	0.5267
+stop	0.6758	0.5928	0.5733	0.6478	0.5834	0.5433
+map	0.6960	0.6191	0.5867	0.6746	0.6217	0.5533
+wp	0.7015	0.6288	0.5867	0.6760	0.6216	0.5500
+POS	0.7109	0.6402	0.5867	0.6802	0.6216	0.5533
+exp	0.7121	0.6411	0.5900	0.6804	0.6225	0.5533
+tune	0.7082	0.6352	0.5900	0.6832	0.6305	0.5567

Table 17: Effect of devices applied in combination on the CU60 data set

4.2. Summary of Results on TREC-7 data

It can be seen that although it was previously thought that adding the word-pair device decreases performance, in fact the performance is higher if the word-pair information is included. This shows the dangers of making general conclusions from such small increases in precision on a relatively small data set. It also illustrates the interaction that occurs between the devices.

	AveP ref	R-P ref	@5docs ref	AveP HTK	R-P HTK	@5docs HTK
orig	0.4661	0.4481	0.5304	0.4345	0.4242	0.5478
stop	0.4689	0.4617	0.5565	0.4512	0.4385	0.5826
"+map	0.4769	0.4694	0.5565	0.4422	0.4344	0.5565
wp	0.4597	0.4246	0.5130	0.4287	0.4097	0.5391
POS	0.4695	0.4328	0.5652	0.4446	0.4310	0.5565
exp	0.4668	0.4481	0.5304	0.4373	0.4242	0.5478
tune	0.4696	0.4438	0.5391	0.4361	0.4221	0.5565

Table 18: Devices applied separately on the TREC-7 data set

	AveP ref	R-P ref	@5docs ref	AveP HTK	R-P HTK	@5docs HTK
orig	0.4661	0.4481	0.5304	0.4345	0.4242	0.5478
+stop	0.4689	0.4617	0.5565	0.4512	0.4385	0.5826
+map	0.4769	0.4694	0.5565	0.4422	0.4344	0.5565
+wp	0.4714	0.4549	0.5652	0.4423	0.4199	0.5565
+POS	0.4808	0.4636	0.5913	0.4499	0.4372	0.5652
+exp	0.4868	0.4673	0.5913	0.4565	0.4408	0.5739
+tune	0.4935	0.4639	0.6000	0.4572	0.4493	0.5652
run*	0.4817	0.4603	0.6000	0.4509	0.4330	0.5565

* The loss in the submitted run was due to bugs in the POS weighting and expansion code

Table 19: Devices applied in combination on the TREC-7 data set

	AveP ref	R-P ref	@5docs ref	AveP HTK	R-P HTK	@5docs HTK
orig	0.4661	0.4481	0.5304	0.4345	0.4242	0.5478
+stop	0.4689	0.4617	0.5565	0.4512	0.4385	0.5826
+map	0.4769	0.4694	0.5565	0.4422	0.4344	0.5565
+POS	0.4869	0.4818	0.5913	0.4499	0.4408	0.5739
+exp	0.4868	0.4673	0.5913	0.4556	0.4408	0.5739
+tune	0.4903	0.4639	0.6000	0.4567	0.4493	0.5652

Table 20: Cumulative Improvements on TREC-7 without wp

5. REPRESENTING RECOGNITION ACCURACY IN SDR

5.1. Word and Term Error Rates

Speech recognition accuracy is conventionally expressed in terms of word error rate (WER). To calculate this an alignment of the hypothesised and reference transcriptions is made and the number of insertion (I), deletion (D) and substitution (S) errors are found. For W words in the reference transcription, the word error is then given by:

$$\text{WER} = \frac{(S+I+D)}{W}.100\%$$

When the transcriptions are subsequently used for information retrieval, WER does not accurately reflect the input to the retrieval stage (see figure 3). Firstly, stop words are removed, some words are mapped, and the words are stemmed; secondly, the order of the words is not considered in the standard retrieval case, so an alignment is not necessary, and finally a traditional substitution error can be thought of as two errors, as it not only misses a correct word, but also introduces a spurious one. When investigating recognition accuracy for SDR, we therefore use a Term Error Rate

$$\text{TER} = \frac{\sum_w |A(w) - B(w)|}{W}.100\%$$

where $A(w)$ and $B(w)$ represent the number of times word w occurs in the reference A and the transcription B. TER therefore models the output of the pre-processor rather than the speech recogniser and is more appropriate when considering subsequent retrieval performance.

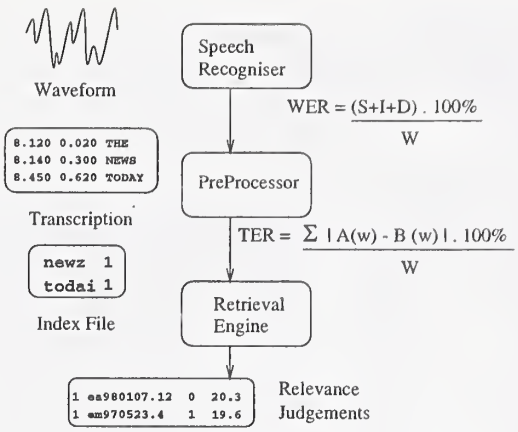


Figure 3: Defining Recognition Accuracy during Processing for SDR

5.2. Stopping, Stemming and Mapping

The pre-processing stages of stopping, stemming and mapping have a great influence on the property of the data input to the information retriever. For example, the number of words for each stage for the TREC-6 and TREC-7 test data using our TREC-7 preprocessor is given in Table 21.

TREC-6 DATA

Recogniser	original	+stop	+map	+abbrev+stem
Reference	408036	199861	198971	193383
IBM Baseline	404559	188117	187595	184214
Sheffield	382855	186447	185937	182864
HTK-1	397942	183475	182869	178805
HTK-2	393592	185527	184951	181105

TREC-7 DATA

Recogniser	original	+abbrev	+map	+stop	+stem
Reference	765274	757848	756262	354258	354250
CUHTK	764707	757141	755774	347364	347322
Dragon	749253	742857	741633	348581	348578
ATT	759899	753153	751890	340680	340680
Base1	787199	780518	779252	349221	349159
Sheff	757870	750966	749889	361135	361053
Base2	845284	838188	836757	344622	344535
DERA2	776151	770109	769259	359326	359296
DERA1	717027	712844	712238	371499	371443

Table 21: Number of Words for TREC-6 and TREC-7 SDR after various stages of processing

5.2.1. *Word Error Rates*

The corresponding WER at each of these processing stages is given in Table 22 and the relationship between these WERs is shown in Figure 4.

The results on the TREC-6 data suggest that the WER after stopping and stemming can be predicted reasonably accurately from the original WER. However, this is not as clear from the TREC-7 results. The DERA1 run is the only one where the WER goes up after stopping, meaning that the stopped words are recognised better on average than the non-stop words. This is not usually the case for ASR systems, since the smaller stop-words which carry less information content are generally more confusable than content words.

TREC-6 DATA

Recogniser	original	+stop	+map	+abbrev+stem
IBM Baseline	50.0	47.5	47.2	44.3
Sheffield	39.8	37.6	37.1	34.6
HTK-1	28.6	24.9	24.7	22.2
HTK-2	24.1	21.5	21.2	18.7

TREC-7 DATA

Recogniser	original	+abbrev	+map	+stop	+stem
CUHTK	24.8	25.0	24.9	22.8	22.3
Dragon	29.8	29.9	29.8	27.6	26.8
ATT	31.0	31.2	31.1	28.2	27.4
Base1	34.6	34.3	34.2	30.8	30.0
Sheffield	35.8	36.0	35.9	34.4	33.4
Base2	47.1	47.2	47.1	43.4	42.0
DERA2	61.5	61.7	61.6	60.0	59.0
DERA1	66.2	66.4	66.3	69.1	67.7

Table 22: % Word Error Rate for TREC-6 and TREC-7 SDR

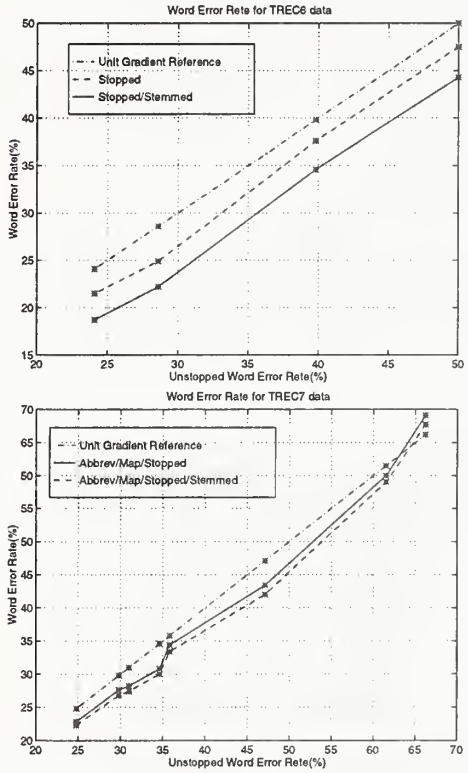


Figure 4: Correlation between Word Error Rates whilst preprocessing.

5.2.2. *Term Error Rates*

Term error rates are more appropriate when considering speech recognition for SDR problems because they model the input to the retriever more accurately. Note, not producing any output gives a TER of 100% whereas misrecognising every word as on OOV word produces a TER of 200%, due to each substitution error counting as both an insertion and deletion error. Misrecognising every word will in practise give a TER of below 200% as the word ordering is unimportant, so some recognition errors will cancel out.

The corresponding TER at each of the pre-processing stages is given in Table 23 and the relationship between these TERs is shown in Figure 5.

It is interesting to note that on both data sets at low TER, complete preprocessing does not seem to affect the TER. As the recognition performance of the system decreases, so the effect on TER of preprocessing increases.

TREC-6 DATA

Recogniser	original	+stop	+map+abbrev+stem
IBM Baseline	61.1	73.7	67.4
Sheffield	48.4	59.2	53.0
HTK-1	32.9	37.6	32.8
HTK-2	28.2	32.8	28.2

TREC-7 DATA

Recogniser	original	+abbrev+stop	+stem	+map
CUHTK	31.6	37.2	32.7	32.1
Dragon	36.9	44.6	39.6	39.0
ATT	39.5	45.9	40.8	40.2
Base1	43.5	50.2	45.0	44.3
Sheffield	45.6	57.0	51.2	50.4
Base2	59.4	70.7	64.2	63.4
DERA2	81.5	98.7	92.3	91.7
DERA1	89.9	114.7	107.3	106.7

Table 23: % Term Error Rate for TREC-6 and TREC-7 SDR

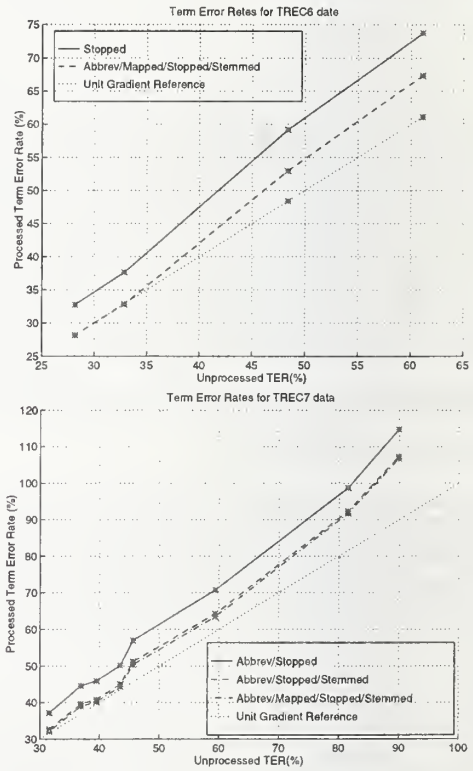


Figure 5: Correlation between Term Error Rates whilst preprocessing.

It is interesting to compare the relationship between our new metric, the stopped/ stemmed/ mapped TER and the standard measure of speech recognition performance, namely the unprocessed word error rate. A graph showing the relationship between these is shown in Figure 6.

The difference between unprocessed WER and processed TER increases as WER increases. This implies that in fact the input to the retriever degrades more rapidly than would be predicted from the WER.

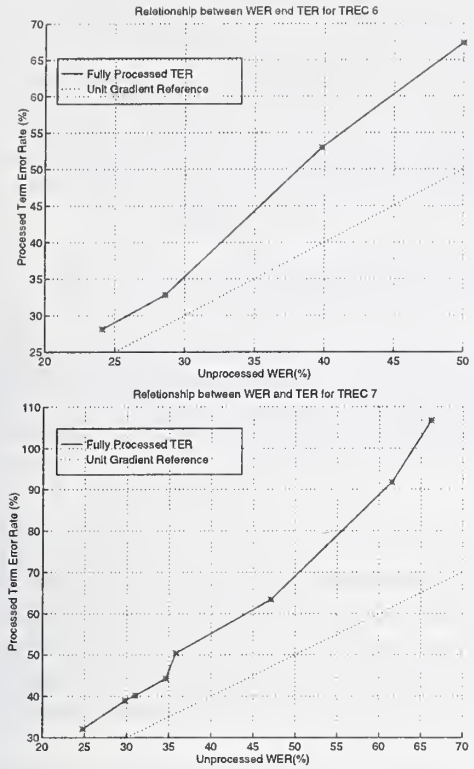


Figure 6: Relationship between Unprocessed WER and Processed TER

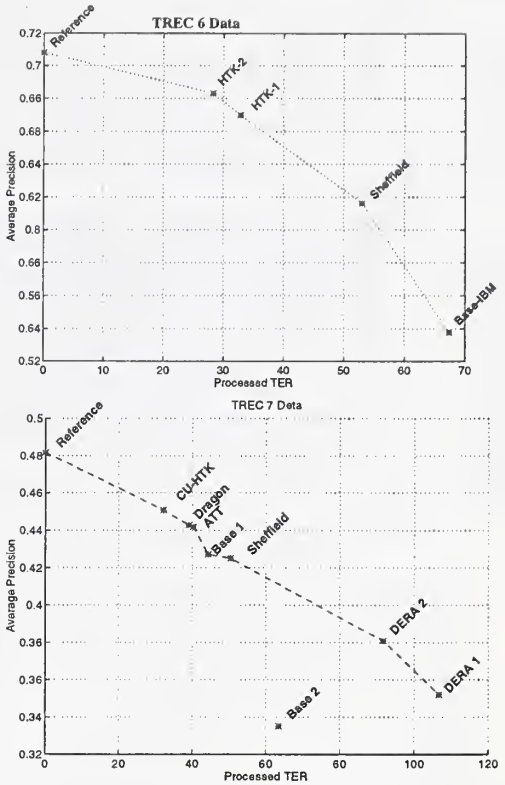


Figure 7: Relationship between TER and Average Precision

It is also interesting to realise that stopping the documents increases term error rate, although it decreases (aligned) word error rate. This is thought to be because the majority of cancelling errors occur with the shorter, stopped words, so the cancelling effect is reduced by stopping, hence increasing TER. Stemming will always reduce both WER and TER.

6. RELATIONSHIP BETWEEN TER AND IR PERFORMANCE

The average precision, R precision and precision at 5, 10 and 30 documents recall is given for the different SDR runs in Table 24. This relationship between the average precision and the stop/stem/map TER is plotted in Figure 7 with R-precision in Figure 8. The interpolated recall-precision averages, plotted in Figure 9 show the IR performance of the different systems.

These results show that in general average precision decreases with TER. The insertion rate may be an important influence on this as the TREC-7 base 2 recogniser, which has an insertion rate of 13.0% as opposed to the others which have between 4.3 and 8.8%, seems to produce worse IR performance than predicted. The relationship between R-precision and TER is not as clear cut and the precision-recall graphs show the degradation of IR performance on TREC-7 is not just related to TER.

7. CONCLUSIONS

It is not possible to draw any strong conclusions from our TREC-7 experiments. This is partly because, in contrast to TREC as a whole, there is no trend data: TREC-6 used the different, known-item retrieval task. But more importantly, the test data is too small for reliable and informative inference. It seems to be the case that the basic Okapi-

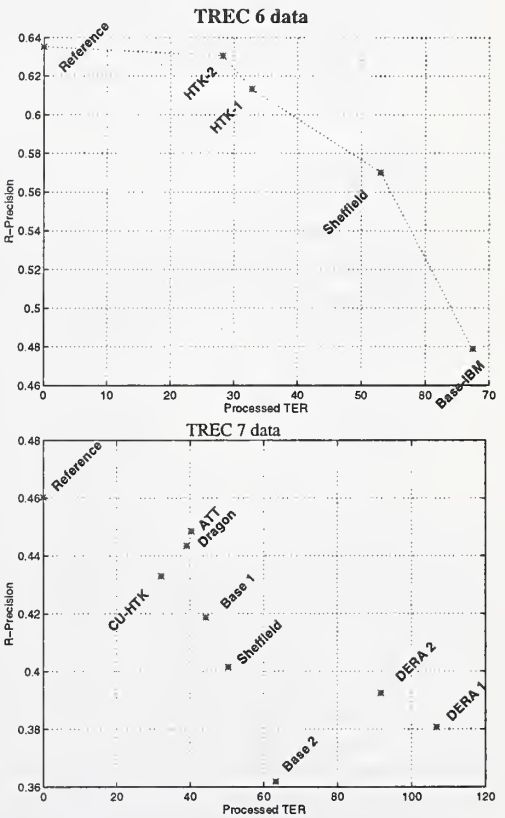


Figure 8: Relationship between TER and R Precision

TREC-6 DATA

	PTER	AveP	R-Prec.	@5docs	@10docs
Reference	0.0	0.7082	0.6352	0.5900	0.4483
HTK-2	28.2	0.6832	0.6305	0.5567	0.4350
HTK-1	32.8	0.6697	0.6134	0.5700	0.4267
Sheffield	53.0	0.6164	0.5701	0.5500	0.4050
IBM Base	67.4	0.5377	0.4787	0.5100	0.3750

TREC-7 DATA

	PTER	AveP	R-Prec.	@5docs	@10docs
Reference	0.0	0.4817	0.4603	0.6000	0.4739
CUHTK	32.1	0.4509	0.4330	0.5565	0.4522
Dragon	39.0	0.4428	0.4434	0.5652	0.4435
ATT	40.2	0.4419	0.4485	0.5652	0.4485
Base1	44.3	0.4272	0.4187	0.5478	0.4261
Sheff	50.4	0.4251	0.4015	0.5478	0.4391
Base2	63.4	0.3352	0.3619	0.4348	0.3826
DERA2	91.7	0.3810	0.3925	0.5217	0.4043
DERA1	106.7	0.3521	0.3806	0.5478	0.3957

Table 24: Effect of TER on IR Performance

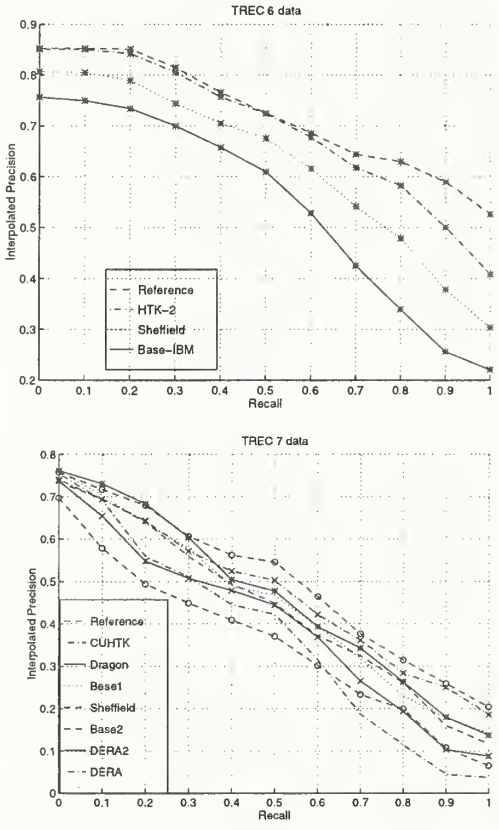


Figure 9: Overall IR performance using the different recognisers

style system works satisfactorily on both reference and automatically-transcribed data, illustrating its power for documents of a rather different discourse type from those used hitherto, and for ones in a different medium. But while retrieval performance using our recogniser transcriptions is near that for the reference data, it is not clear what impact recognition failures would have on retrieval with a much larger data set or very different forms of query. For the same reason, while we can hypothesise that particular retrieval devices may be not just useful in general but particularly appropriate for speech data, we cannot come to any firmly predictive conclusions on their individual or combined value.

We have investigated the effects of changing stop-lists, adding bad-spelling correctors, a stemming exceptions list and basic synonym mapping, including word-pair information, weighting query terms by their part-of-speech and adding pre-search statistical expansion. Whilst all of these have been shown to increase IR performance under certain circumstances, the increases are small. Nevertheless the combination of these devices led to an increase in average precision on the TREC-7 evaluation data of 2.74% on the reference and 2.27% on the automatic transcriptions over last year's system.

Acknowledgements

Our thanks go to Steve Renals at Sheffield for providing their transcriptions for the TREC-6 test data and to Cedric Auzanne at NIST for confirmation of WER results for the TREC-7 evaluation. This work is in part funded by an EPSRC grant reference GR/L49611.

8. REFERENCES

[1] D.Abberley, S.Renals, G.Cook & T.Robinson *The THISL Spoken Document Retrieval System* Proc. TREC-6, pp. 747-752, 1997.

[2] E.Brill *Some Advances in Transformation-Based Part-of-Speech Tagger* Proc. AAAI-94, Vol. 1 pp. 722-727, 1994.

[3] M.F.G.Gales & P.C.Woodland *Mean and Variance Adaptation Within the MLLR Framework* Computer Speech & Language, Vol. 10 pp. 249-264, 1996.

[4] J.S.Garofolo, E.M.Voorhees, V.M.Stanford & K.Spärck Jones *TREC-6 1997 Spoken Document Retrieval Track Overview and Results* Proc. TREC-6, pp. 83-91, 1997.

[5] T.Hain, S.E.Johnson, A.Tuerk, P.C.Woodland & S.J.Young *Segment Generation and Clustering in the HTK Broadcast News Transcription System* Proc. DARPA Broadcast News Transcription and Understanding Workshop, pp. 133-137, Feb. 1998.

[6] S.E.Johnson & P.C.Woodland *Speaker Clustering Using Direct Maximisation of the MLLR-Adapted Likelihood* Proc. ICSLP 98, Vol. 5 pp. 1775-1779, 1998.

[7] C.J.Leggetter & P.C.Woodland *Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models* Computer Speech & Language, Vol. 9 pp. 171-185, 1995.

[8] M.F.Porter *An Algorithm for Suffix Stripping* Program, 14 pp. 130-137, 1980.

[9] S.E.Robertson & K.Spärck Jones *Simple, Proven Approaches to Text Retrieval* Technical Report TR356 Cambridge University Computer Laboratory, May 1997.

[10] K.Spärck Jones, S.Walker & S.E.Robertson *A Probabilistic Model of Information Retrieval : Development and Status* Technical Report TR446 Cambridge University Computer Laboratory, Sept. 1998.

[11] S.Walker, S.E.Robertson, M.Boughanem, G.J.F.Jones & K.Spärck Jones *Okapi at TREC-6 Automatic Ad Hoc, VLC, Routing, Filtering and QSDR* Proc. TREC-6, pp. 125-136, 1997.

[12] P.C.Woodland, T.Hain, S.E.Johnson, T.R.Niesler, A.Tuerk, E.W.D.Whittaker & S.J.Young *The 1997 HTK Broadcast News Transcription System* Proc DARPA Broadcast News Transcription and Understanding Workshop, pp. 41-48, Feb. 1998.

[13] J.Xu & W.B.Croft *Corpus-Based Stemming Using Cooccurence of Word Variants* ACM Trans. on Information Systems, Vol. 16 No. 1 pp. 61-81, 1998.

INQUERY and TREC-7

James Allan, Jamie Callan, Mark Sanderson, Jinxi Xu, and Steven Wegmann*

Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, Massachusetts USA

*Dragon Systems, Inc.
320 Nevada Street
Newton, Massachusetts USA

This year the Center for Intelligent Information Retrieval (CIIR) at the University of Massachusetts participated in only four of the tracks that were part of the TREC-7 workshop. We worked on ad-hoc retrieval, filtering, VLC, and the SDR track. This report covers the work done on each track successively. We start with a discussion of IR tools that were broadly applied in our work.

1 Tools applied

Although UMass used a wide range of tools, from Unix shell scripts, to PC spreadsheets, three major tools were applied across almost all tracks: the Inquery search engine, the InRoute filtering engine, and a a query expansion technique known as LCA. This section provides a brief overview of each of those so that the discussion does not have to repeated for each track.

1.1 Inquery

All tracks other than the filtering track used Inquery[6] as the search engine, sometimes for training, and always for generating the final ranked lists for the test. We used Inquery V3.2, an in-house development version of the Inquery system made available by the CIIR (V3.1). The differences between the two are not consequential for this study.

The current belief function used by Inquery to calculate the belief in term t within document d is:

$$w_{t,d} = 0.4 + 0.6 \times \frac{\mathrm{tf}_{t,d}}{\mathrm{tf}_{t,d} + 0.5 + 1.5\frac{\mathrm{length}(d)}{\mathrm{avg\ len}}} \times \frac{\log \frac{N+0.5}{n_t}}{\log N + 1}$$

where n_t is the number of documents containing term t, N is the number of documents in the collection, "avg len" is the average length (in words) of documents in the collection, length(d) is the length (in words) of document d, and $\mathrm{tf}_{t,d}$ is the number of times term t occurs in document d.

1.2 InRoute

The InRoute filtering system is based on the same Bayesian inference network model as InQuery. It is designed to operate efficiently in high-volume filtering environments, where incoming documents must be processed rapidly, one at a time. It uses the same document indexing techniques, query language, and scoring

algorithms as InQuery. Corpus statistics for the document stream are estimated using an archival corpus or are learned as documents stream by. InRoute also incrementally learns improved profiles [1] and improved thresholds [4], as relevance judgments become available for documents that have been disseminated.

InRoute was used only in the filtering track.

1.3 Local Context Analysis (LCA)

In SIGIR '96, the CIIR presented a new query expansion technique that worked more reliably than previous "pseudo relevance feedback" methods.[17] That technique, Local Context Analysis (LCA), locates expansion terms in top-ranked passages, uses phrases as well as terms for expansion features, and weights the features in a way intended to boost the expected value of features that regularly occur near the query terms.

LCA has several parameters that affect its results. The first is the choice of LCA database: the collection from which the top ranked passages are extracted. This database could be the test collection itself, but is often another (perhaps larger) collection that it is hoped will broaden the set of likely expansion terms. In the discussion below, if the LCA database is not the test collection itself, we identify what collection was used.

LCA's other two parameters are the number of top passages used for expansion, and the number of expansion features added to the query. The LCA features were put into a query construct that allows a weighted average of the features. Assuming n features, f_1 through f_n, they are combined as:

$$\begin{array}{lcl} \texttt{\#wsum(1.0} & 1.0 & f_1 \\ & \vdots & \vdots \\ & 1-(i-1)*0.9/s & f_i \\ & \vdots & \vdots \\ & 1-(n-1)0.9/s & f_n \texttt{)} \end{array}$$

Here, s is scaling factor that is usually equal to n. The weighted average of expansion features is combined with the original query as follows:

$$\texttt{\#wsum(1.0 1.0 original-query} \qquad w_{\text{lca}} \text{ lca-wsum)}$$

where w_{lca} is the weight that the LCA features are given compared to the original query. Note that the final query is a weighted combination of the original query and the expansion features. As will be discussed below, in the SDR track the combination was unintentionally done differently, slightly shifting the balance between the original query and the expansion concepts.

2 Ad-hoc track

Considering the excellent results of our TREC-6 runs (had we indexed all the documents), we basically copied what we did in TREC-6 with few changes. The main techniques used for TREC-7 are still phrase recognition and query expansion using local context analysis (LCA) [17]. Phrases are recognized from the topic text (title, description and narrative) using a phrase dictionary and are added to a query. Phrase acquisition and recognition are described in the UMass TREC-6 report[2].

As in TREC-6, query expansion is carried out using a database consisting of TREC volumes 1 to 5 except the Federal Register documents. LCA concepts are extracted from the top 30 passages retrieved from the

database for a query. 50 concepts are added to the query with decreasing weights in proportion to their ranks. Referring to the discussion of Section 1.3, for the ad-hoc work we set $s = 70$ and $w_{\text{lca}} = 1.25$.

The effectiveness of all query expansion techniques that are based on the top retrieved documents (passages) obviously depends on the quality of the top ranked set of documents (passages). As an attempt to improve the quality of the top ranked set, in TREC-7 we used a filter to modify the ranks of the passages (obtained by normal tf·idf ranking) based on whether a passage contains all the title words of the topic. The idea is that the title words are the essential requirements of relevancy and passages missing any of such words are unlikely to be relevant. The effect of the filter is to rank passages containing all the title words before those missing some title words. This is done using the INQUERY `#filreq` operator. We call the technique *filter-require* on the title words (to pass the filter a document is required to match the title words).

2.1 Ad-hoc runs

We submitted three ad-hoc runs in TREC-7. The runs are labeled INQ501, INQ502 and INQ503. Our query processing includes 3 steps:

1. Basic query processing—removing stop words and stop phrases (such as "relevant documents") from the topic texts. Sentences discussing criteria of non-relevance in the narratives (such as "Documents discussing ... are not relevant") are also removed. After that, the narratives still contain a lot of verbiage not directly related to relevance. Therefore we further reduce the size of the narratives by removing the non-content bearing words from them. Words in the narrative of a topic are ranked by the formula

   $$v(t) = freq(t) \times idf(t) \times avtf(t)^{0.7}$$

   where $freq(t)$ is the frequency of t in the narrative, $idf(t)$ is the inverse document frequency of t in the TREC-7 ad-hoc collection and $avtf(t)$ is the average frequency of t in TREC-7 documents when it occurs. Terms with a value less than 0.4 are discarded.

   INQ501 and INQ502 used the title, description and narrative fields. Narrative words are given 30% of the weight of the title and description words. INQ503 used only the title and description fields.

2. Phrase identification (as in TREC6)

3. Query expansion using LCA: adding 50 LCA concepts per query. INQ502 used filter-require on the title words in addition to tf·idf ranking for retrieving the top ranked passages while INQ501 and INQ502 used only tf·idf for that purpose.

2.2 Ad-hoc results of submitted runs

The retrieval results are shown in Table 1. The average precision for INQ501, INQ502 and INQ503 is 0.2739, 0.2815, and 0.2521 respectively. Comparing with the TREC-7 ad-hoc average 0.1822, our results are very satisfactory. Of 50 topics, INQ502 (our best run) is below the average for only 2 topics.

2.3 Ad-hoc analysis

This section considers the various stages of query processing and how they impact effectiveness individually and collectively.

```
    Run:       INQ501     INQ502           INQ503
Total number of documents over all queries
    Retrieved:   50000      50000  (+ 0.0)      50000  (+ 0.0)
    Relevant:     4674       4674  (+ 0.0)       4674  (+ 0.0)
    Rel_ret:      3206       3215  (+ 0.3)       3017  (- 5.9)
Interpolated Recall - Precision Averages at:
    0.00        0.7823    0.8314  (+ 6.3)    0.7730  (- 1.2)
    0.10        0.5883    0.6070  (+ 3.2)    0.5586  (- 5.0)
    0.20        0.4526    0.4579  (+ 1.2)    0.4301  (- 5.0)
    0.30        0.3656    0.3848  (+ 5.3)    0.3434  (- 6.1)
    0.40        0.3101    0.3269  (+ 5.4)    0.2843  (- 8.3)
    0.50        0.2490    0.2571  (+ 3.3)    0.2097  (-15.8)
    0.60        0.1919    0.1906  (- 0.7)    0.1597  (-16.8)
    0.70        0.1393    0.1414  (+ 1.5)    0.1095  (-21.4)
    0.80        0.0893    0.0790  (-11.5)    0.0715  (-19.9)
    0.90        0.0417    0.0449  (+ 7.7)    0.0279  (-33.1)
    1.00        0.0175    0.0175  (+ 0.0)    0.0151  (-13.7)
Average precision (non-interpolated) over all rel doc
                0.2739    0.2815  (+ 2.8)    0.2521  (- 8.0)
Precision at:
       5 docs:  0.5760    0.6160  (+ 6.9)    0.5680  (- 1.4)
      10 docs:  0.5540    0.5800  (+ 4.7)    0.5240  (- 5.4)
      15 docs:  0.5240    0.5333  (+ 1.8)    0.4747  (- 9.4)
      20 docs:  0.4850    0.4950  (+ 2.1)    0.4500  (- 7.2)
      30 docs:  0.4240    0.4347  (+ 2.5)    0.4007  (- 5.5)
     100 docs:  0.2502    0.2572  (+ 2.8)    0.2394  (- 4.3)
     200 docs:  0.1833    0.1840  (+ 0.4)    0.1720  (- 6.2)
     500 docs:  0.1063    0.1070  (+ 0.7)    0.0994  (- 6.5)
    1000 docs:  0.0641    0.0643  (+ 0.3)    0.0603  (- 5.9)
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:      0.3117    0.3178  (+ 2.0)    0.2899  (- 7.0)
```

Table 1: Submitted ad-hoc retrieval runs

2.3.1 Ad-hoc basic processing

Table 2 shows the retrieval results when we only use the basic query processing (i.e., no phrase recognition and no LCA). The results show that retrieval becomes better as queries get longer. However, even very short queries with 2–3 words (the title queries) can still achieve reasonable retrieval performance.

2.3.2 Using phrases in ad-hoc

Table 3 shows that adding phrases to queries causes a modest 3.6% improvement in average precision. The improvement is not statistically significant. This is consistent with our TREC-6 results: phrases can improve retrieval performance but the benefit is limited.

2.3.3 Expanding ad-hoc queries

Table 4 shows that query expansion by LCA causes a substantial improvement in average precision. Precision at all document cut-offs are improved as well. The improvement in average precision is 18.5% and statistically significant (t-test, $p = 0.00004$). This is also consistent with our TREC-6 results: query expansion using LCA can significantly improve retrieval performance.

Run:	title	desc		title_desc		title_desc_narr	
Total number of documents over all queries							
Retrieved:	50000	50000	(+ 0.0)	50000	(+ 0.0)	50000	(+ 0.0)
Relevant:	4674	4674	(+ 0.0)	4674	(+ 0.0)	4674	(+ 0.0)
Rel_ret:	2070	2391	(+15.5)	2547	(+23.0)	2712	(+31.0)
Interpolated Recall - Precision Averages at:							
0.00	0.6984	0.7546	(+ 8.0)	0.7815	(+11.9)	0.8046	(+15.2)
0.10	0.4359	0.4864	(+11.6)	0.5058	(+16.0)	0.5630	(+29.2)
0.20	0.3088	0.3415	(+10.6)	0.3568	(+15.5)	0.4100	(+32.8)
0.30	0.2169	0.2674	(+23.3)	0.2731	(+25.9)	0.3036	(+40.0)
0.40	0.1484	0.2014	(+35.7)	0.2054	(+38.4)	0.2262	(+52.4)
0.50	0.1055	0.1412	(+33.8)	0.1554	(+47.3)	0.1803	(+70.9)
0.60	0.0783	0.1075	(+37.3)	0.1037	(+32.4)	0.1241	(+58.5)
0.70	0.0396	0.0602	(+52.0)	0.0641	(+61.9)	0.0762	(+92.4)
0.80	0.0218	0.0205	(- 6.0)	0.0286	(+31.2)	0.0333	(+52.8)
0.90	0.0140	0.0045	(-67.9)	0.0091	(-35.0)	0.0123	(-12.1)
1.00	0.0010	0.0000	(-100.0)	0.0000	(-100.0)	0.0004	(-60.0)
Average precision (non-interpolated) over all rel doc							
	0.1634	0.1924	(+17.7)	0.2008	(+22.9)	0.2231	(+36.5)
Precision at:							
5 docs:	0.4640	0.5120	(+10.3)	0.5200	(+12.1)	0.5400	(+16.4)
10 docs:	0.4020	0.4600	(+14.4)	0.4820	(+19.9)	0.5120	(+27.4)
15 docs:	0.3667	0.4067	(+10.9)	0.4320	(+17.8)	0.4640	(+26.5)
20 docs:	0.3360	0.3790	(+12.8)	0.3950	(+17.6)	0.4290	(+27.7)
30 docs:	0.2840	0.3220	(+13.4)	0.3353	(+18.1)	0.3727	(+31.2)
100 docs:	0.1658	0.1960	(+18.2)	0.1978	(+19.3)	0.2154	(+29.9)
200 docs:	0.1178	0.1358	(+15.3)	0.1434	(+21.7)	0.1568	(+33.1)
500 docs:	0.0694	0.0772	(+11.2)	0.0829	(+19.5)	0.0884	(+27.4)
1000 docs:	0.0414	0.0478	(+15.5)	0.0509	(+22.9)	0.0542	(+30.9)
R-Precision (precision after R (= num_rel for a query) docs retrieved):							
Exact:	0.2074	0.2435	(+17.4)	0.2504	(+20.7)	0.2712	(+30.8)

Table 2: Basic query processing of ad-hoc queries

2.3.4 Filter-require in ad-hoc queries

When filter-require on title words is used for query expansion (INQ502), the average precision is improved by another 2.8% (Table 5). The improvement at low recall is improved more substantially. The technique does not dramatically improve average performance, but it has a significant impact on individual queries. A few queries are significantly improved. One example is topic 385 about "hybrid fuel cars". Standard tf·idf ranking tends to pick up passages containing many occurrences of "fuel" and "cars" but missing "hybrid", an essential element of relevance for this query. Filter-require corrects this problem by requiring the top ranked passages to contain "hybrid". As a result, it results in more effective query expansion. But a few queries are hurt by this technique. The problem is caused by non-essential words in the titles. One example is the word "risks" in the title of topic 354 "journalist risks"—a document can be relevant without literally using the word "risks". Further investigation is needed to determine the value of this technique.

3 Filtering Track

Our goal for the Filtering track was to test the InRoute filtering system [5] and the threshold-learning algorithm that we used last year [2, 4]. These were relatively modest goals, because there seemed to be *significant* changes from last year's system.

```
    Run:      title_desc_narr    title_desc_narr_phr
Total number of documents over all queries
    Retrieved:   50000     50000  (+ 0.0)
    Relevant:     4674      4674  (+ 0.0)
    Rel_ret:      2712      2753  (+ 1.5)
Interpolated Recall - Precision Averages at:
    0.00        0.8046    0.8003  (- 0.5)
    0.10        0.5630    0.5483  (- 2.6)
    0.20        0.4100    0.4058  (- 1.0)
    0.30        0.3036    0.3179  (+ 4.7)
    0.40        0.2262    0.2476  (+ 9.5)
    0.50        0.1803    0.1880  (+ 4.3)
    0.60        0.1241    0.1367  (+10.2)
    0.70        0.0762    0.0914  (+19.9)
    0.80        0.0333    0.0530  (+59.2)
    0.90        0.0123    0.0242  (+96.7)
    1.00        0.0004    0.0086  (+2050.0)
Average precision (non-interpolated) over all rel doc
                0.2231    0.2311  (+ 3.6)
Precision at:
       5 docs:  0.5400    0.5320  (- 1.5)
      10 docs:  0.5120    0.5200  (+ 1.6)
      15 docs:  0.4640    0.4827  (+ 4.0)
      20 docs:  0.4290    0.4350  (+ 1.4)
      30 docs:  0.3727    0.3887  (+ 4.3)
     100 docs:  0.2154    0.2242  (+ 4.1)
     200 docs:  0.1568    0.1603  (+ 2.2)
     500 docs:  0.0884    0.0910  (+ 2.9)
    1000 docs:  0.0542    0.0551  (+ 1.7)
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:      0.2712    0.2745  (+ 1.2)
```

Table 3: Addition of phrases to ad-hoc queries

One obvious departure from last year's system was that the user preference that influences threshold settings was raised to "high precision"; last year it was set halfway between "high precision" and "high recall", and results were reasonably good. Studies during the Winter and Spring indicated that raising the preferences was risky, because they pushed up the dissemination thresholds, which in turn reduced the amount of training data available to the system.

A second difference is that the initial queries for this year's system were intended to more closely match the query-creation process used for the Ad-hoc track. In the past, the initial filtering queries were very simple. Our hypothesis was that the initial query should be as good as possible, and then be further modified by incremental relevance feedback on documents disseminated during filtering.

An initial assessment suggests that the filtering results are quite poor. We do not yet know why, but it appears that there may have been several causes.

Raising the user dissemination preference was clearly a mistake. It caused thresholds to be too high, and hence no documents were disseminated for many profiles. It also reduced the amount of training data available for learning profiles, resulting in less accurate profiles. Lowering the preference to last year's value causes more documents to be disseminated for more profiles; the resulting increase in training data leads to dramatic precision improvements on many profiles. This accounts for some of the poor filtering results, but not all.

The use of more complex initial queries may also have been a factor, but we have not yet had an opportunity

```
    Run:     title_desc_narr_phr  INQ501
Total number of documents over all queries
    Retrieved:   50000     50000  (+ 0.0)
    Relevant:     4674      4674  (+ 0.0)
    Rel_ret:      2753      3206  (+16.5)
Interpolated Recall - Precision Averages at:
    0.00        0.8003    0.7823  (- 2.2)
    0.10        0.5483    0.5883  (+ 7.3)
    0.20        0.4058    0.4526  (+11.5)
    0.30        0.3179    0.3656  (+15.0)
    0.40        0.2476    0.3101  (+25.2)
    0.50        0.1880    0.2490  (+32.4)
    0.60        0.1367    0.1919  (+40.4)
    0.70        0.0914    0.1393  (+52.4)
    0.80        0.0530    0.0893  (+68.5)
    0.90        0.0242    0.0417  (+72.3)
    1.00        0.0086    0.0175  (+103.5)
Average precision (non-interpolated) over all rel doc
                0.2311    0.2739  (+18.5)
Precision at:
       5 docs:  0.5320    0.5760  (+ 8.3)
      10 docs:  0.5200    0.5540  (+ 6.5)
      15 docs:  0.4827    0.5240  (+ 8.6)
      20 docs:  0.4350    0.4850  (+11.5)
      30 docs:  0.3887    0.4240  (+ 9.1)
     100 docs:  0.2242    0.2502  (+11.6)
     200 docs:  0.1603    0.1833  (+14.3)
     500 docs:  0.0910    0.1063  (+16.8)
    1000 docs:  0.0551    0.0641  (+16.3)
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:      0.2745    0.3117  (+13.6)
```

Table 4: Query expansion (LCA) of ad-hoc queries

to investigate this change sufficiently to draw any conclusions.

4 Spoken Document Retrieval track

This section describes the work by CIIR on the SDR track. Repeating the partnership from last year's TREC submission with Dragon Systems, the CIIR submitted a number of runs primarily investigating alternate configurations of the retrieval system, Inquery. This centered on investigating different sources of evidence for use in automatic query expansion. The experimental set up of the system is first described followed by the motivations for the various configurations tested. The results of the experiments are presented and briefly discussed.

4.1 Recognition

The TREC-7 system used by Dragon is a faster version of their 1997 Hub4 evaluation system, and it was also used to transcribe 1000 hours of broadcast data automatically for the TDT task. This system is fully described in [11] and [12], but we give a brief description here.

```
    Run:        INQ501         INQ502
Total number of documents over all queries
    Retrieved:   50000      50000  (+ 0.0)
    Relevant:     4674       4674  (+ 0.0)
    Rel_ret:      3206       3215  (+ 0.3)
Interpolated Recall - Precision Averages at:
    0.00        0.7823     0.8314  (+ 6.3)
    0.10        0.5883     0.6070  (+ 3.2)
    0.20        0.4526     0.4579  (+ 1.2)
    0.30        0.3656     0.3848  (+ 5.3)
    0.40        0.3101     0.3269  (+ 5.4)
    0.50        0.2490     0.2571  (+ 3.3)
    0.60        0.1919     0.1906  (- 0.7)
    0.70        0.1393     0.1414  (+ 1.5)
    0.80        0.0893     0.0790  (-11.5)
    0.90        0.0417     0.0449  (+ 7.7)
    1.00        0.0175     0.0175  (+ 0.0)
Average precision (non-interpolated) over all rel doc
                0.2739     0.2815  (+ 2.8)
Precision at:
       5 docs:  0.5760     0.6160  (+ 6.9)
      10 docs:  0.5540     0.5800  (+ 4.7)
      15 docs:  0.5240     0.5333  (+ 1.8)
      20 docs:  0.4850     0.4950  (+ 2.1)
      30 docs:  0.4240     0.4347  (+ 2.5)
     100 docs:  0.2502     0.2572  (+ 2.8)
     200 docs:  0.1833     0.1840  (+ 0.4)
     500 docs:  0.1063     0.1070  (+ 0.7)
    1000 docs:  0.0641     0.0643  (+ 0.3)
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:      0.3117     0.3178  (+ 2.0)
```

Table 5: Use of filter-require on ad-hoc runs

4.1.1 Front End

A total of 36 parameters are computed every 10 milliseconds: 12 cepstral parameters, 12 cepstral differences, and 12 cepstral second differences. This set of 36 parameters is linearly transformed using IMELDA techniques [8] to a set of 24 parameters which are used for training and recognition. We use PLP-based cepstra [14], computed in the style of Cambridge/HTK, as reported in [9].

Speaker normalization [10] is used to reduce variability among speakers due to vocal tract length. During the signal processing stage, the frequency scale is "warped" using a piecewise linear transformation.

4.1.2 Acoustic Modeling

The model for a sentence hypothesis is obtained by concatenating models Dragon calls PICs (for "phonemes-in-context"). For this evaluation, Dragon used triphone models as described below. A 51-element phoneme set was used that has syllabic consonants and two stress levels for certain vowels. PICs are modeled using from 3 to 4 nodes, with each node having an output distribution (PEL) and a duration distribution. Which PEL model to employ for any given position of any PIC is determined based on a decision tree whose nodes ask linguistic questions about neighboring phonemes as well as questions about the position of word boundaries. The PEL models themselves are general mixture models with basis components given by multivariate Gaussian distributions with diagonal covariance.

In addition to speaker normalization, Dragon also makes use of rapid adaptation, using linear regression techniques to construct transformations of acoustic parameter space mapping speaker-independent model means to speaker-specific ones. This approach was inspired by, and represents a simplification of, speaker adaptation strategies implemented by Cambridge [15]. Dragon also used speaker adaptation techniques (SAT) during training ([7, 16]): Training speech is force-aligned to transcripts and the usual adaptation transformations are computed mapping speaker-independent model to speaker-specific data, and then a sort of "inverse" transformation is performed on the speech frames. This permits the training of new models with the transformed data which should behave well under test-time adaptation. Dragon used four transformation classes at training, determined by grouping related phonemes. (For another approach to speaker-adaptive training, see [13].)

A thumbnail sketch of the system used follows.

1. The system overall:
   - A amplitude based silence detector is used to break the input into chunks that are 20 to 30 seconds long.
   - A phoneme recognizer is used to produce a more refined chopping of these chunks.
   - The segments are clustered for speaker normalization and unsupervised adaptation.
   - Channel normalization is performed on each segment.
   - Speaker normalization is performed within each cluster by doing a quick, errorful recognition with small acoustic models (5000 PELs) and a small bigram LM (300,000 bigrams), and then re-scoring this transcript with each warp scale in order to pick the best scoring scale.
   - Speaker normalized, SAT models with 12,000 PELs are used along with an interpolated trigram LM to obtain an initial transcription for each cluster.
   - Unsupervised rapid adaptation is performed within each cluster, followed by the final recognition pass using the adapted acoustic models and the same trigram LM.
2. Acoustic training:
   - Used the 100 hours of Broadcast News training data to train the seed models for the SAT process.
   - The SAT models were trained from the above data plus the following:
     - A 24 hour subset of the WSJ1 si284 corpus.
     - The WSJCAM0 training corpus.
     - The 1995 Marketplace development corpus.
3. Grammar training:
   - A trigram language models were trained from 500 million words of text from three sources:
   - The Broadcast News acoustic training transcriptions plus the 1995 Marketplace development transcriptions.
   - The Broadcast News LM training corpus.
   - The 1995 Hub4 and Hub3 newswire texts were combined with 190 million words of commercially available newspaper data collected from the period January 1995 through June 1996.
4. Recognition lexicon description:
   - The three LMs share a 57K vocabulary list constructed by mixing the unigram probabilities from the three LM's and selecting the top 57K words.
   - The mixture weights were determined from preliminary recognition runs on the 1996 devtest: 0.30 for A, 0.50 for B, and 0.20 on C.
5. Speed:
   - The entire system ran at 16 x RT on a 300 MHz P-II.

4.2 Experimental components

Retrieval was performed using the Inquery IR system (see Section 1.1). In addition to use of standard IR techniques such as stop word removal, stemming and a tf·idf-like weighting scheme, Inquery was set up to use two additional proven methods.

First, SDR topics were pre-processed where phrases within the topics were recognized and some proper nouns were expanded with synonyms. This type of processing is the same that was done for the ad-hoc queries, and is described in last year's report.[2]

Second, SDR topics were automatically expanded using Local Context Analysis (LCA), essentially as described in Section 1.3. The difference is that the expansion terms were added to the query as "siblings" of the query features rather than balancing the two in an overall weighted sum. The combined version of the query therefore had the form:

$$\texttt{\#wsum(1.0} \quad \begin{matrix} w_{q_1} & q_1 \\ \vdots & \vdots \\ w_{q_i} & q_i \\ \vdots & \vdots \\ w_{q_m} & q_m \\ \\ 1.0 & f_1 \\ \vdots & \vdots \\ 1-(i-1)*0.9/s & f_i \\ \vdots & \vdots \\ 1-(n-1)0.9/s & f_n \end{matrix} \texttt{)}$$

where q_i are the original query features (after processing) and w_{q_i} is the weight assigned to that feature. In the case of SDR, 60 expansion features were add($n = 60$), expansion weights were assigned with $s = 60$. We discuss the impact of this alternate form of LCA expansion below.

Normally, the two retrieval steps of LCA are performed on the same collection; however, this is not required and it is possible that other collections of text can be used in the initial retrieval to provide expansion terms. All the variation in configurations of Inquery centered around what form of collection the initial retrieval was performed on. Three possibilities were tried.

1. The default configuration of just using the collection being searched on: e.g,. this year's SDR test collection as recognized by the Dragon speech recognizer. This was the strategy used in the umass-b1.dragon, umass-b2.dragon, umass-r1.dragon, umass-s1.dragon runs.

2. A combination of the SDR recognized collection and a corpus of AP newswire documents produced at the same time as the SDR audio broadcasts were made. This was the approach used in the umass-s2.dragon run.

3. A combination of different versions of the SDR collection each version produced by a different speech recognizer. The Sheffield, Cambridge, Dragon, NIST-B1, NIST-B2, and DERASRU recognizer transcripts were used. It was hoped that the different recognizers would make different recognition mistakes and that by combining the transcripts the mistakes would be less prevalent and, therefore, the expanded query would produce better retrieval. This approach was used in a number of additional runs that were not assessed by TREC.

In experiments on the SDR training set, this final strategy proved to be the most successful of the three. Results on the test set, however, proved to be different.

4.3 Main SDR results

The best retrieval effectiveness on the test collection was found on the umass-s1-dragon run. The average (non-interpolated) precision for this run was 0.5075, which was 89% the effectiveness of the r1 (hand transcribed) reference run: 0.5668. Such a result indicates that retrieval on audio data of the quality used in the SDR collection can be expected to be almost as good as retrieval on a hand transcribed version.

Description	AvgPrec
s1-dragon	0.5075
r1 (LTT)	0.5668

Comparison of the overall "Best, Median, Worst" statistics (compiled by TREC from all of the groups' data) against the s1 result revealed that s1 had the best average precision for 6 of the 23 topics, above median for 13, median for 1 and below the median for 3. That turned out to represent "top" performance among the participating systems.

4.4 Cross-recognizer runs

The CIIR had interpreted the cross-recognizer run description to allow a run such as the final approach listed earlier, where we combined the results of *several* recognizer outputs to help generate higher-quality expansion features. (It does not appear to have worked.) The TREC evaluation process chose to eliminate our run because—although it was a valid interpretation of the request—there were no similar runs to compare it to!

We have expended some effort to run various parts of our query processing across different recognizer outputs to see the impact of differing word error rates on the results. The following table includes a row for each of several recognizer outputs, and the columns represent different amounts of query processing. All numbers are non-interpolated average precision.

Recognizer	WER	SWER	No QP	Basic QP	With LCA
Human (ltt)	–	–	0.4401	0.5240	0.5677
Cambridge (htk)	24.8	24.6	0.4294	0.5001	0.5305
Dragon	29.8	29.9	0.3935	0.4711	0.5089
AT&T-s1	33.1	36.0	0.4197	0.4843	0.5243
Sheffield	36.8	34.1	0.3771	0.4551	0.5061
NIST-B1	34.1	34.7	0.3837	0.4581	0.5066
NIST-B2	46.9	49.2	0.3053	0.3725	0.4193
DERA/SRU-2	61.5	61.7	0.3427	0.4043	0.3858
DERA/SRU	66.2	64.1	0.3053	0.3598	0.4064

In all cases, the same techniques were applied. When LCA was used, it was used on the same database that was being run against. The type of LCA used was as described above (the style submitted for our SDR runs) in all cases, although later experiments showed that a modest improvement was possible.

What is interesting to note is that the effectiveness of retrieval degrades uniformly with increases in WER. We had hoped that either the query processing ("Basic QP") or query expansion ("With LCA") would help compensate for the recognition errors. Unfortunately, the results show that although those techniques are useful in all cases, they degrade at the same rate as the others so appear to be equally impacted by WER.

The one exception to that uniform drop in effectiveness is the AT&T run that had higher WER than Dragon, but also had better effectiveness across the board. We have not investigated this aberration, though hypothesize is may be the result of AT&T's recognizer that worked with larger portions of the word lattice.

4.5 Variants on LCA

We were interested in the effects of varying how we did query expansion. We tried several options, varying the number of expansion features, and varying the way of combining the original query and the expansion concepts. For the latter trials, we used either the method submitted (as described above) or our more standard method (in Section 1.3)—in the former, the expansion features are each treated as equal to original query features; in the latter, the expansion features are treated as a group and balanced against the original query as a group.

First we consider the result of different numbers of expansion concepts. In all cases, the expansion concepts were added to the query as above—that is, the first expansion concept was weighted equal to query features, and the rest had monotonically dropping weights. The runs are done with three different levels of WER to explore the impact of ASR errors on the expansion.

	AvgPrec	R-Prec	Prec@20
Human (LTT)			
10 LCA features	0.5648	0.5081	0.4022
60 LCA features	0.5677	0.5096	0.4087
100 LCA features	0.5645	0.4902	0.3935
Dragon (29.8% WER)			
10 LCA features	0.4923	0.4694	0.3761
60 LCA features	0.5089	0.4894	0.3671
100 LCA features	0.5097	0.4980	0.3804
DERA/SRU (66.2% WER)			
10 LCA features	0.3820	0.4007	0.3239
60 LCA features	0.4064	0.3992	0.3457
100 LCA features	0.3869	0.3948	0.3435

The results suggest that the number of expansion features is not critical, but that too few or too many can slightly damage performance. This supports our typical decision to expand with 50-70 features.

We also investigated the impact of different ways of adding the expansion features to the query. This set of runs was done because the expansion approach used by LCA was not intentional. We tried each of the following approaches. Recall that the query consists of a set of words or phrases, and the expansion features are a similar set of words or phrases.

equal In this run, the query features and the LCA features were all put together and treated equally, except that the LCA features were assigned descending weights starting with 1.0 and working down to 0.01 for the last one. (This was the submitted approach.)

add-group In this run, the LCA expansion features were treated as a single group and that entire group had the same potential contribution as a *single* query feature.

two-groups-Q In this run, the new query consisted of two parts: the original query and the expansion features, each treated as a single group. The query feature group was weighted 1.25 to the 1.0 weight of the LCA query group.

two-groups-LCA This is the same as the *two-groups-Q* run except that the weights were reversed. This is the way that queries were expanded in the ad-hoc track.

We ran these various forms of expansion on several recognizer outputs. In all cases, the queries were expanded with 60 LCA features.

	AvgPrec	R-Prec	Prec@20
Human (LTT)			
equal	0.5677	0.5096	0.4087
add-group	0.5374	0.4963	0.3761
two-groups-Q	0.5523	0.4906	0.4000
two-groups-LCA	0.5662	0.4976	0.4109
Dragon (29.8% WER)			
equal	0.5089	0.4894	0.3761
add-group	0.4823	0.4670	0.3587
two-groups-Q	0.5033	0.4927	0.3783
two-groups-LCA	0.5143	0.5011	0.3935
DERA/SRU (66.2% WER)			
equal	0.4064	0.3992	0.3457
add-group	0.3782	0.3760	0.3261
two-groups-Q	0.3945	0.3776	0.3283
two-groups-LCA	0.3987	0.3851	0.3348

Interestingly, our failing to group the concepts was a mixed blessing. For the human transcripts, it gave us a slight benefit overall, with a negligible drop in high precision. But for our official transcript (Dragon's), we got a slight drop: we would have done about 2% better (and the difference is significant by the sign test, $p = 0.03$). Given that the difference is small, however, we conclude that it did not have much of an impact. We expect to use the *two-groups-LCA* variant in the future, though, since that has been our preferred approach.

4.6 Other results

The s2 run, which used AP news wire as an additional source of LCA expansion terms proved to be worse than s1. The use of other groups transcripts in the LCA process proved to be even more detrimental to retrieval effectiveness than the s2 run. We have tentatively concluded that this approach is not valuable.

5 Very Large Corpus (VLC) Track

Our goals for the Very Large Corpus (VLC) track were modest: To gain experience with a larger corpus, and to contribute to the large-scale corpus-building effort by adding documents to the assessment pool.

Retrieval speed was *not* a research goal. The Inquery system, version 3.1, was used, with no special optimizations.

It was also not a goal to be the most accurate system. The queries created for the Ad-hoc track were used in the VLC track. Ordinarily query expansion with Local Context Analysis (LCA) is database-specific, but in the interests of doing the track with minimal effort, we simply used the query expansion terms created for the Ad-hoc track. Our hypothesis, based on other work with LCA, is that using the VLC database(s) for query expansion would produce more effective results.

5.1 VLC Query Sets

Three sets of queries were developed. All three are of the "Fixed Query" type, because query expansion was done on a separate database (the Ad-hoc database), as described above. The queries were:

inq5vlc1: The INQ501 query set, described above, which used the Title, Description, and Narrative fields;

inq5vlc2: The INQ502 query set, described above, which used the Title, Description, and Narrative fields; and

inq5vlc3: The INQ503 query set, described above, which used the Title and Description fields.

The judged runs were all done on inq5vlc1, because we believed it would be the most accurate query set.

5.2 Indices

The BASE1 collection was indexed as a single database.

Various data buffer size and index word size constraints prevented building the larger collections into a single database (in particular, the VLC2 collection contains more tokens than can be counted in a 32 bit word). Although these constraints could have been overcome, a multi-database approach fit more naturally with our other research interests, and was hence the approach adopted.

The BASE10 collection was indexed as two databases. Each indexed used its own *idf* values, i.e., global *idf* values were *not* created. Although this makes document scores slightly incompatible, it was assumed that queries would be long enough to minimize problems.

The VLC2 collection was indexed as 16 databases. As with the BASE10 collection, each index used its own *idf* values, i.e., global *idf* values were *not* created.

Indexing was done on a multi-user system during ordinary daily use, so little can be said about Inquery's indexing speed or resource requirements. It was not practical to devote one or more computers to the VLC effort; nor was it necessary, as Inquery is quite capable of handling this volume of data on a machine being used by a number of other processes.

VLC2 indexing, from initiation to completion, took 43 hours and 36 minutes, with 4 processors busy for almost all that time, and some competition from other users and tasks. About 8 hours and 9 minutes of that time was spent uncompressing data. Different parts and stages of the build were running in parallel on the 3 processors of the computer, with an attempt to keep all processors busy without thrashing. Because of limited disk space, uncompressed versions of bundles were created as needed and then deleted immediately after use. All of the processes used were "niced". The computer used was a Sun Sparc server with 4 processors each running at 167 MHz and 1000 MB of memory.

The BASE10 build required 26:09 on the same processors but was overlapped with the VLC2 build for 14:30 of those hours.

The BASE1 build took 2:40 including re-compressing the source collection (the other builds made uncompressed copies and removed them; this build actually re-compressed the bundles after using them, taking much longer).

The indices required 0.68 gigabytes for BASE1, 6 gigabytes for BASE10, and 53 gigabytes for VLC2.

5.3 Retrieval

The BASE1 collection was organized as a single index. Queries were run against that index.

The BASE10 and VLC2 collections were organized as multiple indices. Queries were run against *all* indices associated with a collection. Document rankings from each index were merged to produce a final ranking for the collection.

The approach to merging document rankings was quite simple. The top-ranked 20 documents from each index were candidates for the final result set. These 20-document rankings were merged based upon the document scores. No attempt was made to normalize the scores returned by each index, or to favor documents returned by the "better matching" index.

Retrieval was done on a multi-user system during ordinary daily use, so little can be said about Inquery's retrieval speed or resource requirements. Inquery is clearly not one of the faster systems doing the VLC track. Based upon last year's results, it is also likely that Inquery uses longer queries than most of the other groups. The timing figures shown below are 'wall-clock' times.

	inq5vlc1	inq5vlc2	inq5vlc3
BASE1	7 min	8 min	6 min
BASE10	71 min	74 min	45 min
VLC2	593 min	598 min	438 min

The retrieval results appear to have been quite good. Precision at 20 documents is shown below.

	inq5vlc1	inq5vlc2	inq5vlc3
BASE1	0.202	0.204	0.208
BASE10	0.429	0.441	0.419
VLC2	0.625	0.624	0.598

We haven't done any interesting post-hoc analysis at this point.

Acknowledgments

We thank Daniella Malin, Michael Scudder, Fang-fang Feng, Chiinga Musonda, and Karen Priore of the CIIR for their assistance in the work described here.

This study is based on research support by several grants: the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623; the National Science Foundation under grant number IRI-9619117; and the NSF Center for Intelligent Information Retrieval at the University of Massachusetts, Amherst.

Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsors.

References

[1] J. Allan. Incremental relevance feedback. In *Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval*, pages 270–278, Zurich, 1996. Association for Computing Machinery.

[2] J. Allan, J. P. Callan, W. B. Croft, L. Ballesteros, D. Byrd, R. Swan, and J. Xu. INQUERY does battle with TREC-6. In D. K. Harman and E. M. Voorhees, editors, *The Sixth Text REtrieval Conference (TREC-6)*. National Institute of Standards and Technology, Special Publication, 1998.

[3] James Allan, Jamie Callan, W. Bruce Croft, Lisa Ballesteros, Don Byrd, Russell Swan, and Jinxi Xu. INQUERY does battle with TREC-6. In D. Harman, editor, *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*. National Institute of Standards and Technology Special Publication, (in press).

[4] J. Callan. Learning while filtering documents. In *Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval*, Melbourne, 1998. Association for Computing Machinery.

[5] J. P. Callan. Document filtering with inference networks. In *Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval*, pages 262–269, Zurich, 1996. Association for Computing Machinery.

[6] J. P. Callan, W. B. Croft, and S. M. Harding. The INQUERY retrieval system. In *Proceedings of the Third International Conference on Database and Expert Systems Applications*, pages 78–83, Valencia, Spain, 1992. Springer-Verlag.

[7] B. Peskin et al. Progress in recognizing conversational telephone speech. In *Proc. ICASSP-97*, Munich, April 1997.

[8] M.J. Hunt et al. An investigation of plp and imelda acoustica representations and of their potential for combination. In *Proc. ICASSP-91*, Toronto, May 1991.

[9] P. Woodland et al. Broadcast news transcription using htk. In *Proc. DARPA Speech Recognition Workshop*, Chantilly, February 1997.

[10] R. Roth et al. Dragon systems' 1994 large vocabulary continuous speech recognizer. In *Proc. ARPA Spoken Language Systems Technology Workshop*, Austin, 1995.

[11] S. Wegmann et al. Dragon systems' 1997 broadcast news transcription system. In *Proc. of the Broadcast News Transcription and Understanding Workshop*, Lansdowne, VA, February 1998.

[12] S. Wegmann et al. Dragon systems' automatic transcription system for the new tdt corpus. In *Proc. of the First International Conference on Language Resources and Evaluation*, Granada, Spain, May 1998.

[13] T. Anastasakos et al. A compact model for speaker-adaptive training. In *Proc. ICSLP'96*, Philadelphia, October 1996.

[14] H. Hermansky. Perceptual linear prediction (plp) analysis for speech. *J. Acoust. Soc. Amer.*, 87:1738–1752, 1990.

[15] C.J. Leggetter and P.C. Woodland. Speaker adaptation of continuous density hmms using multivariate linear regression. In *Proc. ICSLP'94*, Yokohama, September 1994.

[16] V. Nagesha and L. Gillick. Studies in transformation based adaptation. In *Proc. ICASSP-97*, Munich, April 1997.

[17] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval*, pages 4–11, Zurich, 1996. Association for Computing Machinery.

Natural Language Information Retrieval: TREC-7 Report

Tomek Strzalkowski, Gees Stein, G. Bowden Wise
GE Research & Development
Jose Perez-Carballo
Rutgers University
Pasi Tapanainen, Timo Jarvinen, Atro Voutilainen
University of Helsinki
Jussi Karlgren
Swedish Institute of Computer Science

Team id: GERSH

1. Summary

The GE/Rutgers/SICS/Helsinki team has performed runs in the main ad-hoc task. All submissions are NLP-assisted retrieval. We used two retrieval engines: SMART and InQuery built into the stream model architecture where each stream represents an alternative text indexing method.

The processing of TREC data was performed at Helsinki using the commercial Functional Dependency Grammar (FDG) text processing toolkit. Six linguistic streams have been produced, described below. Processed text streams were sent via ftp to Rutgers for indexing. Indexing was done using Inquery system. Additionally, 4 steams produced by GE NLToolset for TREC-6 were reused in SMART indexing.

Adhoc topics were processed at GE using both automatic and manual topic expansion. We used the interactive Query Expansion Tool to expand topics with automatically generated summaries of top 30 documents retrieved by the original topic. Manual intervention was restricted to accept/reject decisions on summaries. We observed time limit of 10 minutes per topic. Automatic topics expansion was done by replacing human summary selection by an automatic procedure, which accepted only the summaries that obtained sufficiently high scores. Two sets of expanded topics (automatic and manual) were sent to Helsinki for NL processing, and then on to Rutgers for retrieval. Rankings were obtained from each stream index and then merged using a combined strategy developed at GE and SICS.

2. Background

The work reported here was part of the Natural Language Information Retrieval project (NLIR) (Strzalkowski et al., 1997; Strzalkowski, 1995). One of the thrusts of this project has been to demonstrate that robust NLP techniques can help to derive better representation of text documents for indexing and search purposes than any simple word and string-based methods commonly used in statistical full-text retrieval. This was based on the premise that linguistic processing can uncover certain critical semantic aspects of document content, something that simple word counting cannot do, thus leading to more accurate representation. We demonstrated that NLP can be done efficiently on a very large scale, and that it can have a significant impact on IR. At the same time, it became clear that exploiting the full potential of linguistic processing is harder than originally anticipated. In particular, simple linguistically motivated indexing (LMI) techniques turned out to be no more effective than well-executed statistical approaches, while more advanced NLP techniques, such as concept extraction, remained too expensive for large-scale applications (Sparck-Jones, 1999).

Given this state of affairs, we went on to investigate specific conditions under which LMI could be more beneficial. For example, we have noticed that the amount of improvement in recall and precision which we could attribute to NLP, appeared to be related to the type and length of the initial search request. Longer, more detailed topic statements responded well to LMI, while terse one-sentence search directives showed little improvement. This is not particularly surprising considering that the shorter queries either

contain a handful of highly discriminating terms or are deliberately vague. On the other hand, detailed statements are more typical for situations where the user is uncertain of how to succinctly express the query. In such cases linguistic processing helps to sharpen the query thus making it more effective.

We adopted the *topic expansion* approach in which the original topic is expanded using passages selected from sample retrieved documents. The intent was to expand the initial search specifications in order to cover their various angles, aspects and contexts. Based on the observations that NLP is more effective with highly descriptive queries, we designed an expansion method in which passages from related, though not necessarily relevant documents were imported into the user queries. This method produced a fairly dramatic improvement in the performance of several different statistical search engines that we tested boosting the average precision by anywhere from 40% to as much as 130%. Therefore, we concluded that topic expansion appears to lead to a genuine, sustainable advance in IR effectiveness. Moreover, we showed at TREC-7 that this process can be automated while maintaining at least some of performance gains.

It is not difficult to see why topic expansion, when done properly, works this well. As the expansion progresses, the expanded search topic takes the form of an extended brief on that topic, a meta-document that contains the information that the user seeks. In other words, as we keep improving the query to get more relevant documents, we are in effect forming an answer. The expanded topic is indeed close to an answer, as demonstrated by the following example:

> ORIGINAL TOPIC (Topic 362 Description):
> *Identify incidents of human smuggling.*
>
> EXPANDED TOPIC:
> *Federal immigration agents arrested 31 illegal aliens at Los Angeles International Airport overnight, bringing to more than 200 the number of people nabbed in a nationwide crackdown on high-altitude "people smuggling," authorities said today.*
>
> *A federal grand jury indicted a Carlsbad motel operator, five Los Angeles men and a Mexican national on charges of running an alien-smuggling ring that whisked about 600 people per month to Santa Ana and Los Angeles.*
>
> *INS officials described Eastern's flight, which departs daily from LAX at 10:50 p.m., as an unwitting conduit in a massive transcontinental smuggling operation that apparently moved several thousand illegals out of Los Angeles in the last month alone. Some of the aliens arrested said they paid as much as $4,000 for a package deal, transportation from home, housing, the Eastern plane ticket and a job in New York.*
>
> *Federal agents took 60 illegal Chinese aliens into custody in southern Alabama and announced the arrest of the alleged ringleader of a smuggling operation that planned to bring 35,000 more into the United States. Officials from the Immigration and Naturalization Service and the U.S. Customs Service said the aliens were caught as they arrived by plane from Panama in Fairhope, Ala. The ring planned to bring in Chinese nationals from Panama and Bolivia in an operation believed to be run by persons linked to former Panamanian dictator Manuel A. Noriega.*
>
> *U.S. Border Patrol agents intercepted a tractor-trailer rig late Tuesday packed with 105 illegal aliens on Interstate 15 near Rancho Bernardo, authorities said. Agents stopped the truck about 11:30 p.m. and arrested the driver, Frank Ellinger, 45, of National City on suspicion of smuggling aliens, patrol spokesman Michael Gregg said.*

The reader may note that this expanded topic, reads a bit like the *News from Every State* column in *USA Today*. It is in effect a (likely incomplete) brief on a single subject. It its thus more than just an expanded

search topic, and represents an important step towards a new kind of information retrieval where the information, not the document containing it, becomes the target.[1]

The example shown above has been obtained through a human-assisted interactive topic expansion process explained in more details below. In a fully automated expansion, where NLP techniques replace human judgments, the results are not nearly as good as yet. Thus far we have used only very simple linguistic tools (i.e., those suitable for high-volume IR applications) to assist automatic expansion, but we see this area as ripe for more advanced processing techniques, including entity and event extraction, coreference and cross-reference techniques, etc.

3. Ad-Hoc submissions

In TREC-7 we participated in the ad-hoc track only. Below are short descriptions of official runs.

3.1. Summarization-based manually-assisted topic expansion

This was a multi-stream run using InQuery against the manually expanded topics. Summaries used in expansion were derived from top-ranked documents retrieved by SMART using the initial topics (title+description only). The key characteristics of this run is the 10 minute time limit imposed on topic expansion. All expansion has been performed via the Query Expansion Tool interface (QET) which allows the user to view only the summaries of top retrieved documents, and select or deselect them for topic expansion. By default, summaries of all top 30 documents were used for expansion unless the user manually deselected some (this was precisely the only form of manual intervention allowed.)

We observed that for many queries 2 interactions were possible within the 10 minute interval. The first interaction (submit original query, wait for result, get 30 summaries, review & deselect summaries, and commit the selections) would take typically 4-6 minutes. In the second interactions, only the new documents retrieved in top 30 ranks (if any) were considered, therefore usually 3-4 minutes were sufficient. The target of expansion was to get between 5 and 10 "relevant" summaries within the allotted time. If this was achieved within the first interaction, no further search was performed. Otherwise, the second interaction was attempted if at least 3 minutes remained. This 6-4 split was determined in dry-run trials with TREC-6 queries.

The topic expansion interaction proceeds as follows:

1. The initial natural language topic statement is submitted to a standard retrieval engine via a Query Expansion Tool (QET) interface. The statement is converted into an internal search query and run against the database.
2. The system returns topic-related summaries of top N (=30) documents that match the search query.
3. The user reviews the summaries (approx. 5-15 seconds per summary) and *de-selects* these that are *not* relevant. For TREC-7 evaluations, we set time limit of 10 minutes per query (clock time).
4. All remaining summaries are automatically attached to the search topic.
5. The expanded topic is passed through a series of natural language indexing steps and then submitted for the final retrieval.

3.2. Summarization-based automatic topic expansion with InQuery

This was a single-stream automatic run using InQuery against the automatically expanded topics. Plain stems stream and syntactic noun phrase stream were combined and converted into a single InQuery-syntax representation. Again, the expanded topics were generated using summaries obtained from SMART-retrieved documents. The original un-expanded short topics (title+description only) were submitted to

[1] This approach bears only superficial similarity to passage retrieval used in standard IR (Callan, 1994; Kwok et al., 1993). In passage retrieval fixed-size segments are weighted against the search query which is helpful in assessing relevance of longer documents. However, no attempt is made at extracting coherent "stories".

SMART (version. 11) in stems-stream mode, and the top 100 returned documents were retained. These documents were automatically summarized with GE Summarizer using topic title as to obtain a 5% topical indicative summaries.

Summaries were selected for expansion if they had a sufficient level of "overlap" with the original search topic. The "overlap" score was determined by the number of shared terms, as well as the "locality" of the summary. In this experiment we required that there was at least 60% overlap on the content terms between the summary and the original topic. In addition, multi-paragraph summaries were required for each paragraph to have at least 40% overlap with the topic, except for the blocks of consecutive paragraphs.

These selection criteria are fairly simplistic and tests performed with TREC-6 data were generally inconclusive as to their effectiveness. This is because term overlap is not a good indication of relevance (we know that!). Moreover, the goal of expansion was to add new terms to the topic, not just more of the same, thus a too-high degree of overlap would not be of much use.

3.3. Summarization-based automatic topic expansion with SMART

This was a multi-stream automatic run produced using SMART rather than InQuery. Automatically expanded queries were NL processed using GE NLToolset and run against the 4-stream index originally produced for TREC-6. Streams were merged using the same procedure that was developed for TREC-6.

4. Helsinki's NLP System overview

We used Helsinki's Functional Dependency Grammar (FDG) includes the EngCG-2 tagger and dependency syntax which links phrase heads to their modifiers and verbs to their complements and adjuncts. FDG was applied to the whole corpus, with the output passed to the stream extractor. The streams were generated as follows:

4.1. Simple Streams

0. *stem*: just stemmed words, stopwords removed.
1. *name*: all proper names
2. *aan*: simple noun phrases with attributes. Basically adjective-noun sequences minus some exceptions.

4.2. Direct Dependency Streams

3. *sv*: subject-verb pairs where the subject is a noun phrase.
4. *vo*: verb-complement pairs. The complement includes objects and some object-like adverbial classes.

4.3. Indirect Dependency Streams

5. *nofn*: "N1 ... of ... N2" pairs, where N1 and N2 are heads of simple noun phrases.
6. *sc*: subject-complement pairs where the complement modifies the subject (flowers grow wild => wild+flower).

5. Details of Helsinki's FDG System

5.1. Functional Dependency Grammar

The Functional Dependency Grammar (FDG) parser (Jarvinen and Tapanainen, 1997; 1998; Tapanainen and Jarvinen, 1997) produces surface-syntactic analyses for sentences in terms of explicit dependency structures. These structures are trees where the words correspond to the nodes.

In principle, each word in a sentence is connected to an unique head by a labeled arc, though also partial analyses for complex sentences are allowed. The labels refer to syntactic functions such as subject, object, and so on. The highest node is connected to an external root.

A simplified example in Table 1 shows the analysis of the sentence *I tamed a bird*. The arc between *I* and *tamed* denotes that *I* is the modifier of *tamed* and its syntactic function is that of subject. Similarly, *a* modifies *bird*, and it is a determiner.

The text format of the analysis in Figure 1 shows the functional labels with a numeric pointer to the head, and some additional information produced by the parser. The third column shows the base form of the word, and the last column contains the word-class information. Due to the strong correlation of the syntactic analysis produced by the FDG to the semantic relations, the output is usable to tasks where semantic rather than syntactic information is required.

We use the FDG output to collect pairs of words that were connected by certain syntactic relations. For example, if we excerpt words connected by the object relation, the analysis in Figure 1 produces the normalised string "tame bird".

Table 1. Sample analysis

1 I i	2 tamed tame	3 a a	4 bird bird
subj:>2 PRON	main:>0 V	det:>4 DET	obj:>2 N

5.2. Morphological analysis and lemmatization

The adjectives and nouns were returned to their morphological baseforms. The participial adjectives and nouns are returned to the verbal form (e.g. growing economy → grow+economy) which makes them equivalent to the verbal usage (e.g. the economy grows → grow+economy).

5.3. Names

The name recognizer, based on the Conexor Name Recognizer (at www.conexor.fi), identifies "named entities" consisting of one or more words. Typically, names are nominal heads written in the upper case, with any number of pre-modifiers. Also coordinations and certain types of post-modification (e.g. post-modifying PPs) are recognized as legitimate parts of names, e.g. "Procter & Gamble"; the "City of London". We do not regard titles as names (though they certainly are useful clues for identifying names).

In our system, names are identified on the basis of three types of information: lexical, orthographical and grammatical. This information is used on the basis of hand-written linguistic rules. The name recognizer is reasonably fast; on a mid-range Pentium PC running Linux, it processes well above 2,000 words per second. At present, our name recognizer performs no sub-classification. The ability to identify e.g. persons, organizations and locations remains to be added in the program. No rigorous evaluation of the name recognizer has been carried out. Our hunch is that well over 90% of names occurring in many types of English text (at least journalistic, fiction and scientific texts) are recognized.

5.4. Noun phrase streams

The simple noun phrases with attributives are collected to one stream. The syntactic position in the sentence was used to filter the noun phrases. Adverbs of time (e.g. "tomorrow night") and other generic adverbs (e.g. *Emissaries returned *home**) were excluded by using the syntactic function given by FDG. We did not apply stop word lists here. The *of*-genetive streams are represented through the head words of noun phrases. For instance, the noun phrase *[large burlap sacks] of [the imported material]* contains two noun phrases. The head words of both are collected into the stream: *material+sack*. Word-class information is used for filtering out undesired candidates for the streams.

5.5. Valency streams

Also verbs are excerpted together with their various dependents. There are two classes: subject-verb and verb-complement pairs. The latter include predicatives, direct objects and other object-like adverbials.

Many of the subject and object types are filtered out by using word-class labels and various heuristics to exclude non-nominal elements.

5.6. Indirect dependency

Sometimes the connecting information is obtained indirectly using the syntactic functions in the FDG output. A typical pair is a subject and its complement e.g. {\em flowers grow wild => wild+flower}.

5.7. An example: text and streams

"Gardening: The perennial pleasures of spring- Robin Lane Fox prepares to strike an economic blow for a better garden on a shoestring.

BEFORE LONG, better weather ought to have caused gardeners' sap to rise: act now while enthusiasm is fresh and strike an economic blow for a better garden on a shoestring.

Seeds are no longer as cheap as they were and, admittedly, I sometimes grow them for the hell of it, just to see if I can make them come up.

It is no longer time to postpone the plunge, but the first seeds to go in are not the most obvious.

It is still too early to be sowing tobacco plants, cosmos daisies and all the mainstays of summer bedding which grow quickly and will be too far advanced if started before March.

Perennial flowers are another matter.

Among these early sowers, I am casting my net more widely and am being prompt with less-familiar perennials which ought to flower from July onwards.

Geraniums are obvious candidates, especially now that so many colours have been selected and bred for seed-raising: even for amateurs, cuttings are almost a matter of the past.

I leave most of the geraniums to others, but carnations are another matter.

Not long ago, I was editing Vita Sackville-West's old gardening columns when I was carried away by her description of the Chabaud strain of carnation.

Their colours, she felt, had the quality of a Van Gogh painting- I remember that she described some of them as bistre.

On the spur of a good read, I tried to grow my own, but started too late.

From a sowing in mid-March, I had none of her fancies, no bistred beauties or blooms of old blood-red.

The wretched plants never flowered at all.

Once bitten, never shy: you know the gardening instinct.

So, this year I am starting Chabaud carnations from seed in the first week of February.

Somewhere in Britain, people must still grow them happily because garden centres stock them on open shelves in their standard ranges of seed from Suttons or Thompson and Morgan.

The seed will germinate in the usual amateur's pot, filled with a standard seed compost and covered with a tight stretch of cling-film to retain the heat and sweat.

Chabaud carnations like heat in order to spring into growth.

They will germinate in a warm cupboard, below the spare bath towels, if you remember to retrieve them and roll back the cling-film at the first signs of emergent shoots....."

5.7.1. Noun phrase stream

perennial+pleasure spring robin+lane+fox economic+blow good+garden shoestring good+weather gardener+sap enthusiasm economic+blow good+garden shoestring seed hell time plunge seed tobacco+plant cosmos+daisy mainstay summer+bed

5.7.2. Of-genitive stream

spring+pleasure bed+mainstay past+matter geranium+many strain+description carnation+strain paint+quality read+spur fancy+none blood-red+bloom February+week seed+range cling-film+stretch shoot+sign seed+list success+three Shirley+butcher stockist+list flower+mass pink+touch flower+variety

5.7.3. Name stream

robin+lane+fox Vita+SackvilleWest chabaud van+Gogh midMarch chabaud Britain Sutton Thompson Morgan Chiltern+seed Cumbria northwest+England gaura+lindheimeri Chiltern+seed butcher+of+Shirley Croydon south+London snowcloud gaura

5.7.4. Verb-object stream

strike+blow strike+blow postpone+plunge sow+plant cast+net edit+column have+quality grow+own start+late know+instinct start+carnation retain+heat have+day sell+mixture join+list reach+ft equal+gaura give+mass have+habit have+hybrid catch+mood

5.7.5. Subject-verb stream

fox+prepare sap+rise colour+select plant+flower people+grow centre+stock seed+germinate sowing+owe seedsman+sell seed+join border+need dozen+come nursery+charge balloon+show

5.7.6. Subject-complement stream

fresh+enthusiasm flower+matter geranium+candidate carnation+matter available+catalogue white+valerian valerian+plant confuse+white variety+valerian wild+flower good+success name+platycodon good+form name+blue

6. Stream Merging

Our goal was to merge results from searches over several stream indexes built over the same data set. The merging program takes the ranked lists of documents obtained from each stream, and produces a single unified ranking. As in the past, we aimed at obtaining a better ranking than any single stream search could produce.

6.1. What we know from past experience

We can draw on experience from past searches to estimate behavior in future ones. The parameters we have recourse to are

- past average precision for the respective streams;
- how these measures vary at different combinations of ranges of ranks and scores; and
- consistency in behavior - how much the precision measures and the overlap fluctuate from query to query.

The observable measures we can make use of at merge time are for each document its rank and relative score for each stream. The parameters we have chosen to disregard the consistency measure after some not very systematic measurements indicating the fluctuation is small for TREC data, and fold in the total average precision with the others in a matrix of estimated precision.

6.2. Example data

Here is an example: the table shows the average precision at various ranks for thirty TREC 6 queries designated as training material. This tells us that documents that are ranked between 0 and 10 for both streams (stems on the y axis and pairs on the x axis) have an average precision of 0.482; documents that are ranked between 26 and 100 in the stems stream and top 10 by the pairs stream have an average precision of 0.289; documents that are ranked between 26 and 100 in the pairs stream but not ranked at all by the stems stream have an average precision of 0.046.

	0	10	25	100	200	1000
0	0.000	0.077	0.066	0.046	0.022	0.008
10	0.241	0.482	0.262	0.400	0.250	0.270
25	0.228	0.303	0.318	0.226	0.094	0.183
100	0.105	0.289	0.303	0.192	0.206	0.146
200	0.066	0.100	0.211	0.163	0.173	0.130
1000	0.026	0.124	0.158	0.150	0.157	0.095

6.3. Theoretical issues - distribution

This distribution ideally should be modeled by some useful function of two or three parameters which could be used for an estimate of future behavior: something like

prec(rankA,rankB,overlap(A,B))

which could be trained by using data such as past overlaps - relevant and non-relevant - at various ranks. Until we come up with something like it, we can use the matrix directly as an estimate.

6.4. Merging - using the information

The simplest merging approach would be to use the precision estimates as probability estimates directly. Thus, in the example, first pick all documents that are shared in the top ten, thereafter the documents that are in the top ten for the stems stream but rank 26-100 in the pairs stream, thereafter documents that have ranks 11-25 in both streams, and so forth. As is obvious, the training data are insufficient: the matrix cell values should be expected to grow monotonically from the 10-10 origin outwards, or possibly to retreat eventually, not to waver like the ones in the example.

6.5. Experiment

The approach is easy enough to test. First, however, the order of documents within cells must be determined. Below are a couple of experiments. The result hinges on how the cell ranges are chosen. Choose the intervals too small, and overfitting will occur; too large, and no learning will take place.

Also, we must determine if the rank or the relative score is a better measure of document relevance. Relative scores gave somewhat better performance in our experiments, but are dependent on the implementation of the stream, and thus somewhat less convincing in the general case. We find that a slight improvement indeed occurs in the test data. However, the approach is vulnerable to overfitting, and needs to be tested on more material to be useful.

Also, an noticeable problem was the number of non-judged documents in the collection. The standard trec_eval script judges non-judged documents to be one hundred per cent non-relevant, which seems overly pessimistic.

6.6. Some further observations

In our experiments we choose to model past relative performance by the TREC 11-point average precision, but take into account the precision distribution over retrieval scores. Average precision and recall is arguably the most important measure. If we know from historical data that a certain stream produces consistently excellent results and another consistently low-grade results, we should weight them in proportion to that performance.

We ran the results stream by stream, and built seven-way relative average precision matrices, where each cell represented a score range within the seven participating streams. We then merged the streams, picking the cells with highest average precision in the training data first. For example, if the stream aan scores a document more than 0.926 and it is not ranked in any other stream it has only 0.101 likelihood of being

relevant; if it is scored more than 0.942 by stream aan and more than 0.935 by the plain stems stream it has 0.318 likelihood of being relevant. Not a surprising result.

The experiment went well, and we tuned a number of parameters - score ranges, cell sizes, etc. - to produce usefully improved results, but for the main run, we found that reworking the entire processing chain gave us too little training data for the algorithm to produce stable results. In the end we went back to the tried hand-worked scheme of previous years.

7. Stylistics experiments

This year, as previous years, we ran several experiments to predict relevance of an item from non-topical features: stylistic features which in other experiments have predicted the genre of texts with reasonable accuracy, and which seem to give significant correlation with relevance using standard statistical measures (e.g. Karlgren, 1996). However, the predictive power is too weak for rule generation, and several important assumptions - such as requirements of normal distribution of variables - underlying standard multivariate categorization metrics are not met. We are currently performing a series of experiments using machine learning techniques.

8. Preliminary Results Analysis

We continue to analyze the results. The preliminary examination indicates that the merging system did not work as we anticipated. The problem may not be with the merging itself, rather with the way streams were defined in TREC-7. In contrast with TREC-6, we decided to make more fine-grained distinctions between various text representations, resulting in many "thin streams", i.e., streams that retrieve few documents based on very limited information, although highly specialized features. This, we believe, created ranked lists that were far less reliable than with "fatter" streams, i.e., the score differentials due to content were too small to be reliably distinguished from noise. Therefore, any merging system using these ranks would produce unreliable results. This is indeed our experience this year. While the main unmerged stream runs performed quite well, all merged runs did poorly.

The table below summarizes the "unofficial" results obtained with the expanded topics, before any NLP indexing and stream merging took place. The results correspond to NLIR "stems stream", the basic word-based stream. The reader should note the performance increase. These results are significantly better than any merged runs officially submitted.

queries	*original T+D*	*long T+D+N*	*expanded automatic*	*expanded interactive*
SYSTEM	RU-INQUERY	RU-INQUERY	RU-INQUERY	RU-INQUERY
PRECISION AVERAGES				
11pt Average	0.1692	0.2036	0.2019	0.2932
%change		*+20.0*	*+20.0*	*+73.0*
At 10 docs	0.4620	0.5000	0.4120	0.6140
%change		*+8.0*	*-11.0*	*+33.0*
At 30 docs	0.3153	0.3587	0.3013	0.4327
%change		*+14.0*	*-3.0*	*+37.0*
At 100 doc	0.1756	0.2068	0.1922	0.2668
%change		*+18.0*	*+9.0*	*+52.0*
Recall	0.44	0.51	0.46	0.69
%change		*+16.0*	*+5.0*	*+57.0*

In automatic expansion we observed a good precision increase (about 20%) over the unexpanded topics. This is encouraging, but not nearly as effective as in manual expansion where we noted 73% increase.

Still, however unsophisticated, the automated expansion did produce an increase nearly identical to what is attributed to the *narrative* field in the topics.

Acknowledgments

The work reported here was supported in part by the Defense Advanced Research Projects Agency under the Tipster III contract 97-F157200-000 through the Office of Research and Development.

References

Callan, Jamie. 1994. "Passage-Level Evidence in Document Retrieval." Proceedings of ACM SIGIR'94. pp. 302-310.

Jarvinen, T., and Tapanainen, P. 1997. "A dependency parser for English." Tech. Rep. TR-1, Department of General Linguistics, University of Helsinki, Finland.

Jarvinen, T., and Tapanainen, P. 1998. "Towards an implementable dependency grammar." In Processing of Dependency-Based Grammars. Montreal, Canada. S. Kahane and A. Polguere, Eds., COLING-ACL'98, Association for Computational Linguistics, Universite de Montreal, pp. 1-10.

Kwok, K.L., L. Papadopoulos and Kathy Y.Y. Kwan. 1993. "Retrieval Experiments with a Large Collection using PIRCS." Proceedings of TREC-1 conference, NIST special publication 500-207, pp. 153-172.

Sparck-Jones, Karen. 1999. "What Is The Role for NLP in Text Retrieval". In T. Strzalkowski (ed.) Natural Language Information Retrieval. Kluwer. pp. 1-25.

Strzalkowski, Tomek, Louise Guthrie, Jussi Karlgren, Jim Leistensnider, Fang Lin, Jose Perez-Carballo, Troy Straszheim, Jin Wang, and Jon Wilding. 1997. "Natural Language Information Retrieval: TREC-5 Report." Proceedings of TREC-5 conference.

Strzalkowski, Tomek. 1995. "Natural Language Information Retrieval." Information Processing and Management, Vol. 31, No. 3, pp. 397-417. Pergamon/Elsevier.

Strzalkowski, Tomek, Fang Lin, Jose Perez-Carballo, and Jin Wang. 1997. "Natural Language Information Retrieval: TREC-6 Report." Proceedings of TREC-6 conference.

Tapanainen, P., and Jarvinen, T. 1997. "A non-projective dependency parser." In Proceedings of the 5th Conference on Applied Natural Language Processing, Washington, D.C. Association for Computational Linguistics, pp. 64-71.

Twenty-One at TREC-7: Ad-hoc and Cross-language track

Djoerd Hiemstra
University of Twente, CTIT
P.O. Box 217, 7500 AE Enschede
The Netherlands
hiemstra@cs.utwente.nl
http://www.cs.utwente.nl/~hiemstra/

Wessel Kraaij
TNO-TPD
P.O. Box 155, 2600 AD Delft
The Netherlands
kraaij@tpd.tno.nl
http://www.tpd.tno.nl/TPD/smartsite304.html

Abstract

This paper describes the official runs of the Twenty-One group for TREC-7. The Twenty-One group participated in the ad-hoc and the cross-language track and made the following accomplishments: We developed a new weighting algorithm, which outperforms the popular Cornell version of BM25 on the ad-hoc collection. For the CLIR task we developed a fuzzy matching algorithm to recover from missing translations and spelling variants of proper names. Also for CLIR we investigated translation strategies that make extensive use of information from our dictionaries by identifying preferred translations, main translations and synonym translations, by defining weights of possible translations and by experimenting with probabilistic boolean matching strategies.

1 Introduction

Twenty-One is a 2 MECU project with 12 partners funded by the EU Telematics programme, sector Information Engineering. The project subtitle is "Development of a Multimedia Information Transaction and Dissemination Tool". Twenty-One started early 1996 and is currently in its evaluation phase. The TREC ad-hoc and CLIR tasks fit our needs to evaluate the system on the aspects of monolingual and cross-language retrieval performance. Partners in Twenty-One are: Getronics Software, TNO-TPD[1], DFKI GmbH, Xerox Research Center Europe, Highland Software, University of Twente, University of Tübingen, MOOÏ foundation, Environ, Climate Alliance, VODO and Friends of the Earth.

1.1 Cross-language retrieval in Twenty-One

The Twenty-One database consists of documents in different languages, initially Dutch, English, French and German but extensions to other European languages are envisaged. The primary approach to CLIR in Twenty-One is Document Translation (DT). There are certain advantages and disadvantages to DT:

- DT reduces the cross-language retrieval task to a monolingual search task
- The quality of a translation can in principle be better because the full document context is available. In the case of query translation there is often very little context.

[1]TNO participated also in the SDR and filtering tracks, cf. "TNO TREC7 site report: SDR and filtering", elsewhere in this volume

- DT is slow but can be done off-line, therefore the time constraints are less severe.
- DT requires a full translation of the document base for each supported language, which makes it not really scalable.

The DT approach in Twenty-One can be supplemented with or replaced by query translation depending an the application and collection size. A more elaborate description can be found in [7].

1.2 The Twenty-One retrieval system

The Twenty-One demonstrator system is on-line since March 1998.[2] It is based on two types of indexes:

1. a fuzzy phrase index based on n-gram search on phrases,
2. a standard Vector Space Model (VSM) index based on lemmas.

The first index type is well suited for short queries and interactive query refinement, whereas the VSM index is better suited for longer queries. Before a document is processed by the Twenty-One system it goes through a number of NLP preprocessing steps.

1. The language of the document is identified.
2. The document is translated to the other supported languages using the Logos machine translation system. A copy of the translated document is kept in the database to give a high quality preview of the document in the users preferred language.
3. The original document and the machine translated documents are tokenised, part-of-speech disambiguated and lemmatised using the morphological tools of Xerox. The lemmas are used to build the vector space index.
4. Part-of-speech tags are used to extract noun phrases from the documents using the TNO parser. Noun phrases are used to build the fuzzy phrase index.
5. For language pairs not supported by Logos, the lemmatised and noun phrase bracketed document is translated using the VLIS lexical database of Van Dale Lexicography. Translations are used to build the fuzzy index and the VSM index for the other languages. Again a copy is kept in the database for previewing in the user-preferred language.

1.3 Twenty-One in TREC

For the TREC-7 evaluations we did not use the Twenty-One system in the setup mentioned above, primarily because of the size of the CLIR document collection. Instead we followed a query translation approach. Because of its modular design we were able to build versions of the Twenty-One system that are specifically suited for the TREC tasks.

One of our main goals after the first TREC-participation in TREC-6 [8] was to upgrade the monolingual performance of our system to a level that is comparable with groups already participating in TREC. To test the baseline performance of our system we entered the ad-hoc task with a simple version of the Twenty-One system, only using the TNO VSM engine and Porters stemmer for English. For the CLIR task we were able to use most of the Twenty-One modules. We used

[2] http://twentyone.tpd.tno.nl/

the VSM engine for retrieval, but also the fuzzy index to recover from missing translations and spelling variants of proper names. As a preprocessing step before indexing and translation we used the Xerox morphological tools. The VLIS lexical database of Van Dale Lexicography was used for the translation of the queries.

This paper is organised as follows. In section 2 we describe our work on the ad-hoc task. Section 3 describes our work on the CLIR task. Section 4 will give concluding remarks.

2 The ad-hoc task

The main contribution of Twenty-One to the TREC ad-hoc task is an experiment with a new weighting algorithm. The weighting algorithm was developed from scratch within the linguistically motivated probabilistic model of information retrieval [2]. The model uses language models and techniques that find their origin in the field of statistical natural language processing, an approach that others are also beginning to investigate [11]. Advances already made in the field of statistical NLP (see e.g. [9] for an overview) are used to give a probabilistic justification for using *tf.idf* weights. The experiment described in this paper shows that the new weighting algorithm outperforms the Cornell version of BM25 algorithm on the TREC-7 collection.

2.1 A linguistically motivated probabilistic model of IR

In the linguistically motivated probabilistic model documents and queries are described by compound events. A compound event is an event that consists of two or more single events, as when a die is tossed twice or three cards are drawn one at a time from a deck [10]. The single events that define the compound event are the index terms in the collection. Given a document-id, index terms are assumed to be independent. The probability of the compound event should therefore be calculated by multiplying the probabilities of the single events as in equation 1.

$$P(T_1, T_2, \cdots, T_n|D) = \prod_{i=1}^{n} P(T_i|D) \tag{1}$$

The general idea is the following. Each document contains a small sample of natural language for which the retrieval system should build a statistical language model $P(T|D)$ where T is a single event. If the user enters a query $T_1, T_2, \cdots, T_n$ the system uses equation 1 to calculate the probability of that query given each possible value of the document-id D. Perhaps the most straightforward way to estimate the probabilities $P(T|D)$ would be maximum likelihood estimation. A maximum likelihood estimate maximises the probability of observed events and assigns zero probability to unseen events. This makes the maximum likelihood estimate unsuitable for directly estimating $P(T|D)$ because it would assign zero probability to each document that does not contain all of the query terms. The problem that many of the possible events do not occur in the actual data is a well known problem in the field of statistical NLP: the sparse data problem. There are a number of standard solutions to the sparse data problem, one of them being estimation by linear interpolation. It is possible to remove the zero probabilities by mixing the maximum likelihood model of $P(T|D)$ with a model that suffers less from sparseness like the marginal $P(T)$ as in equation 2.

$$P_{li}(T|D) = \alpha_1 P_{mle}(T) + \alpha_2 P_{mle}(T|D), \;\; 0 < \alpha_1,\alpha_2 < 1 \text{ and } \alpha_1 + \alpha_2 = 1 \tag{2}$$

In equation 2 global information $P(T{=}t)$ on the term t is mixed with local information $P(T{=}t|D)$ on the term. The mix of global and local information is determined by the value of α_1 (which also

determines the value of α_2 by $\alpha_2 = 1 - \alpha_1$). It is standard practice in IR to use the document frequency for global information and the term frequency for local information. The document frequency $df(t)$ is defined by the number of documents in which the term t occurs. The term frequency $tf(t,d)$ is defined by the number of times the term t occurs in the document d. Equation 3 defines how probabilities are estimated from document frequencies and term frequencies.

$$P(T_i = t_i|D = d) = \alpha_1 \frac{df(t_i)}{\sum_t df(t)} + \alpha_2 \frac{tf(t_i,d)}{\sum_t tf(t,d)} \tag{3}$$

2.2 Rewriting to vector product normal form

We used the TNO vector space retrieval engine for the TREC-7 experiments. Obviously the ranking formula based on equations 1 and 3 cannot be used directly in a vector space engine. There is however a ranking formula that produces the same ranking as the ranking formula introduced in the previous section which can be implemented using the vector product similarity measure. This can be shown by rewriting.

$$P(T_1 = t_1, \cdots, T_n = t_n|D = d) = \prod_{i=1}^{n} (\alpha_1 \frac{df(t_i)}{\sum_t df(t)} + \alpha_2 \frac{tf(t_i,d)}{\sum_t tf(t,d)}) \tag{4}$$

$$\propto \sum_{i=1}^{n} \log(1 + \frac{tf(t_i,d)}{df(t_i) \sum_t tf(t,d)} \cdot \frac{\alpha_2 \sum_t df(t)}{\alpha_1}) \tag{5}$$

Equation 4 follows directly from equation 1 and 3. Multiplying equation 4 with values that are the same for each document, like α_1 and $df(t)$, will not affect the final ranking. Moreover, any monotonic transformation of the ranking formula will also produce the same ranking of documents. Instead of the product of weights we could therefore rank the documents by the sum of logarithmic weights. Using these two considerations the ranking formula can be rewritten as shown in equation 5. As a final step the query weights of the vector product formula can be used to account for multiple occurrences of the same term in the query. The resulting vector product version of equation 4 is displayed in table 1. Note that we end up with a document term weight that by the definition

vector product formula:	$\text{similarity}(Q,D) = \sum_{k=1}^{l} w_{qk} \cdot w_{dk}$
query term weight:	$w_{qk} = tf(t_k, q)$
document term weight:	$w_{dk} = \log(1 + \frac{tf(t_k,d)}{df(t_k) \sum_t tf(t,d)} \cdot \frac{\alpha_2 \sum_t df(t)}{\alpha_1})$

Table 1: vector product version of weighting algorithm

of Salton et al. [13] can be interpreted as a *tf.idf* weight with document length normalisation. However, the $\sum_t tf(t,d)$ in the denominator of the document term weight in table 1 is the result of the requirement that probabilities have to sum up to one and not the results of document length normalisation. Document length normalisation *is* assumed by the fact that we ignore the prior probability $P(D)$. In fact we may assume that longer documents are more likely to be relevant by using the prior probability of equation 6 and ranking the documents using $P(T_1 = t_1, \cdots, T_n =$

$t_n|D = d) \cdot P(D = d)$.

$$P(D = d) = \frac{\sum_t tf(t,d)}{\sum_t \sum_d tf(t,d)} \quad (6)$$

This results in a weighting algorithm that cannot be rewritten into the vector product normal form. It can however be implemented fairly easily by initialising similarities to $\log(\sum_t tf(t,d))$ instead of to zero when processing the query. This version of the weighting algorithm was used in the TREC-7 experiments.

2.3 Pseudo relevance feedback

Supplementary runs were done with a Rocchio-like pseudo relevance feedback. After an initial retrieval run, the top 200 of the weighted index terms extracted from a concatenation of the top 3 documents were added to the query with a ratio of 20 : 3, i.e. the weight of the added terms was multiplied by 3/20 before adding. These parameters were determined empirically by experimenting with the English topics of the TREC-6 CLIR track.

2.4 Synonyms

We also experimented with query expansion based on synonyms taken from the Van Dale lexical database. The expansion worked well with the TREC-6 CLIR queries, but performed disappointing with the TREC-6 ad-hoc queries. No official run was submitted for this experiment

2.5 Experimental setup and results

For the ad-hoc task we built the simplest possible version of the Twenty-One system. We used the TNO vector space engine for indexing and Porters algorithm for stemming. Stopwords were removed from the documents, including words that are frequent in previous TREC topics like *relevant* and *document*. Queries were generated automatically from the full topics (title, description and narrative) using the same procedure as used for indexing.

As already mentioned above, the linguistically motivated weighting algorithm has one free parameter that defines the mix of local and global frequency information. We did three pilot experiments using Cranfield, the English CLIR collection topics 1-24 and the TREC-6 ad-hoc collection topics 251-300. The pilot experiments indicated that the best values for α_1 and α_2 are approximately the same for all three collections: $\alpha_1 = 0.85$ and $\alpha_2 = 0.15$. We used these values in the TREC-7 ad-hoc experiments.

runname	description
tno7cbm25	run using the Cornell version of BM25
tno7tw4	run using the linguistically motivated weighting algorithm
tno7exp1	run using the linguistically motivated algorithm with pseudo relevance feedback

Table 2: description of ad-hoc runs

According to Voorhees and Harman [16] the most popular weighting algorithm in TREC-6 was the Cornell implementation of the Okapi BM25 algorithm [12, 14]. We decided to compare the performance of the new weighting algorithm with the Cornell version of BM25, resulting in the

first two runs listed in table 2. As a third run we submitted a run that uses the new weighting algorithm and pseudo relevance feedback.

runname	avg. prec.
tno7cbm25	0.2315
tno7tw4	0.2490
tno7exp1	0.2785

Table 3: results of ad-hoc runs

Table 3 lists the results of the three runs on the TREC-7 collection. The results show that the linguistically motivated weighting algorithm outperforms the Cornell version of BM25. The results also show a remarkable performance gain due to pseudo relevance feedback. The pseudo relevance feedback run performed above or equal to median on 44 of the 50 topics.

3 The CLIR task

In TREC-7 we intended to improve our TREC-6 results in the following ways:

- improve the monolingual system,
- improve lexical lookup,
- extend context sensitive disambiguation.

After reassessment of TREC-6 runs and experiments with manual disambiguation, we chose to experiment with an extension of the retrieval model in the direction of boolean interpretation. The manually disambiguated runs gave only a very small improvement in performance whereas initial experiments with a probabilistic interpretation of boolean structured translated queries (cf. sections 3.1 and 3.3) gave promising results on the TREC-6 CLIR topic set [3].

3.1 Query translation using the Van Dale dictionaries

The VLIS database from Van Dale Lexicography is a relational database which contains all lexical knowledge that is used for publishing the dictionaries Dutch → foreign language (German, French, English, Spanish). So the database is based on Dutch headwords with translation relations to equivalent lemmas in the foreign languages. The lexical material from the foreign language → Dutch companion dictionaries is not included in the VLIS database. Translation from one TREC language to another will go by using Dutch headwords that have the query words as their translations as interlingua, and then translating the Dutch headwords into one ore more words in the target language.

language	simple	composit	total
english	260k	40k	300k
german	224k	24k	248k
french	241k	23k	264k
spanish	139k	28k	167k

Table 4: number of translation relations in the VLIS database

The VLIS database contains simple and composite (multi-word) lemmas for 5 languages, Dutch being the pivot language. For Dutch there are 270k entries corresponding to about 513k concepts. These concepts have translations into French, Spanish, German and English. For TREC-7 we only used the simple lemmas. We used the the Xelda toolkit of Xerox Research Centre in Grenoble for tagging and lemmatisation of the topics. Translation of the topics was done in a series of steps:

1. Tokenising
2. Tagging
3. Lemmatisation
4. Stopword removal
5. Multi stage lexical lookup:
   - Lookup of compounds, subcompounds and compound parts
   - Lookup with and without syntactic constraint
   - Lookup of main / synonym translation
   - Lookup with or without capitalisation
   - Fuzzy lookup (not used in official runs)
6. Weighting of translations / Selection of best translation

The weighting of translations is based on the number of occurrences of a certain translation in the dictionary. Some head words carry over to the same translation for different senses. For example the Dutch head word *jeugd* can be translated to *youth* in three senses: the sense of 'characteristic', 'time-frame' and 'person'. The sense of 'person' has a synonym translation: *youngster*. As *youth* occurs in the dictionary under three senses we assign it a weight that is three times as high as the weight for *youngster*. Dutch serves as an interlingua, therefore translation can be carried out via several Dutch pivot lemmas. This possibly generates even more occurrences of the same possible translation. The implicit assumption made by weighting translations is that the number of occurrences generated from the dictionary may serve as rough estimates of actual frequencies in parallel corpora. Ideally, if the domain is limited and parallel corpora on the domain are available, weights should be estimated from actual data [4]. Weighting of possible translations is used for structured queries (see section 3.3).

A complication with respect to TREC-6 was the extension of the document base with Italian documents. We decided to use the machine translation system Systran[3] to handle our query translations to Italian. This Web service hosts translation capabilities from and to English for 5 European languages. For the German queries, we used English as a pivot language to translate to Italian because the language pair German-Italian is not available. Morphological stemming of the Italian translations were produced by ETH Zürich.

3.2 Fuzzy expansion of query terms

We developed a fuzzy expansion algorithm based on relative frequencies of letter trigrams and edit distance. This algorithm matches a query term with index terms that are similar but not exactly equal to the query term. This is useful for source language terms that do not have an entry in our dictionary, but do have a similar translation in the target language like domain specific jargon, person names and geographical names. It is also useful for spelling variation and spelling mistakes. We applied two forms of fuzzy expansion, a conservative version where only (translated) query

[3] http://babelfish.altavista.com

terms which are not found in the index are expanded. A more liberal version expanded *every* query term. The expansions were treated as a single concept by the TNO vector space engine. The fuzzy matching module ISM[4] was developed in the 1970's at TNO by de Heer [1].

3.3 Probabilistic interpretation of boolean queries

For almost every headword the VLIS lexical database gives a number of senses, each with a main translation and synonym translations. Instead of picking the preferred translation from the dictionary, it might be advantageous to use all possible translations to search for relevant documents as this might lead to higher recall. If possible translations are weighted and structured properly the document collection itself can be used for implicit disambiguation of possible translations [5]. Disjunction is a natural operator to combine possible translations of a query term, whereas conjunction is used to link all translations in a way that reflects the original query.

We developed a weighting algorithm that inputs boolean queries in conjunctive normal form and assigns probabilities to documents given these queries. Boolean queries are generated automatically from the topics by translation. The algorithm takes into account the relative frequencies of possible translations which are based on information from the VLIS lexical database as mentioned above. Details of the algorithm will be published in the near future.

3.4 Merging of runs

We created a separate index for each language. An alternative approach would be to build one big index on all four collections. The main reason for using four separate indexes is that our experimental setup was still based on experiments with only one target language. There is however a more fundamental reason for using four indexes. From the perspective of the linguistically motivated model of IR it would be silly to build a language model which mixes words of four different languages.

Merging the retrieval runs on separate collections into a combined run is not a trivial problem [6]. For the monolingual case, the problem is known as the *collection fusion problem* [15]: similarities are not comparable across collections because of the incorporation of collection-dependent frequency counts like document frequencies. One solution to the problem is to bias the similarities for each subcollection differently, e.g. by a collection specific linear transformation of the similarities.

We used the following approach to merging. Documents were retrieved from each collection and merged after adding a collection specific constant. The collection specific constant was determined by forcing the average similarities over all 28 topics to be the same for each language collection. The collection specific constant makes sure that, on average, the same number of documents is retrieved from each language. This approach is not an elegant solution to the merging problem. In fact, one could argue that it violates the TREC ad-hoc task description. We incorporated this method anyway to make sure that our contribution to the pool would not be biased towards one or two languages. In the near future we will develop a more elegant merging strategy.

3.5 Experimental setup and results

For the CLIR task we were able to use most of the Twenty-One modules. We used the VSM engine for retrieval and the fuzzy index to recover from missing translations and spelling variants of proper names. As a preprocessing step before indexing and translation we used the Xerox morphological tools. The translation of the topics was based on a word by word translation process, using the VLIS lexical database from Van Dale.

[4]ISM: Informatie Sporen Methode = Information Trace Method

runname	description
tno7edp	dictionary preferred translation of English queries into 3 other languages; fuzzy expansion of query terms without dictionary entry
tno7edpx	dictionary preferred translation of English query into 3 other languages; fuzzy expansion of each query term
tno7egr	probabilisticly interpreted boolean query of all possible translations of the English queries into 3 other languages
tno7ddp	dictionary preferred translation of German queries into 3 other languages; fuzzy expansion on query terms without dictionary entry
tno7eef	dictionary preferred translation of English query into French; fuzzy expansion of query terms without dictionary entry
tno7mx	unofficial run: merged run of four monolingual searches; fuzzy expansion of each query term

Table 5: description of CLIR runs

Table 6 lists the results for our official runs. As a baseline we included **tno7mx**, an unofficial run which is based on a merge of 4 monolingual runs. The best result is achieved by **tno7egr** the probabilistic boolean run. The more liberal fuzzy expansion run also produced improved results. It could be interesting to combine both techniques. The results with the German version of the topics is lower than expected, a topic-wise comparison is needed to assess the cause. **tno7eef**, the run with English topics in the English and French subcollection has a higher score, probably because of the reasons addressed in section 3.6 and because of the major contribution of the monolingual English run in the merged run.

runname	avg. prec.	relative to tno7mx (%)
tno7edp	0.2716	83
tno7edpx	0.2846	87
tno7egr	0.3009	92
tno7ddp	0.2382	73
tno7eef	0.3404	-
tno7mx	0.3282	100

Table 6: results of CLIR runs

Table 7 shows the results of our monolingual runs which were the building blocks for our merged runs. The table only shows the results on topics that had hits in all four languages: topics 26, 44, 46 and 51 are not included in this evaluation. For the per language evaluation we used the qrels from the merged runs. This will give a distorted picture (more about this aspect in section 3.6) but we think that the qrels are still useful to look at relative performances of runs for a particular language, and to get some idea of the merging effect. The merging effect turned out to be quite impredictable because the ranking of the runs based on column 6 is quite different from the ranking based on merged runs (column 7). For the average figure (column 6), the dictionary preferred runs are better than the probabilistic interpreted boolean run. However this ordering is reversed in the official ranking. Apparently the merging procedure that we applied is suboptimal. Further research is needed to explain these results.

runname	avg.prec. english	avg.prec. french	avg.prec. german	avg.prec. italian	average over 4	merged	relat. to avg. (%)
tno7edp	0.4923(m)	0.2893	0.2427	0.3846	0.3522	0.2750	78
tno7edpx	0.4985(m)	0.2945	0.2771	0.3927	0.3657	0.2901	79
tno7egr	0.4923(m)	0.3005	0.2253	0.3846	0.3506	0.3084	88
tno7ddp	0.3549	0.2452	0.4199(m)	0.3266	0.3367	0.2573	76
tno7mx(m)	0.4985	0.4542	0.4187	0.4651	0.4591	0.3569	78

Table 7: per language performance and the effect of merging on 24 topics TREC-7, (m) indicates monolingual run

3.6 Pool construction

The methodology for pool construction has some important but often neglected assumptions which complicate the evaluation of merging. The pool is based on a merge of the top 100 fraction of the runs which are judged. The assumption is that the pool will contain most relevant documents. Suppose that the average number of relevant documents per query is much larger than 100, then the validity of the assumption is questionable. In that case it is quite probable that a considerable amount of relevant documents is not judged because they are not in the pool. We computed some pool statistics, in particular we looked at the average judged fraction of the collection for each task. This is defined as the number of relevance judgements divided by the product of the number of queries and the collection size. The pool statistics are listed in table 8.

collection	total docs.	judged docs.	relevant docs.	no hits in topic	judged fraction
english	242,866	9,810	1,689	26,46	0.0014
french	141,637	6,130	991	-	0.0015
german	185,099	4,558	917	26	0.0009
italian	62,359	3,062	501	26,44,51	0.0018
total	631,961	23,560	4,098	average:	0.0013

judged docs.	relevant docs.	no hits in	judged fraction
8,713	1,385	8	0.0015
12,663	1,518	22	0.0037
9,086	1,172	22	0.0020
-	-	-	-
30,462	4,075	average:	0.0022

Table 8: CLIR task statistics (a) 28 topics TREC-7, (b) 24 topics TREC-6

The judged fraction of the ad-hoc task is 0.0030, which means that on average 3 per mill of the document collection has been judged for each query. For this year's CLIR track this figure is 0.0013 on average. This clearly reflects that the CLIR pool is much smaller than the ad-hoc pool. The pool is also smaller than the TREC-6 CLIR pool which has a judged fraction of 0.0022. Conclusion is that results derived from the CLIR pool are less reliable than last year. This might be partly due to the decision to base the pool on merged runs and not on monolingual runs.

The average number of relevant documents per topic is 93.5 in ad-hoc and 146.35 for the merged CLIR collection. For the monolingual subcollections the figures are: 60(EN), 35(FR), 33(DE) and 18(IT). This means that the "pool validity assumption" is probably violated for the merged CLIR task, meaning that the average precision figures for the merged runs are probably too high.

The figures for the monolingual results in table 7 are probably even more flattered. This can be explained as follows: suppose that we have subcollections of similar size, and that topics have roughly similar amounts of relevant documents in each subcollection. This means that on average only the top 25 documents of each subcollection run is judged in comparison to the top 100 documents for the merged runs. When we apply the qrels of the merged runs pool to the subcollection

runs, the average precision is artificially high. Most probably a pool based on the top 100 of subcollection runs would bring in more relevant documents with lower ranks. This explanation also partly accounts for the high score of the merged run on the English French subcollections **tno7eef**.

4 Conclusion and outlook

The new linguistically motivated retrieval model outperforms the popular Cornell BM25 weighting scheme in the ad-hoc task with 7.5 %. Pseudo relevance feedback improves the average precision with an additional 12.7 %. The most successful run in the CLIR task was a probabilistic interpreted boolean run. However, at this point the merging process is ill understood. Evaluation of the merging process is complicated because the qrels of the merged runs cannot be used for a true comparison with subcollection runs. In future CLIR tracks, this problem should be tackled, for example by adding the top 100 documents from subcollection runs to the pool. There is some evidence that the 'probabilistic boolean' approach to CLIR is more successful than the 'dictionary preferred translation' approach. Further research is needed to assess whether 'boolean retrieval' is a better approach to the CLIR problem than disambiguation. A second result of the CLIR evaluation is that ISM (the fuzzy matching algorithm) gives a significant improvement in performance. The best result was obtained with the liberal variant: each query term is expanded with orthographically close variants, catching spelling variation in proper names and cognates in case of missing translations.

Acknowledgements

The work reported in this paper is funded in part by the Dutch Telematics Institute project Druid and the EU projects Twenty-One (IE 2108) and Pop-Eye (LE 4234). We would like to thank ETH Zürich for stemming the Italian collection, the Italian topics and our translations to Italian. Furthermore we would like to thank Renée Pohlmann and Rudie Ekkelenkamp both from TNO-TPD for their help with manual disambiguation of test runs and data preparation respectively.

References

[1] T. de Heer. Quasi comprehension on natural language simulated by means of information traces. *Information Processing & Management*, 15:89–98, 1979.

[2] D. Hiemstra. A linguistically motivated probabilistic model of information retrieval. In C. Nicolaou and C. Stephanidis, editors, *Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries (ECDL-2)*, pages 569–584, 1998.

[3] D. Hiemstra and F.M.G. de Jong. Cross-language retrieval in Twenty-One: using one, some or all possible translations? In *Proceedings of the 14th Twente Workshop on Language Technology (TWLT-14)*, pages 19–26, 1998.

[4] D. Hiemstra, F.M.G. de Jong and W. Kraaij. A domain specific lexicon acquisition tool for cross-language information retrieval. In *Proceedings of RIAO'97 Conference on Computer-Assisted Searching on the Internet*, pages 255–266, 1997.

[5] D.A. Hull. Using structured queries for disambiguation in cross-language information retrieval. In *AAAI Symposium on Cross-Language Text and Speech Retrieval.* American Association for Artificial Intelligence, 1997.

[6] D.A. Hull and G. Grefenstette. A dictionary-based approach to multilingual information retrieval. In *Proceedings of the 19th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96)*, 1996.

[7] W. Kraaij. Multilingual functionality in the Twenty-One project. In *AAAI Symposium on Cross-Language Text and Speech Retrieval.* American Association for Artificial Intelligence, 1997.

[8] W. Kraaij and D. Hiemstra. Cross-language retrieval with the Twenty-One system. In E. Voorhees and D. Harman, editors, *Proceedings of the 6th Text Retrieval Conference TREC-6*, pages 753–761. NIST Special Publication 500-240, 1998.

[9] C. Manning and H. Schütze, editors. *Statistical NLP: Theory and Practice, draft.* http://www.sultry.arts.su.edu.au/manning/courses/statnlp/, 1998.

[10] A.M. Mood and F.A. Graybill, editors. *Introduction to the Theory of Statistics, Second edition.* McGraw-Hill, 1963.

[11] J.M. Ponte and W.B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, 1998.

[12] S.E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*, pages 232–241, 1994.

[13] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[14] A. Singhal, G. Salton, M. Mitra and C. Buckley. Document length normalization. Technical Report TR95-1529, Cornell University, 1995. http://cs-tr.cs.cornell.edu/.

[15] E.M. Voorhees, N.K. Gupta and B. Johnson-Laird. The collection fusion problem. In D.K. Harman, editor, *Proceedings of the 3rd Text Retrieval Conference TREC-3*, pages 95–104. NIST Special Publication 500-225, 1995.

[16] E.M. Voorhees and D.K. Harman. Overview of the 6th text retrieval conference. In *Proceedings of the 6th Text Retrieval Conference TREC-6*, pages 1–24. NIST Special Publication 500-240, 1998.

AT&T at TREC-7

Amit Singhal John Choi Donald Hindle David D. Lewis
Fernando Pereira

AT&T Labs–Research
{singhal,choi,hindle,lewis,pereira}@research.att.com

Abstract

This year AT&T participated in the ad-hoc task and the Filtering, SDR, and VLC tracks. Most of our effort for TREC-7 was concentrated on SDR and VLC tracks. On the filtering track, we tested a preliminary version of a text classification toolkit that we have been developing over the last year. In the ad-hoc task, we introduce a new tf-factor in our term weighting scheme and use a simplified retrieval algorithm. The same weighting scheme and algorithm are used in the SDR and the VLC tracks.

The results from the SDR track show that retrieval from automatic transcriptions of speech is quite competitive with doing retrieval from human transcriptions. Our experiments indicate that document expansion can be used to further improve retrieval from automatic transcripts. Results of filtering track are in line with our expectations given the early developmental stage of our classification software. The results of VLC track do not support our hypothesis that retrieval lists from a distributed search can be effectively merged using only the initial part of the documents.

1 Introduction

Spoken document retrieval (SDR) and retrieval from very large collections (VLC) are the main areas of interest for AT&T at TREC-7. For most of our work, we use an internally modified version of the SMART retrieval system developed at Cornell University. [1, 9] We are in the process of developing a text classification toolkit called ATTICS. The filtering track gave us an opportunity to stress test an early version of this toolkit on a large task.

For speech retrieval, we believe that parallel text corpora, for example printed news from the same time period, can be successfully exploited to improve retrieval effectiveness of a system. This is especially true for the news material currently being used in the SDR track. We use these ideas in our SDR track participation. Initial results from the use of a parallel corpus are quite encouraging.

As the amount of data available electronically (for example on the Web) grows exponentially, it is becoming increasingly hard to maintain a single centralized index to the data. Distributed retrieval is a solution; however most distributed retrieval algorithms expect some cooperation from the individual servers, which is often hard or even impossible to get. In the VLC track we simulate "totally independent" distributed collections, and use a new result merging algorithm that leverages the summary text for retrieved documents, usually provided by the search engines, to merge results from various servers.

2 Ad-hoc Runs

Over the last year, we have modified the SMART retrieval system with the following minor changes: 1) we use a shorter stop-list of 410 stopwords instead of the standard SMART stop-list of 571 stopwords, 2) we use a new tokenizer that handles hyphens and ampersands (as in AT&T) in a better manner, 3) phrase formation is governed by a new set of rules, and 4) we have generated a new set of statistical phrases (187,908 phrases) from the news corpora included in disks 1–5 (AP, WSJ, SJMN, FT, FBIS, LATimes). These phrases are used in ad-hoc, SDR, and the SMART run of the filtering track. We do not use any phrases in the VLC track.

d *tf* factor:

$$1 + ln(1 + ln(tf)) \qquad 0 \text{ if } tf = 0$$

t *idf* factor:

$$log(\frac{N+1}{df})$$

b pivoted byte length normalization factor:

$$\frac{1}{0.8 + 0.2 \times \frac{length\ of\ document\ (in\ bytes)}{average\ document\ length\ (in\ bytes)}}$$

where, tf is the term's frequency in text (query/document)
N is the total number of documents in the (training) collection
df is the number of documents that contain the term, and
the average document length depends on the collection.

dnb weighting: **d** factor × **b** factor
dtb weighting: **d** factor × **t** factor × **b** factor
dtn weighting: **d** factor × **t** factor

Table 1: Term Weighting Schemes

Poor performance of the logarithmic *tf*-factor—$1 + ln(tf)$—on the TREC-6 ad-hoc task forced us to revisit our term weighting methodology. The main reasoning for use of a logarithmic *tf*-factor instead of (say) the raw *tf* of the terms is to ensure that high *tf* for one query term in a document does not place that document ahead of other documents which have multiple query terms but with low *tf* values. By using a logarithmic *tf*-factor, the effect of *tf* dampens with increasing *tf* values, and a single match usually doesn't outweigh multiple matches. We found that for short queries (our main interest in the ad-hoc task) a logarithmic *tf*-factor does not adequately reduce the effect of large *tf* values. Therefore we need a *tf*-function that reduces the *tf* contribution even more than the logarithmic *tf*-factor as *tf* increases. Double logarithm—$1 + ln(1 + ln(tf))$—is an obvious choice. We also like double logarithm since it doesn't introduce any new parameters into our weighting scheme.

Next we revisited our pivoted unique document length normalization function. [12] When both words and phrases are used as terms in a document vector, the definition of the number of unique terms in the document is not elegant, especially since phrases are formed from words but are counted as independent unique terms. Earlier work has shown that byte length of a document can be successfully used in a pivoted formula for document length normalization. [12] We switch to using *pivoted byte size normalization* in our weighting scheme. As a side benefit, the weighting code inside SMART is simplified when pivoted byte size normalization is used. With these changes, we use the term weighting schemes shown in Table 1.

2.1 Approach

This year we use a simple two pass *pseudo-feedback* based approach in the ad-hoc task. Many groups have used such an approach in the last few TRECs. Here are the steps in the process:

- **Pass-1**: Using *dtn* queries and *dnb* documents, a first-pass retrieval is done.
- *Expansion:* Top ten documents retrieved in the first pass are *assumed* to be relevant to the query and documents ranked 501–1000 are assumed to be non-relevant. Rocchio's method (with parameters $\alpha = 3,\ \beta = 2,\ \gamma = 2$) is used to expand the query by adding twenty new words and five new phrases with highest Rocchio weights. [8] To include the *idf*-factor in the expansion process, documents are *dtb* weighted.
- **Pass-2:** The expanded query is used with *dnb* documents to generate the final ranking of 1,000 documents.

Task Title+Desc	Baseline *dnb.dtn*	Expansion from target collection	Expansion from TREC D12345	Conservative Collection Enrichment
TREC-6 (AvgP)	0.2290	0.2661 (+16.2%)	0.2781 (+21.4%)	0.2906 (+26.9%)
# Q better/worse		0/0	31/19	34/16
TREC-7 (**att98atdc**)	0.2182	0.2830 (+29.7%)	0.2861 (+31.1%)	**0.2961** (+33.7%)
# Q better/worse		0/0	25/25	32/18

Table 2: Effect of conservative collection enrichment

2.2 Conservative Collection Enrichment

At TREC-6 some groups (*e.g.*, City University and University of Massachusetts) used a technique that Kwok calls "collection enrichment". [5] The main idea is to run the first pass on a much larger collection than the target collection. The hope is that a larger text collection will have more relevant documents for the query and our methods will pull more relevant documents in the top ten or twenty documents, thereby strengthening the assumption of relevance employed in the query expansion stage. We employ such a technique in our ad-hoc runs using all TREC disks (1–5) as the large collection.

One obvious problem with this approach is possible "domain mismatch". If the top documents retrieved from the large collection are from a completely different domain than the top, presumably query-related, documents in the target database, collection enrichment can cause the query to drift away from target relevance. During our test with collection enrichment, we found that this indeed was a problem. Therefore we devised a conservative collection enrichment technique which forces the expanded query to remain in the target domain by not allowing any expansion terms proposed only by expansion on the large collection.

For example, for query 354: *journalist risks*, the first pass from the large corpus (TREC disks 1–5) retrieves several stories that talk about a list of missing or detained U.N. workers around the world, some of them were killed, including a journalist. Even though these documents are reasonably relevant to the topic, the focus of these documents is different than the topical documents in the target database. These documents add terms related to the U.N. and its missing employees to the query and drift the query away from relevance in the target database which does not have reports on missing U.N. journalists.

To fix this problem, we allow only those new terms to be added to the query that are proposed by top documents retrieved from the target database. To capture some effect of collection enrichment, we allow the large collection to impact the term selection and the final weights of the terms. Here are the steps involved in our conservative collection enrichment:

1. Use the query to retrieve the top ten document from the target collection and build a list of potential expansion terms along with their proposed Rocchio weights. Only terms that appear in at least two of the ten documents and have a positive Rocchio weight are considered.

2. Use the query to retrieve ten document from the large collection, and compute the Rocchio weights for all the expansion terms proposed in step 1.

3. Add term weights proposed in steps 1 and 2 and add the top weighted twenty words and five phrases to the original query. This allows for terms that are weakly proposed (*i.e.*, with a low weight) by the target collection but are strongly proposed by the larger collection to enter the query and vice-versa.

2.3 Results

We submitted two fully-automatic runs based on the above described algorithms. One run, **att98atc**, was based on title-only queries and the other run, **att98atdc**, used title+description queries. We do not use the *narrative*-portion of the TREC topics in any of our runs since we don't believe that real users will ever provide us with such long queries. To highlight the benefits of conservative collection enrichment, we first present the results of running this algorithm on last year's (TREC-6) adhoc task (title+description) and this year's task (our run att98atdc) in Table 2. The second column in Table 2 shows the baseline average

Run	Average Precision	Best	>= Median	< Median
att98atc (title only)	0.2488	0	32	18
att98atdc (title+desc)	0.2961	1	41	8
att97atde (title+desc)	0.2940	0	44	6

Table 3: Results for adhoc runs

precision of a straight retrieval when documents are *dnb* weighted and queries are *dtn* weighted. The third column shows the results when only the target database is used in the query expansion process. For the TREC-6 task, we get a 16.2% improvement in average precision. This improvement is almost 30% for this year's task. If we use the large collection for query expansion, column four shows that the performance improves some more. The last column shows that our conservative collection enrichment further improves the performance some.

Even though in terms of average precision, the conservative method is not much better than expanding purely from the large collection, but as the rows labeled *# Q better/worse* show, it is more stable with respect to the number of queries that improve or deteriorate in comparison to expansion from the target collection (the sensible baseline for this comparison since if we compare to unexpanded queries, which is the baseline used in the average precision rows, all expansion strategies will show large gains and the relative performance of different expansions will be hard to judge). For the TREC-6 task, when expansion is done from D12345, 31 queries improve but 19 queries deteriorate (column 4) in comparison to base expansion. But if we do conservative collection enrichment, we cut our losses to 16 queries instead, (column 5) and we gain some in average precision. These numbers are even more striking for this year's task. Expansion from the large collection is worse than expansion from the target collection for half the queries, it is better for the other half. But the conservative expansion reduces our losses from 25 queries to 18 queries only, and the losses are, in general, smaller. These results show that conservative collection enrichment is more stable than pure enrichment, even though the gains in term of average precision are not much.

The official results for our ad-hoc runs are shown in Table 3. Both the runs att98atc and att98atdc did well, especially given that the medians are drawn from a pool which includes runs that use the full topic text (rich with content words from the "narrative" portion of the topics) to construct the query. Our runs just use either very short queries (title only) or short queries (title+description). We also submitted an experimental run **att97atde** which enriched the term set by using word cooccurrence pairs as we have used in the past in the routing task. [11] We were hoping that these pairs would be useful in the ad-hoc setting too, but using word-pairs didn't improve our retrieval effectiveness.

3 SDR Runs

We use our own speech recognizer to process the SDR track data. One of our submitted runs **att-s1** uses the one-best recognizer transcript and does retrieval using an algorithm quite similar to the algorithm we have used in the ad-hoc task. The other run **att-s2** uses word-lattices generated by our recognizer and a parallel text corpora to do *document expansion.* The expanded documents are then used for retrieval instead of the one-best recognizer transcript.

3.1 Speech Recognizer

Our speech recognition process involves the following steps. Prior to recognition, each speech story is segmented into approximately one minute long prosodically well-formed segments using a CART based classifier. [2] The resulting segments are submitted to another wideband/narrowband classifier for selection of the acoustic model to be used in recognition of that segment.

The recognizer is based on a standard time-synchronous beam search algorithm. The probabilities defining the transduction from text-dependent phone sequences to word sequences are estimated on word level grapheme-to-phone mappings and are implemented in the general framework of weighted finite-state transducers. [7] Transducer composition is used to generate word lattice output.

We use continuous density, three-state, left-to-right, context-dependent hidden Markov phone models. These models were trained on 39-dimensional feature vectors consisting of the first 13 mel-frequency cepstral coefficients and their first and second time derivatives. Training iterations included eigenvector rotations, k-means clustering, maximum likelihood normalization of means and variances and Viterbi alignment. The output probability distributions consist of a weighted mixture of Gaussians with diagonal covariance, with each mixture containing at most 12 components. The training data were divided into wideband and narrowband partitions, resulting in two acoustic models.

Language Models

We used a two pass recognition process. In the first pass, we built word lattices for all the speech using a minimal trigram language model and a beam that we had determined heuristically to provide manageable word lattices. These word lattices were then rescored, by removing the trigram grammar weights while retaining the acoustic weights and intersecting these lattices with a 4-gram language model. The 1-best path was extracted from the rescored lattices.

Both the first pass trigram language model and the rescoring 4-gram model are standard Katz backoff models [4], using the same 237 thousand word vocabulary. For choosing the vocabulary, all of the words from the SDR98 training transcript were used. This base vocabulary was supplemented with all words of frequency greater than two appearing in the New York Times and LA Times segments of LDC's North American News corpus (LDC Catalog Number: LDC95T21, see `www.ldc.upenn.edu`), in the period from June 1997 through January 1998. The vocabulary includes about 5,000 common acronyms (e.g "N.P.R."), and the training texts were preprocessed to include these acronyms.

The language model training was based on three transcription sources (the SDR98 training transcripts, HUB4 transcripts, transcripts of NBC nightly news) and one print source (the LDC NA News corpus of newspaper text). The first-pass trigram model was built by first constructing a backoff language model from the 271 million words of training text, yielding 15.8 million 2-grams and 22.4 million 3-grams. This model was reduced in size, using the approach of Seymore and Rosenfeld [10], to 1.4 million 2-grams and 1.1 million 3-grams. When composed with the lexicon, this smaller trigram model yielded a manageable sized network. The second pass model used 6.2 million 2-grams, 7.8 million 3-grams, and 4.0 million 4-grams. For this model, the three transcription sources (SDR, HUB4, NBC) were in effect interpolated with the text source (NA News), with the latter being give a weight of 0.1.

3.2 Retrieval System

For the SDR track, we use the NA News corpus (also used in the language model training described above) as the large collection for conservative collection enrichment (see Section 2.2). Since the test data is dated from June 1997 to January 1998, we used news dated from May 1997 to February 1998 (one month before and after) from the NA news corpus.

Algorithm

We use the following algorithm in our reference run—**att-r1**, our two baseline runs—**att-b1** and **att-b2**, and our first full SDR run—**att-s1**. This algorithm is exactly the algorithm we have used in the ad-hoc track with some parameters changed to deal with the small size of the speech database.

- **Pass-1**: Using *dtn* queries and *dnb* documents, a first-pass retrieval is done.
- *Expansion:* Top five documents retrieved in the first pass are *assumed* to be relevant to the query and documents ranked 101–200 are assumed non-relevant. The query is expanded by adding ten new words and two new phrases using Rocchio's formulation (parameters $\alpha = 2$, $\beta = 1$, $\gamma = 1$). To include the *idf*-factor in the expansion process, documents are *dtb* weighted.
- The above expansion step is performed once on the target collection and once on the large NA news collection. Conservative collection enrichment is done as described in Section 2.2.

- **Pass-2:** The expanded query is used with *dnb* documents to generate the final ranking of 1,000 documents.

One of the main motivations for using the above algorithm is to keep our ad-hoc algorithm uniform across tasks.

Lattice Based Document Expansion

The one-best transcript from a recognizer misses many content words and adds some spurious words to the spoken document. The misses reduce the *word-recall* (proportion of spoken words that are recognized) and the spurious words reduce the *word-precision* (proportion of recognized words that were spoken). We believe that information retrieval algorithms would benefit from a higher word recall and are robust against poor word precision. An approach to enhance word recall is to add new words that "could have been there" (words that were probably spoken but weren't the top choice of a speech recognizer) to the automatic transcriptions of a spoken document.

Several techniques are plausible for bringing new words into a document. An obvious one from an IR perspective is *document expansion* using similar documents: find some documents related to a given document, and add new words from the related documents to the document at hand. And from a speech recognition perspective, the obvious choice is to use word lattices which contain multiple recognition hypotheses for any utterance. A word lattice contains words that are acoustically similar to the recognized words could have been said instead of the words recognized in the one-best transcription.

We use both these techniques to do controlled document expansion for our second full SDR run **att-s2**. In our experiments we found that each method when used alone adds more spurious words to a document than is desirable. However, a controlled document expansion that incorporated information from both the sources helps in reducing the spurious words, allowing the good words to still be added to a document.

We take the one-best recognition for a story and look for similar stories in the print media (NA news). This is done by simply running the one-best recognition for the story as a raw-$tf \times idf$ weighted query on the NA news database. The idea being that important news would also be reported in the print media, and we can leverage words from there to enrich our spoken documents. We do not enforce any conditions like 'the returned stories from NA news should be from the same day or near the same date as the spoken document', even though one can imagine that this could possibly help.

We found that for speech stories that are not reported in the print media, marginally related stories are retrieved in response to the query (speech story), and unrelated words are brought into the story. To contain this problem, we force our expansion algorithm to choose only those words that are also present in the word lattice generated by our recognizer for the speech story. This restriction guarantees that the words being added to a document are also proposed by the speech recognizer, albeit with a low confidence.

The parameters for document expansion were chosen somewhat arbitrarily based on a quick inspection of the expansion terms and our experience with relevance feedback. We didn't have any testbed to tune our parameters. The following steps are used:

1. Twenty documents are retrieved from the NA news corpus for a given speech document. The one-best transcription for a speech document weighted using raw-$tf \times idf$ is used as a query.
2. 25% of the unique words, at most 50, new words are added to the speech document using Rocchio's formula. I.e., if the original one-best recognition has 80 unique (not counting repetitions) words, then 20 new words are added; if it has 200, then 50 new words are added; and if it has 300, then also only 50 new words are added. Following are some details of the expansion process:
   - Rocchio parameters $\alpha = 4,\ \beta = 1,\ \gamma = 0$ are used.
   - Only words that occurred in at least 10 of the 20 documents retrieved in step 1 are used as expansion terms.
   - Both the original speech document and the top documents from step 1 are *dtb* weighted (see Table 1) for Rocchio's formula. This yields expanded document vectors that have *idf* in the weights.

Transcription	Used in Run	WER	Term Recall	Term Precision
Reference	att-r1	0	100	100
Baseline-1	att-b1	34.1%	78.98%	78.02%
Baseline-2	att-b2	46.9%	68.40%	68.04%
Full SDR-1	att-s1	32.4%	81.79%	81.58%
Expanded Docs	att-s2	—	83.74%	67.50%

Table 4: Analysis of recognition and document expansion

Retrieval Condition	Baseline *dnb.dtn*	Expansion from target collection	Expansion from NA News	Collection Enrichment	Best	Above Median	Below Median
Reference	0.4548	0.5083	0.4864	**0.4992**	5	16	2
att-r1	—	+11.7%	+6.9%	+9.8%			
Baseline-1	0.4115	0.4925	0.4493	**0.4700**	9	10	4
att-b1	—	+19.7%	+9.2%	+14.2%			
Baseline-2	0.3358	0.3941	0.3983	**0.4065**	6	14	3
att-b2	—	+17.4%	+18.6%	+21.1%			
Full SDR-1	0.4371	0.5069	0.4839	**0.5065**	4	16	3
att-s1	—	+16.0%	+10.7%	+15.9%			
Full SDR-2	0.4535	**0.5300**	0.4981	**0.5120**	7	13	3
att-s2	—	+16.0%	+10.7%	+15.9%			

Table 5: Results for SDR track

3. *Idf* is removed from the expanded document vectors by dividing by component words' *idf*s. Now we have expanded document vectors that are *dnb* weighted. These expanded documents are used instead of the one-best transcriptions in the same retrieval algorithm as used in the run att-s1, and this run was submitted as our second SDR track run att-s2.

Results and Analysis

The recognition word error-rate, average term recall (proportion of spoken words that are recognized), and average term precision (proportion of recognized words that are spoken) for various automatic transcriptions are shown in Table 4. To compute the word recall and the word precision, we take all the stories that have more than ten index terms (non-stop word-stems as indexed by SMART, multiple occurrences of a stem counted as one term) in the human transcription and compute how many of the original index-terms (non-stop word-stems only, repetitions not counted) are recognized in the automatic transcription. We also count the number of spurious terms recognized by a recognizer. We compute per-document term recall and term precision, and average these values across documents to get the numbers reported in Table 4. We omit all storied that have less than ten terms since the recall/precision figures for them are quite unstable. The main reason behind discounting multiple occurrence of a term is that the *tf*-factor of our weighting scheme has a similar effect. Recognizing a word once is more important than correctly recognizing multiple occurrences of the same word.

Table 4 shows that the word error rate for our recognizer is 32.4%, slightly better than the 34.6% for the medium error transcriptions provided by NIST. This is also reflected in the higher term recall and precision for our recognizer. Document expansion is doing its job, though not as well as we would like it to. It does increase the term recall by another 2% on an average but it significantly reduces term precision from 81.58% to 67.5%. But as discussed in the following results, the deterioration in term precision does not hurt retrieval effectiveness. This supports our hypothesis that term recall is more important for our retrieval systems.

The results for various runs are shown in Table 5. The official numbers are presented in bold in column 5, along with various other numbers. The baseline retrieval results—*dnb* documents, *dtn* queries, no query expansion—are shown in column 2. The average precision on human transcription is 0.4548. The retrieval effectiveness falls when retrieval is done on a speech recognizer's automatic transcription of the speech. The

Code	Provided By	WER
Human	NIST	0%
CUHTK-S1	Cambridge University	24.8%
Dragon98-S1	Dragon Systems	29.8%
ATT-S1	AT&T Labs	31.0%
NIST-B1	Carnegie Melon (CMU)	34.1%
SHEF-S1	Sheffield University	36.8%
NIST-B2	Carnegie Mellon (CMU)	46.9%
DERASRU-S2	DERA	61.5%
DERASRU-S1	DERA	66.2%

Table 6: Different automatic transcriptions.

base effectiveness for the medium-error baseline transcription (B1) is 0.4115, a loss of 9.5%. As expected, the average precision falls further to 0.3358 for the high-error recognition (B2).

When retrieval is done on our own recognition, which has a marginally higher term recall/precision, the retrieval effectiveness jumps from 0.4115 for B1 to 0.4371, a gain of 6.2%, and we are running just 3.9% behind retrieval on human transcriptions. Document expansion removes even this difference and retrieval from expanded documents is at par with retrieval from human transcriptions. This is quite encouraging, especially when the expansion parameters were chosen without any guidance. This also shows that term recall is indeed more important than term precision. The term precision for the expanded documents is noticeably worse than our one-best recognition, and the term-recall is marginally better; however, the retrieval effectiveness is better despite the poor term precision.

When query expansion from the target collection is done, all results improve noticeably (column 3 in Table 5). Retrieval from our transcriptions is still better than retrieval from the medium-error baseline transcription, and is at par with retrieval from human transcriptions. Retrieval effectiveness on expanded documents gets to 0.53 average precision and it actually surpasses the retrieval from human transcriptions (0.5083) by 4.3%. This results is very encouraging. Our conservative collection enrichment hurt us in this environment and our official runs (column 5 in Table 5) are all lower than if we had expanded just using the target collection. However, the official runs are still very competitive, both our full-SDR runs att-s1 and att-s2 are above median for 20 out of 23 queries. The run using expanded documents att-s2 yields 7 best results in 23 which is quite a substantial proportion.

All these results point us to the possible advantages of doing document expansion in speech retrieval environments. These results also tend to confirm our belief that term recall plays a more important role in retrieval effectiveness for speech than term precision. We can afford to lose term precision for better term recall and this should yield improved retrieval effectiveness for speech. A larger query set would have made these results much more robust, but there is a clear pattern in Table 5 indicating that document expansion consistently yields better results than not doing it.

3.3 Cross-Recognizer Analysis

After the official conference, we did a more rigorous study of document expansion. We first discovered that constraining document expansion to allow only terms from the word-lattices generated by our recognizer held no additional benefit over not doing so. *I.e.* we can do document expansion only from NA news and the results were equally good or better. This also allows us to test document expansion for retrieval from the automatic transcriptions provided by other SDR track participants, for which we don't have the word-lattices. Secondly we found that the query expansion parameters used in our SDR runs were sub-optimal. Using the standard set of parameters used in our ad-hoc run yields consistently better results, and has the added advantage of uniformity of parameters.

We test document expansion on different automatic transcriptions provided to NIST by various track participants. Table 6 lists these transcriptions along with their word error rates. We cleaned some timing mistakes in our transcripts and used the correct WER scripts to get the 31% WER reported in Table 6 (as opposed to the 32.4% WER reported in Table 4). Here are the steps involved in document expansion:

Transcript	Unexpanded Documents				Expanded Documents			
	Baseline	Query expn. from		Coll.	Baseline	Query expn. from		Coll.
	dnb.dtn	target	NA News	Enrich.	*dnb.dtn*	target	NA News	Enrich.
ltt	0.4595	0.5300	0.5211	0.5327	0.5108	0.5549	0.5334	0.5614
cuhtk-s1	0.4376	0.5035	0.5202	0.5285	0.5220	0.5372	0.5444	0.5549
dragon98-s1	0.4190	0.5100	0.5227	0.5147	0.5061	0.5284	0.5459	0.5483
att-s1	0.4353	0.5020	0.5128	0.5251	0.5080	0.5343	0.5505	0.5452
nist-b1	0.4104	0.4820	0.4987	0.4954	0.4862	0.5259	0.5314	0.5316
shef-s1	0.4073	0.4890	0.5042	0.5085	0.5068	0.5421	0.5355	0.5399
nist-b2	0.3352	0.3965	0.4602	0.4220	0.4377	0.4743	0.4961	0.4940
derasru-s2	0.3633	0.3962	0.4614	0.4419	0.4585	0.5065	0.5118	0.5199
derasru-s1	0.3236	0.3613	0.4604	0.4188	0.4526	0.4849	0.5045	0.4959

Table 7: Cross-recognizer analysis.

1. Find documents related to a speech document. We do this by running the automatic transcription of the speech document as a query (raw-$tf \times idf$ weighted) on the NA News corpus and retrieving the *ten* most similar documents. In other words, we use the ten nearest neighbors of the speech document in this process. The documents are weighted by raw-$tf \times idf$ when used as a query because we found that nearest neighbors found using raw-$tf \times idf$ weighted documents yield the best expansion results.

2. The speech transcriptions are then modified using Rocchio's formula.

$$\vec{D}_{new} = \vec{D}_{old} + \frac{\sum_{i=1}^{10} \vec{D}_i}{10}$$

where $\vec{D}_{old}$ is the initial document vector, $\vec{D}_i$ the the vector for the i-th related document, and $\vec{D}_{new}$ is the modified document vector. All documents are *dnb* weighted (see Table 1). New words are added to the document. For term selection, the Rocchio weights for new words are multiplied by their *idf*, the terms are selected, and the *idf* is stripped from a selected term's final weight. Furthermore, to ensure that this document expansion process doesn't change the effective length of the document vectors, and change the results due to document length normalization effects, we force the total weight for all terms in the new vector to be the same as the total weight of all terms in the initial document vector. We expand documents by 100% of their original length (*i.e.* if the original document has 60 indexed terms, then we add 60 new terms to the document).

The results for unexpanded as well as the expanded documents are listed in Table 7. The two main highlights of these results are:

- document expansion yields large improvements across the board, and
- document expansion reduces the performance gap between retrieval from perfect and automatic transcriptions.

These points are highlighted in Figure 1. The left plot shows the average precision on the y-axis, against the WER on the x-axis. All number plotted in Figure 1 are for the unexpanded queries (*i.e.* we use the columns marked *Baseline* in Table 7). This prevents effects of query expansion from affecting these graphs and allows us to study the effects of document expansion in isolation. The horizontal lines are for human transcriptions whereas the other lines are for the different automatic transcriptions. As we can see in the left graph, document expansion (solid lines) yields large improvements across the board for this task over not doing document expansion (dashed lines).

The right graph in Figure 1 plots the %-loss from human transcriptions on the y-axis for unexpanded and expanded documents. The baseline for the expanded documents is the expanded human transcriptions, *i.e.* the solid horizontal line on the left graph. We observe that for the poorest transcriptions (DERASRU-S1) document expansion yields an improvement of an impressive 40% (over 0.3236) and reduces the performance

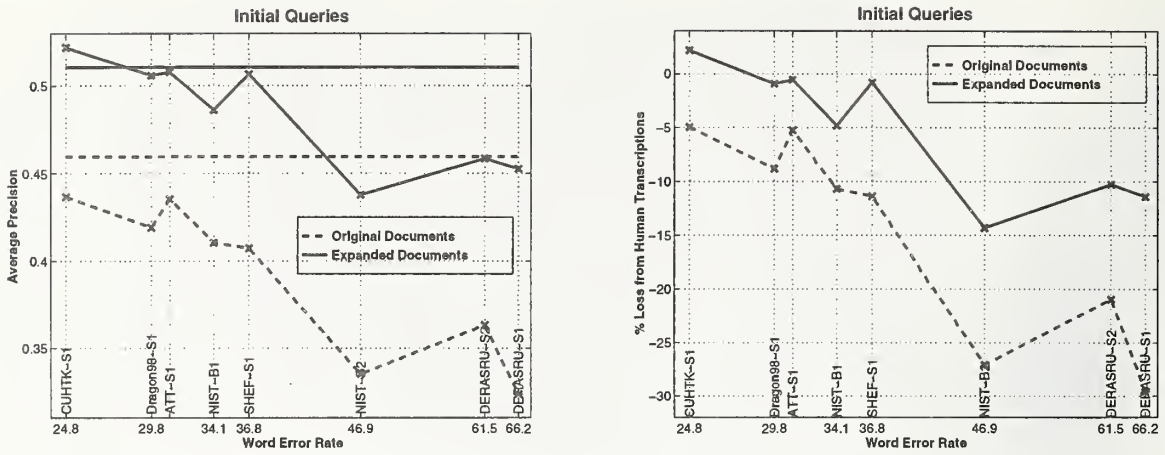


Figure 1: Raw average precision and %-loss from human transcriptions (initial user queries).

gap from human transcription to about 12% instead of the original 30% despite the very high baseline used. The results are similar for other transcriptions.

These results indicate that document expansion is indeed a very useful tool for retrieval from this speech corpus. We should caution that this is a very small test collection and more experimentation is needed, possibly on a larger test collection, before the full effect of document expansion could be analyzed in details.

4 VLC Runs

We come to the VLC track with the belief that with time the collection sizes will outgrow our capability to efficiently maintain a single index for a collection. We already see this happening on the Web. A recently published study reports that any single Web search engine covers less than 35% of the Web. However, if the collections from various engines are somehow pooled, the coverage improves dramatically. [6] For this reason, meta-search systems, systems that search multiple search engines and optionally merge the results are becoming increasingly available on the Web.

Our VLC track participation is modeled like a meta-search system. We divide the 100G collection into twenty independent collections of about 5G each. Each query is submitted in parallel to each of the twenty collections and their retrieval results are processed in various possible ways to obtain a single final ranking for the whole collection. We assume that each small collection is available through a *server*, and a user submits queries on a *client* which passes them to the servers, gathers the results and merges them into a single ranked list for presentation. We further assume that the client has access to some text collection for computation of *idf* values needed in our merging process described below. To make things closer to reality, we force the client to use an entirely different collection (TREC disks 1–5) as its *idf* collection since many clients will not have any Web collection (the target collection for the task) to gather *idf*s.

Four runs were submitted, each with a different set of assumptions to study different effects. Here is a brief description of the runs. Each server returns its top 20 documents to the client and the client generates a single ranked list for the 400 documents (20 each from 20 servers).

- **att98vi**: This run assumes the following:

  1. Each server is running a straight vector match *dnb* documents and *dtn* queries (see Table 1). No pseudo-feedback or query expansion is done at the servers' end.
  2. Each server returns the first 500 bytes (mark-up removed) for each of the top 20 documents to the client. This simulates the Web search environment where it is commonplace for engines to return the initial portion of the documents as a summary on the results page.

3. The client indexes the 400 summaries "on-the-fly" and creates *dnb* documents. The client also indexes the query using *dtn* weighting (*idf* is picked from the TREC collection).
4. The summaries are ranked by the client using a straight vector match, and the rank of a summary defines the rank of the final document. No pseudo-feedback or query expansion is done at the client's end.

Result: Average P@20 0.3570

- **att98vf**: This run is the same as **att98vi** except that servers return full document text instead of the initial 500 bytes and full document text is used by the client to produce the final ranking.

  Result: Average P@20 0.5030

- **att98vie**: This run has the following assumptions:

  1. Each server expands the query using pseudo-feedback.
  2. The client also expands the query in parallel using pseudo-feedback on its local collection (TREC).
  3. Each server returns the first 500 bytes (summary) for documents retrieved using the expanded query from step 1.
  4. The client indexed the 400 summaries "on-the-fly".
  5. The summaries are ranked by the client using the expanded query from step 2.

  Result: Average P@20 0.3750

- **att98vfe**: This run is the same as **att98vie** except that the servers return full document text (instead of the initial 500 bytes) which is used by the client to produce the final ranking.

  Result: Average P@20 0.5870

We wanted to show that merging based on the initial 500 bytes is not much worse from merging based on the full text of the documents. This indeed was the result in our internal studies in which we split the TREC database into several servers and evaluated our merging methods with TREC queries. However, we are disappointed to see that our run att98vi is noticeably worse than the corresponding full-text run att98vf. Similarly att98vie is noticeably worse than att98vfe. We would like to investigate this discrepancy in behavior for the TREC documents and the Web documents. On first thought, one is inclined to believe that Web documents are just different than more traditional TREC documents, and the initial part of a Web document is not a very good representative of the entire document (whereas for the TREC documents it is), but we would like to study this effect further.

5 Filtering Runs

We used the TREC-7 filtering track to test an alpha version of ATTICS, our toolkit for machine learning of classifiers for mixed textual and non-textual information. ATTICS differs from most text retrieval software in its support for numeric and other formatted data, and in its emphasis on classification of streams of data rather than retrieval from a static or slowly changing collection. It differs from most machine learning software in its efficient and flexible support for textual data, particularly in mapping from structured documents to attribute vectors.

ATTICS can be used either as a stand-alone system or as a C++ library that can be embedded in other applications. An extensive API and careful use of the C++ class system enable new preprocessors, data transformations, classifier types, training methods, and output formats to easily be added to the system. We used ATTICS in stand-alone mode for the TREC-7 filtering task. Support in ATTICS for incremental training of classifiers is not yet complete, so only the batch filtering and routing tasks were attempted. This also meant that in the batch filtering task we did not take advantage of training on retrieved test documents.

ATTICS uses XML internally for document mark-up. Since the SGML mark-up for TREC data obeys XML conventions, processing of TREC data was trivial. For the quick experiment reported here we mapped

the HEAD and TEXT fields (lower-cased) of the AP documents to a single vector of raw *tf* counts. Stemming and stop-wording have not yet been implemented and so were not used.

Early runs on TREC and other large data sets showed preprocessing to be unacceptably slow. The problem was traced to the extensive use of C++ streams for communication between preprocessor modules, a holdover from a prototype in the Unix pipeline style. A reimplementation achieved a rate of 220MB/hr for creation of on-disk vectors from on-disk text using one R10000 processor of an SGI Challenge XL with 8GB of RAM. This is still well below the comparable rate (about 3GB/hr) for our latest modification of SMART, but ATTICS can process a much richer set of data types, and the version used was not yet fully optimized.

ATTICS development to date has focused almost exclusively on basic systems issues, data modeling, API design, and so on, with little time left for implementing particular learning algorithms. To meet the TREC deadline, we did a quick implementation in ATTICS of the original Rocchio algorithm. [8] We trained linear models for each of the 50 TREC-7 filtering topics using only the judged AP88 documents, with Rocchio parameters $\alpha = 0$ (topic descriptions were not used), $\beta = 1$, and $\gamma = 1$. Negative Rocchio weights were zeroed out, as usual. Within document weights for all training and test data were computed using SMART *Lnu* style normalization with a slope parameter of 0.2. [12]

This first set of linear models was used to retrieve the top 5000 scoring AP88 documents. Then a second set of linear models was trained using the union of the judged AP88 documents and the top 5000 AP88 documents, with unjudged documents in the top 5000 being treated as non-relevant, *i.e.* a query zoning approach. [13] The second set of linear models were run back over the training documents to score them, and a threshold for each model was chosen that optimized the desired effectiveness measure (filtering measure F1 or F3). The models and thresholds were then applied to the test (AP89-90) documents, with the documents exceeding the threshold constituting the submitted set for batch filtering.

Our batch filtering submissions showed reasonable effectiveness. Run att98fb5, optimizing measure F1, was at or above median utility for 36 of 50 topics, with 16 tied for best. Run att98fb6, optimizing measure F3, had 32 of 50 topics at or above median utility, with 11 tied for best. Statistical significance tests showed the effectiveness of these runs to not be significantly different from that of the best submitted batch filtering runs [3]. This was a bit of a surprise to us, given our omission of feature selection, stop lists, *idf* weighting, dynamic feedback optimization, and other modern enhancements to Rocchio-style training.

On the other hand, our routing run using the scores output by the same Rocchio models (att98fr4) was not state-of-art. Overall average precision was 0.275, and per-topic average precision was above median for only 23 of 50 topics, with 2 bests. For comparison, we submitted a routing run (att98fr5) based on a modern augmented Rocchio approach implemented in SMART. The algorithm was a scaled-down version of our TREC-6 run att97rc, and did not use word cooccurrence pairs as features. [11] Overall average precision for this run was 0.419, with per-topic average precision at or above median for 41 of 50 topics, with 5 best. Incorporating more modern training methods into ATTICS is a priority for us, and should improve both filtering and routing effectiveness.

6 Conclusions

We are encouraged by our SDR performance, especially by the possible advantages of document expansion in this environment. We are quite satisfied by our filtering performance given the initial developmental stage of the software we are using. We would like to further study meta-searching as a model for searching very large collections. We simplified our adhoc algorithm over the complex algorithm we used last year and the results are still very good.

Acknowledgments

We are thankful to Anna Litvinova, Mandar Mitra, Marcin Kaszkiel, Yoram Singer, and Dan Stern for their help with various aspects of this work. We are also very grateful to Andrej Ljolje, Mehryar Mohri, and Michael Riley for their help in building the recognizer for the SDR track data.

References

[1] Chris Buckley. Implementation of the SMART information retrieval system. Technical Report TR85-686, Department of Computer Science, Cornell University, Ithaca, NY 14853, May 1985.

[2] Julia Hirschberg and Christine Nakatani. Using machine learning to identify intonational segments. In *Proceedings of the AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, Palo Alto, CA, March 1998.

[3] David A. Hull. The TREC-7 filtering track: Description and analysis. In this volume.

[4] S.M. Katz. Estimation of probabilities from sparse data from the language model component of a speech recognizer. *IEEE Transactions of Acoustics, Speech and Signal Processing*, pages 400–401, 1987.

[5] K.L. Kwok. Improving two-stage ad-hoc retrieval for short queries. In *Proceedings of the Twenty First Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 250–256. Association for Computing Machinery, New York, August 1998.

[6] Steve Lawrence and C. Lee Giles. Searching the World Wide Web. *Science*, 280(5360):98, 1998.

[7] Fernando C. N. Pereira and Michael D. Riley. Speech recognition by composition of weighted finite automata. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing*, pages 431–453. MIT Press, Cambridge, Massachusetts, 1997.

[8] 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.

[9] Gerard Salton, editor. *The SMART Retrieval System—Experiments in Automatic Document Retrieval.* Prentice Hall Inc., Englewood Cliffs, NJ, 1971.

[10] Kristie Seymore and Ronald Rosenfeld. Scalable backoff language models. In *ICSLP'96*, volume 1, 1996.

[11] Amit Singhal. AT&T at TREC-6. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*, pages 215–226, 1998.

[12] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29. Association for Computing Machinery, New York, August 1996.

[13] Amit Singhal, Mandar Mitra, and Chris Buckley. Learning routing queries in a query zone. In *Proceedings of the Twentieth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 25–32. Association for Computing Machinery, New York, July 1997.

Okapi at TREC–7: automatic ad hoc, filtering, VLC and interactive track

S E Robertson* S Walker† M Beaulieu‡

Advisers: E Michael Keen (University of Wales, Aberystwyth), Karen Sparck Jones (Cambridge University), Peter Willett (University of Sheffield)

1 Summary

Automatic ad hoc

Three runs were submitted: medium (title and description), short (title only) and a run which was a combination of a long run (title, description and narrative) with the medium and short runs. The average precision of the last mentioned run was higher by several percent than any other submitted run, but another participant[1] recently noticed an impossibly high score for one topic in the short run. This led to the discovery that due to a mistake in the indexing procedures part of the SUBJECT field of the LA Times documents had been indexed. Use of this field was explicitly forbidden in the guidelines [1] for the ad hoc track. The official runs were repeated against a corrected index, and the corrected results are reported below, average precisions being reduced by about 2–4%.

Adaptive filtering

This year saw a major departure from the previous Okapi approach to routing and filtering. We concentrated our efforts on the twin problems of (a) starting from scratch, with no assumed history of relevance judgments for each topic, and (b) having to define a threshold for retrieval. The thresholding problem is interesting and difficult; we associate it with the problem of assigning an explicit estimate of the probability of relevance to each document. We describe and test a set of methods for initializing the profile for each threshold and for modifying it as time passes. Two pairs of runs were submitted: in one pair queries remained constant, but in the other query terms were reweighted when fresh relevance information became available.

VLC track

Four runs on the full database were submitted, together with one each on the 10% and 1% collections. Unexpectedly, unexpanded queries did better than expanded ones; the best run used all topic fields and some adjacent term pairs from the topics. The best expanded run used one of the TREC–7 ad hoc query sets (expanded on disks 1–5).

Interactive track

Two pairwise comparisons were made: Okapi with relevance feedback against Okapi without, and Okapi without against ZPrise without. Okapi without performed somewhat worse than ZPrise, and Okapi with only partially recovered the deficit.

*Microsoft Research Ltd, 1 Guildhall Street, Cambridge CB2 3NH, UK, and City University, London, UK. email ser@microsoft.com

†Microsoft Research Ltd, 1 Guildhall Street, Cambridge CB2 3NH, UK. email sw@microsoft.com

‡Department of Information Studies, University of Sheffield, Western Bank, Sheffield S10 2TN, UK. email m.beaulieu@sheffield.ac.uk. *All three previously at:* Centre for Interactive Systems Research, Department of Information Science, City University, Northampton Square, London EC1V 0HB, UK.

[1]David Hawking of CSIRO, Canberra

2 Okapi at TRECs 1–6

The Okapi search systems used for TREC are descendants of the Okapi systems developed at the Polytechnic of Central London[2] between 1982 and 1988 under a number of grants from the British Library Research & Development Department and elsewhere. These early Okapi systems were experimental highly-interactive reference retrieval systems of a probabilistic type, some of which featured automatic query expansion [2, 3, 4].

All the Okapi work in connection with TRECs 1–6 was done at City University, London. Most of the Okapi TREC–7 entries were done from Microsoft Research, Cambridge (UK).

For TREC–1 [5], the low-level search functions were generalized and split off into a separate library — the Okapi Basic Search System (BSS). User interfaces or batch processing scripts access the BSS using a simple command language-like protocol.

Our TREC–1 results were very poor [5], because the classical Robertson/Sparck Jones weighting model [6] which Okapi systems had always used took no account of document length or within-document term frequency.

During TREC–2 and TREC–3 a considerable number of new term weighting and combination functions were tried; a runtime passage determination and searching package was added to the BSS; and methods of selecting good terms for routing queries were developed [7, 8]. During the TREC–2 work "blind" query expansion (feedback using terms from the top few documents retrieved in a pilot search) was tried for the first time in automatic ad hoc experiments, although we didn't use it in the official runs until TREC–3. Our TREC–3 automatic routing and ad hoc results were both relatively good.

TREC–4 [9] did not see any major developments. Routing term selection methods were further improved.

By TREC–5 many participants were using blind expansion in ad hoc, several with more success than Okapi [10, 11]. In the routing, we tried to optimize term weights after selecting good terms (as did at least one other participant); our routing results were again among the best, as were the filtering track runs.

In TREC–6 [12] we continued to investigate blind expansion, with continuing mixed results. We also introduced a new weighting function designed to make use of documents known or assumed to be non-relevant. In routing and filtering we continued to extend the optimization procedure, including a version of simulated annealing. Again our routing and filtering results were among the best.

3 The system

3.1 The Okapi Basic Search System (BSS)

The BSS, which has been used in all Okapi 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 as defined below (equation 1) and described more fully in [8, Section 3]. 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 were again no major changes to the BSS during TREC–7.

Weighting functions

All TREC–7 searches used varieties of the Okapi **BM25** function first used in TREC–3 (equation 1).

$$\sum_{T \in Q} w^{(1)} \frac{(k_1+1)tf}{K+tf} \frac{(k_3+1)qtf}{k_3+qtf} + k_2.|Q|.\frac{avdl-dl}{avdl+dl} \tag{1}$$

where

Q is a query, containing terms T
$w^{(1)}$ is either the Robertson/Sparck Jones weight [6] of T in Q

$$\log \frac{(r+0.5)/(R-r+0.5)}{(n-r+0.5)/(N-n-R+r+0.5)} \tag{2}$$

[2]Now the University of Westminster.

or else a slightly modified and more general version which takes account of non-relevance as well as relevance information [14]

$$\frac{k_5}{k_5+\sqrt{R}}(k_4+\log\frac{N}{N-n})+\frac{\sqrt{R}}{k_5+\sqrt{R}}\log\frac{r+0.5}{R-r+0.5}-\frac{k_6}{k_6+\sqrt{S}}\log\frac{n}{N-n}-\frac{\sqrt{S}}{k_6+\sqrt{S}}\log\frac{s+0.5}{S-s+0.5} \quad (3)$$

N is the number of items (documents) in the collection
n is the number of documents containing the term
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 nonrelevant to a specific topic
s is the number of nonrelevant documents containing the term
K is $k_1((1-b)+b.dl/avdl)$
k_1, b, k_2, k_3 and k_4 are parameters which depend on the on the nature of the queries and possibly on the database. For the TREC–7 experiments, k_1 was 1.2 and b was 0.75 except where stated otherwise; k_2 was always zero and k_3 anything from 0 to 1000; when there was not much relevance information -0.7 was a good value for k_4, otherwise zero.
k_5 and k_6 determine, in equation 3, how much weight is given to relevance and non-relevance information respectively. Typical ranges are 0–4 for k_5 and 4–∞ for k_6
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
dl and *avdl* are the document length and average document length (arbitrary units) resp.

When k_2 is zero, as it was for all the results reported here, equation 1 may be written in the simpler form

$$\sum_{T\in Q} w^{(1)}\frac{(k_1+1)tf}{K+tf}\frac{(k_3+1)qtf}{k_3+qtf} \quad (4)$$

Nonrelevance information

The extension to the basic weighting formula given by equation 3 is motivated mainly by the desire to make use of explicit judgment of nonrelevance, rather than relying entirely on the "complement" method, by which all documents not known to be relevant are assumed to be nonrelevant. There is a fuller discussion in [14]. This formula might be used in various environments; the particular use reported here has to do with "blind" expansion, where there are no explicit judgments of relevance. Further detail is given below.

Term ranking for selection

In [8] there is a brief discussion of some alternative ways of ranking potential expansion terms. It appeared that no method was superior to the method proposed in [15] by which terms are ranked in decreasing order of $TSV = r.w^{(1)}$. In line with the "nonrelevance" version of $w^{(1)}$ (equation 3) Boughanem has proposed the more general function

$$TSV = (r/R - \alpha s/S).w^{(1)} \quad (5)$$

where $\alpha \in [0,1]$ and r, R, s and S are as above.

Passage determination and searching

Since TREC–3 the BSS has had facilities for search-time identification and weighting of any sub-document consisting of an integral number of consecutive paragraphs. It was described, and some results reported, in [8]. Passage searching almost always increases average precision, by about 2–10 percent, as well as recall and precision at the higher cutoffs. It often, surprisingly, reduces precision at small cutoffs, so is not used in pilot searches for expansion runs. Recently a mistake in the passage searching code, undetected since 1993, was found and corrected; some past TREC runs were repeated in the hope that results might be improved, but unfortunately the correction seems to make very little difference in practice.

3.2 Hardware

Apart from the interactive track almost all the TREC–7 processing was done at Microsoft Research, Cambridge. Most of the work was done on a 300 MHz Sun Ultra 10 with 256 MB and a Dell with two 400 MHz Pentium processors and 512 MB. Mainly to cater for the VLC track there was about 170GB of disks, most of which were attached to the Sun. Both machines were running Solaris 2.6. Some use was also made of a PC running Linux. The network was 100MHz ethernet.

3.3 Database and topic processing

For interactive purposes it is necessary to provide for the readable display of documents. Since we have not (yet) implemented a runtime display routine for SGML data, nor adequate parsing and indexing facilities, all the TREC input text is subjected to batch conversion into a uniform displayable format before further processing. This is done by means of shell scripts which are hacked up to suit the input dataset. For most of the TREC data, the processed records have three fields: document number, any content which may be useful for display but not for searching, and the searchable "TEXT" and similar portions. However, only two fields were used for the VLC98 collection.

All the TREC text indexing was of the keyword type. A few multiword phrases such as "New York", "friendly fire", "vitamin E" were predefined and there was a pre-indexing facility for the conflation of groups of closely related or synonymous terms like "operations research" and "operational research" or "CIA" and "Central Intelligence Agency". A stemming procedure was applied, modified from [16] and with additional British/American spelling conflation. The stoplist contained about 220 words.

Initial topic processing deleted terms such as "document", "describe(s)", "relevan...", "cite..." from any description, narrative or summary fields. What is left was then processed in the same way as text to be indexed.

4 Automatic adhoc

Again, there was no significant change in methods for TREC–7. Further experiments were done using adjacent term pairs from topic statements, but any gain was very slight. To try the effect of emphasizing the difference in weights between rare and common terms some runs were done with the w^1 termweights raised to a power $1 + \alpha$ for some $0 < \alpha \leq 0.5$. Where there was a fairly large number of terms in the query, values of α in the range 0.1–0.3 sometimes improved average precision and some other statistics by around 2% over 50 topics. However, any improvement in our results (Table 1) seems to have come from merging the output from a number of runs. In the past we have not obtained any consistently good results from merging ad hoc runs (although the technique has been extensively used in routing to compensate for the overfitting which results from query optimization), but some experiments on the TREC–6 data showed that a considerable gain in most of the TREC statistics could have been obtained. Thus, the best Okapi TREC–7 run, ok7ax, is a linear combination in the proportions 2:2:1 of runs derived from "long", "medium" and "short" queries, document weights of the source runs having first been normalized by dividing by the mean weight. Merging *unexpanded* runs however did not improve average precision.

Table 1 gives the results of the official runs, and some others, after they were repeated using corrected indexes.[3] The average precisions for the original uncorrected runs are given for comparison. The best run, ok7ax, is now equal on average precision to the best non-Okapi TREC–7 run. In more detail, the effect of the correction on average precision for this run was as follows: for 19 topics the score was unchanged, 18 topics were worse (in one case, topic 397, the score went from 321 to 081), and the remaining 13 gained, but not by enough to compensate for the losses.

Comparing the expanded and non-expanded runs in Table 1 suggests that blind expansion worked rather well on the TREC–7 topics, giving gains of 23%–25% on runs derived from short and medium queries. For all the expanded runs the pilot search was on the full disks 1–5 database.

5 Adaptive filtering

5.1 General concepts

We assume for the filtering task the following scenario. There is a constant stream of incoming documents, and a number of requests represented by profiles. Profiles my be expected to remain in existence for some time, but to be modified, possibly at every incoming document. For each profile there will be a history, including information about

[3] The database used for generating expansion terms was also reindexed.

Table 1: Automatic ad hoc results. The first three are corrected versions of the official submitted runs; the *uncorrected* average precision is also given.

Run	AveP corr	AveP uncorr	≥ med	P10	P30	RPrec	Rcl
ok7ax (combination of al, am and as)	296	(303)	47	552	419	323	717
ok7am: expanded from title + desc	281	(288)	47	530	403	309	688
ok7as: expanded from title only	253	(261)	37	472	375	286	605
ok7al: expanded from title + desc + narr	284	(290)	43	548	418	310	700
as ok7al but no expansion	248	(254)	41	518	378	288	647
as ok7am but no expansion	226	(233)	37	474	367	269	607
as ok7as but no expansion	200	(208)	24	438	333	246	543
as ok7ax but no expansion	247		41	510	395	279	655

Table 2: Automatic ad hoc run parameter details

All pilot searches used the disks 1–5 database. The number (S in equation 3) of assumed nonrelevant documents was 500 throughout, and there was a gap G of 500 documents between R and S. All final, but not pilot, runs used passage retrieval.

	Pilot				Final						
Run	R	k_1	b	k_3	k_1	b	k_3	k_5	k_6	# terms	topic term bonus
ok7al	15	1.6	0.8	1000	0.8	0.8	1000	1	64	30	×2.5
ok7am	10	1.4	0.6	1000	1.4	0.6	1000	1	64	30	×2.5
ok7as	6	1.0	0.5	0	1.5	0.6	0	1	128	20 + title terms	×3.5

documents retrieved, and about user judgments on documents. More specifically, we assume that we are operating within the parameters laid down for the TREC–7 filtering track. That is: at time $t = 0$ the system is switched on, with both a set of topics and the start of a stream of documents. There are no previous relevance judgments for these topics, and no previous documents in the stream, although topic representations (queries) may be tested for e.g. indicative term frequencies or document scores against a different database.

The document collection

We will further assume that the document collection is cumulated: at any stage of the process, we may conduct a new search on the collection as it has accumulated up to now, or discover from this collection any information we need (such as, for example, information about term occurrences). This historical information is taken as a more-or-less reliable guide (at least after we have accumulated a reasonable number of documents) to the future behaviour of documents and terms. This assumption clearly ignores the possibility of substantial changes in usage over time, e.g. as might result from a sudden surge of interest in a particular topic.

Thresholding and probability of relevance

The filtering task requires the system to make a binary decision about each incoming document, as to whether or not to send it to the user. In systems based on ranking by means of a matching function, this binary decision can be cast in the form of a threshold on the matching score. Systems using such methods often produce scores on an arbitrary scale, since only the order matters; thresholding could then be treated as an empirical problem on this arbitrary scale. An alternative would be to give the score an absolute value, perhaps in terms of probability of relevance.

The evaluation criteria defined for the TREC–7 filtering task make the connection explicit, by being expressed in terms of an absolute marginal probability of relevance. In what follows, the two problems (of setting thresholds and of obtaining absolute probability scores) will be regarded as strongly linked.

Adapting from sparse data

In the TREC–7 adaptive filtering task, the system needs to be able to adapt to small amounts of information. This adaptation should make use of non-evaluative information (essentially size of output and collection characteristics)

as well as relevance judgments. While the old routing experiments allowed us to take a very empirical approach to optimizing queries on the large training set, without strong dependence on models, the adaptive case will require a very strong model base, to compensate for the sparsity of the information.

Probability of relevance in the probabilistic model

The probability of relevance will be denoted $P(R|D)$. Here D is taken to be all the information we have about a document – its description (a specific query is assumed).

The probabilistic model of retrieval is of course based on the probability of relevance, so one might expect that it would be easy to obtain an estimate of that probability within a probabilistic system. However, this is not the case. The traditional argument of the probabilistic model relies on the idea of ranking: the function of the scoring method is not to provide an actual estimate of $P(R|D)$, but to rank in $P(R|D)$ order.

In practice the document score calculated in Okapi (probably as in many other probabilistic systems) might be assumed to have some connection to the probability of relevance, but the exact functional relation is not derivable from the model. At best, we might assume that the score is linear in the log-odds of relevance, with the required two parameters unknown.

In the next two sections, we discuss the details of a method of defining and modifying thresholds in a filtering process. One objective is that the thresholds should be interpretable in terms of probability of relevance, so that the TREC utility functions can be used directly; hence the section on calibration. A second is that the method should minimise the twin dangers of getting nothing at all or far too much (the "selectivity trap"). This motivates some aspects of the calibration method and some of adaptation. Finally, the method must allow the threshold to be adaptive; indeed, it must adapt sensibly to changes in the query (e.g. reweighting or adding new terms) as well as to feedback from the user.

5.2 Calibration – basic ideas

Logistic regression

The one method that has been tried to calibrate retrieval scores, so that they might be interpreted as probabilities, is logistic regression on a training set [17]. We have used exactly this method, which fits well with the probabilistic model. The object is solely to calibrate a predefined scoring function, not (as in the Berkeley work) to improve the scoring function itself.

A reference point for the score

Scores in the Okapi system tend to be rather unpredictable, in the sense that different queries will have top-scoring documents in very different ranges of scores. In order to avoid the selectivity trap, we need to relate the score for a given query to some identifiable reference point. The two parameters we have considered for the purpose of defining this reference point are (a) the theoretical maximum score, and (b) the average score of the top 1% of the ranking (*ast1*). The first has the advantage of being definable at time $t = 0$, before we have any results; the second has the advantage of being strongly related to the actual behaviour of the particular query, in the part of the scale that we are interested in. We should achieve a reliable estimate of *ast1* after we have seen one or two thousand documents; it is a distributional parameter which does not depend on the size of the sample, so that when we have enough data to estimate it, the estimate should be reasonably stable.

Some preliminary experiments suggest that *ast1* is a slightly better reference point to use than the theoretical maximum score. Under TREC conditions, we cannot estimate it until a couple of weeks into the simulated time period, though under realistic conditions we might expect to be able to estimate it from the beginning. For the TREC experiment we have used the theoretical maximum score for the first week, and then switched to *ast1*.

Initial calibration

The method therefore involves using a model of the form

$$\log O(R|D) = \beta + \gamma \frac{score}{ast1} \tag{6}$$

where β and γ are initially estimated by logistic regression on a training set. Under TREC rules this has to be different sets of both documents and queries from the filtering test; we have used a collection consisting of disks 1–3

without AP and topics 51-150 (further details are given below). A second regression on the same set gives a model of the form

$$ast1 = \alpha_1 + \alpha_2\, maxscore$$

where *maxscore* is the theoretical maximum score. This is used in the first week only to substitute for *ast1* in equation 6.

Given a document score and an estimated *ast1*, equation 6 can be used to estimate the log-odds of relevance of any specific document, to be compared to an absolute threshold determined by the desired utility measure. The result of applying the equation to the score for document D_i will be denoted c_i (for calibrated score). c_i is on a log-odds scale, but can be converted back to a probability p_i:

$$\begin{aligned} c_i &= \beta + \gamma \frac{score_i}{ast1} \\ p_i &= \frac{\exp c_i}{1 + \exp c_i} \end{aligned} \tag{7}$$

for some estimated β, γ and *ast1*.

5.3 Calibration – adaptation

Introduction

The initial calibration defined above will need adapting according to the output from the profile (in a profile-specific fashion). It is assumed that as time passes and documents are matched against the profile, the estimate of *ast1* will be adjusted. Any change to the profile will also require re-estimation of *ast1*, *via* a recalculation of the scores of previously-considered documents (in other words, the new profile will be run as a query against the accumulated collection for this purpose).

In addition, the adaptation mechanism must be capable of responding to either quantitative or qualitative evidence concerning the performance of the threshold setting. The qualitative method suggested here is intended to deal with both qualitative evidence and one direction of quantitative evidence. It assumes that relevance feedback information (positive or negative) is quickly obtained on all items submitted to the user, and that the submission of too many documents will be reflected in the relevance judgments. Quantitative adaptation in the other direction (nothing retrieved) is combined with another aspect – see section 5.4 below.

Qualitative calibration

Given relevance feedback data for a profile, probably a relatively small amount, it may be hard to get any reliable new estimate of γ in equation 6. However, an adaptive approach to the intercept β only may be appropriate. Since equation 7 predicts an explicit probability of relevance for every submitted document, we may compare these predictions against the outcome and use any systematic error to adjust β.

Suppose the user has seen and judged j items, of which r are relevant. Then a straight maximum-likelihood estimate for β (assuming γ given) would be obtained by solving the equation

$$r - \sum_{i=1}^{j} p_i = 0$$

Substituting for p_i via equation 7, and treating β as the single variable, we may apply a Newton-Raphson (gradient descent) method to improve the initial estimate of β.

A problem with this approach (a general characteristic of such maximum likelihood estimators) is that it goes haywire if either $r = 0$ or $r = j$; these events are very likely when we have only a small amount of relevance information. In order to deal with this problem, we may seek a Bayesian estimator which constrains the estimate by means of some prior distribution, so that it remains within sensible bounds, and allows the evidence to dominate only when there is enough of it. One way to achieve this is to assume that we have some (m) mythical documents whose estimated probabilities of relevance are correct; the value of m will determine the strength of the prior constraint. These mythical documents will have most effect on the estimation process if they are near the centre of the probability scale, and in what follows, they are assumed to have the score that would make their estimated probabilities of relevance equal to 0.5. Thus instead of j assessed and r relevant documents, we have $(j+m)$ assessed and $(r + \frac{1}{2}m)$ relevant documents, and $p_{j+1} = p_{j+2} = \ldots = p_{j+m} = 0.5$.

Including these mythical documents and spelling out the gradient descent method in full, we suppose an iterative sequence of estimates $\beta^{(n)}$ and corresponding values $c_i^{(n)}$ and $p_i^{(n)}$ for each document. Then the gradient descent formula is:

$$\beta^{(n+1)} = \beta^{(n)} + \frac{r - \sum_{i=1}^{j} p_i^{(n)} + m \frac{1-\exp(\beta^{(n)}-\beta^{(0)})}{2(1+\exp(\beta^{(n)}-\beta^{(0)}))}}{\sum_{i=1}^{j} \frac{p_i^{(n)}}{1-p_i^{(n)}} + m \frac{\exp(\beta^{(n)}-\beta^{(0)})}{(1+\exp(\beta^{(n)}-\beta^{(0)}))^2}} \tag{8}$$

$\beta^{(0)}$ is the initial estimate provided by the original regression equation 6.

This method for dealing with the qualitative calibration problem also addresses one side of the quantitative problem: if the profile retrieves far too much, many of them are likely to be irrelevant, and the method will raise the threshold.

In the experiments, one iteration only was performed at each stage (simulated week), and only when new documents have been assessed; however, successive stages are cumulative, and at each stage all previously assessed documents are included in the estimation process. This procedure represents a very simple-minded approach to the normal iterative estimation suggested by the above argument.

5.4 The value of feedback

Given that we have an explicit estimate of probability of relevance for any document, and a user-defined utility criterion interpreted as a threshold probability, the obvious strategy is simply to set this threshold from the word go. This, however, ignores a significant factor: every document assessed as relevant by the user provides additional information, from which we may make better estimates of probability of relevance for future documents. Therefore there may be long-term gains from lowering the threshold initially. We have at present no way of quantifying these possible gains; our solution to this problem, highly adhoc and pragmatic, is to define a "ladder" of thresholds. A profile starts at the bottom of the ladder, and climbs one rung for every relevant document found.

The ladder can be defined in terms of probability or log-odds. Two of the points correspond to the thresholds defined for the TREC filtering track as F1 and F3. Thus the target position on the ladder depends on this user-specific utility function. The ladder used is given in Table 3; it must be stressed that the points on it were pulled out of a hat, and do not in any way reflect any quantitative argument.

Table 3: The Ladder: selection thresholds

A profile starts at the bottom, and climbs one rung of the ladder for each relevant document found, up to the target utility point.

$P(R\|D)$	$\log O(R\|D)$	Utility point
0.5	0	
0.4	-0.4	F1
0.3	-0.9	
0.2	-1.4	F3
0.13	-1.9	
0.1	-2.2	
0.07	-2.6	
0.05	-2.9	
0.04	-3.2	

The top point is provided for fine tuning purposes as part of the experiment (to be described). In the circumstances of the TREC filtering task, an additional constraint applies: because the initial estimate (based on *maxscore* rather than on *ast1*) may be unreliable, and may in particular lead to the retrieval of many too many documents, the threshold is kept high (at the appropriate utility point) for the first week. When a direct estimate of *ast1* becomes available, the ladder is brought into effect, and the threshold is moved down to the appropriate place (possibly above the bottom if we found some documents in the first week).

The use of the ladder should have another beneficial side-effect. If we are in danger of retrieving nothing, setting the threshold low initially is likely to avoid the problem. We would not then be raising the threshold until our calibration has improved.

5.5 Model-free threshold discovery

The above arguments are intended to be used in the early life of a profile, when little relevance information is available. In past TRECs, in the routing experiment (where each profile is deemed to be already mature), we achieved good results by iterative optimisation of the queries. This technique was then extended to threshold setting for filtering [12], by running the optimised query retrospectively against the training set, and observing which threshold maximised the utility function concerned (the earliest or highest such threshold in the case of ties). Neither of these processes (iterative optimisation of query or threshold discovery) makes any appeal to the probabilistic model; instead, it treats the utility function as an arbitrary performance criterion to be optimised.

In the experiments to be described, we have attempted no iterative optimisation, although it would probably be desirable to do so at some point. However, in one of the runs below, where we have a reasonable amount of feedback for a particular profile, we initiated a similar threshold-discovery process on the current query.

5.6 Experiments

Initial queries (profiles)

Profiles were derived from the title, concepts, description and narrative fields of topics 51–150, using a database consisting of disks 1–3 without AP (835,248 documents) to derive term weights. The matching function used was BM25 with $k_1 = 2.0$, $b = 0.8$ and $k_3 = 1000$.

Initial calibration

Each topic was searched on this collection, and the top ranked documents were retrieved: the number was specified as 1% of the collection, namely 8352 documents. These were assigned their relevance values as defined for TREC; unjudged documents were assumed irrelevant. The entire set of 835,200 observations was put through the SPSS logistic regression program, with relevance as the dependent variable and $\frac{score}{ast1}$ as the single independent variable. The result was $\beta = -6.77$ and $\gamma = 2.68$. These results are in fact very typical – in a number of different experiments with different topics and documents, we usually obtained very similar results. However, one factor which made a significant difference was the chosen cutoff; if we stopped much earlier, say 1000 documents, we would get a smaller γ and a smaller absolute value of β. Further investigation of the statistical relationships here is desirable.

A regular (not logistic) regression of *ast1* (dependent) against *maxscore* (independent) gave $\alpha_1 = 55$ and $\alpha_2 = 0.036$. These four values were used in the experiment: α_1 and α_2 for the first week, γ throughout the experiment, and $\beta = \beta^{(0)}$ as initial value and seed for the adaptive method.

Adaptive procedure

Documents were processed initially in weekly batches. For speed, after the first six weeks, batches became four-weekly until the end of 1989; as there was no relevance information for 1990 the whole of this year was run as a single batch.

For the first week, the threshold was defined by the appropriate position on the ladder for the utility function being used, and the maximum score was used to provide an estimate for *ast1* and thereby for c_i for each document. From the following week, a direct estimate of *ast1* is available, and the usual ladder rule applied: starting at the bottom, each profile was moved up one notch for each relevant document found. (As some profiles will have found relevant documents in the first week, these ones will never actually be at the bottom of the ladder.) *ast1* is re-estimated from the accumulated collection for the first six weeks.

After each week in which some documents have been found for a profile (irrespective of relevance), the adaptive calibration of β is invoked. That is, for each previously seen document, a value of c_i is calculated according to the current profile and the current value of *ast1*, and one iteration only of the iterative formula 8 is then applied. The value of m in equation 8 was 5. The new value of β remains in force for this profile until the next invocation of the adaptive calibration.

Results

Two pairs of runs were submitted, ok7ff{13}2 and ok7ff{13}3; the first '1' or '3' refers to the utility function number. In the first pair the queries were retained unchanged throughout, so only the threshold was varied. In the second pair query terms were reweighted using equation 2 after each batch which contained one or more relevant documents.

Another difference was that in the first pair, the model-free threshold discovery process replaced β-adjustment and the Ladder threshold after at least 12 documents had been retrieved, of which at least 2 were relevant. This rather early invocation of the model-free process is likely to have resulted in the Ladder method being disabled for many topics. The second pair of runs did not make use of the model-free process.

Table 4 lists the number of topics obtaining at least the median score. For comparison, it also shows the results (relative to median scores) if no documents at all had been retrieved.

Table 4: Adaptive filtering results, number $\geq$ median

Condition	1988	1989	1990
Function F1, constant queries	28	40	40
F1, queries reweighted	28	35	31
F1, no output	42	25	42
F3, constant queries	35	38	42
F3, queries reweighted	35	39	39
F3, no output	24	24	35

With constant queries it appears that the threshold adaptation worked fairly well. The runs in which query terms were reweighted did not do so well. This requires further investigation. It is also worth noting that it is quite hard to do better than a simple "retrieve nothing" rule on a given utility function, particularly F1.

6 VLC

Source processing

Before indexing, the source text was reduced by removing lines starting with "Server:", "Content-type:", "Last modified:", etc, document numbers were then identified, followed by the removal of all text inside '< ... >'. Dates and URLs were retained, but not indexed. This reduced the indexable text by almost 50% to a little over 50 GB.

Examination of a few of the documents suggested that there was quite a lot of non-text material (compressed data etc). It was decided that it would not be practicable to remove (or avoid indexing) this material. This resulted in an index with a very large dictionary file of some 70 million terms most of which are nonsensical nonce-words, a typical sequence being "qetura", "qetutmz7", "qetuwuqgrslk79", "qetv", "qetv9pif0yk9", ... The total number of indexed tokens from the 18.6 million documents was about 5800 million (mean 312 per document), and the corresponding figure for types was 2600 million (140 per document). The main (keyword) index had a total size of about 34 GB, split into six portions. It contained full within-document positional information.

Results

Table 5 summarizes the results. It seems strange that expansion on this very large collection should produce such poor results. More investigation is needed. The use of pairs of adjacent terms from the topic statements seems to have been slightly beneficial. An adjacent pair from a single topic subfield qualified for inclusion if it occurred in the (full) database at least five times as often as the expected number of times; pair weight for a word-pair AB was $log(n(A \text{ AND } B)/n(A \text{ ADJ } B))$ [18].

7 Interactive Okapi experiment

The system used for the interactive experiment for TREC-7 was exactly the same as that used for TREC-6. Essentially the system offers the user an incremental query expansion facility based on the idea of a working query. The working query consists of a combination of user-entered terms and system-generated terms extracted during relevance feedback. Extracted terms are displayed incrementally whenever the user makes a positive relevance judgement, provided the term occurs in at least three relevant documents and has a selection value (TSV) equal to two-thirds the average TSV of all the terms which have appeared in the working query to date. The ranked working query appears in a separate window and the maximum number of terms is defaulted to twenty.

The aim in TREC-7 was to build on the experimental conditions of the previous round, where ZPrise without relevance feedback achieved higher precision than the Okapi system with relevance feedback but recall was better

Table 5: VLC results

Condition	Collection	P20
No expansion, pairs, all topic fields	full	598
As previous	10%	369
As previous	1%	188
No expansion, no pairs, all topic fields	full	588
No expansion, pairs, title and description	full	541
As previous	10%	376
As previous	1%	180
No expansion, no pairs, title and description	full	525
Expansion 30 docs, all topic fields	full	545
As previous	10%	357
As previous	1%	192
Expansion 15 docs, all topic fields	full	538
Queries as ad hoc run ok7al1, all topic fields	full	562
Expansion, title and description	full	509
As previous	10%	357
As previous	1%	202

in Okapi. The focus here was thus on a three way comparison between two versions of Okapi, one with relevance feedback and one without and with ZPrise as a control. In order to optimise on the number of searcher participants, the experiment was conducted on two sites, City and Sheffield. Eight subjects at each site conducted a total of eight searches each, four on each of two systems. Subjects at City carried out searches on the version of Okapi without feedback and on ZPrise, whilst at Sheffield they searched on both versions of Okapi, with and without feedback. Hence across the sites a total of sixty-four searches were carried out on Okapi without feedback and thirty-two on Okapi with feedback as well as thirty-two on ZPrise. Unfortunately due to the constraint within the experimental design of having to limit the number of systems subjects could search, only half as many searches were undertaken on the feedback version of Okapi than on the one without feedback.

All sixteen subjects on both sites were new to the systems concerned. They included eight masters and five doctoral students, as well as three research assistants all of whom were recruited within the respective information science departments.

The overall results for instance recall and precision for the three systems are given in Table 6.

Table 6: Interactive task results

	Relevance	Mean		Number
System	feedback	Recall	Precision	of searches
Okapi	No	.383	.653	64
Okapi	Yes	.397	.692	32
ZPrise	No	.399	.714	32

ZPrise clearly outperformed the Okapi system without relevance feedback on both measures. Although it also achieved better results than the version of Okapi with relevance feedback, recall was only very marginally better and the main difference is in the precision. We conclude that although Okapi with relevance feedback is better than with no relevance feedback, in an interactive environment the query expansion facility does not appear to be achieving the desired level of performance.

References

[1] http://trec.nist.gov/act_part/guidelines/7guides_adhoc.html

[2] Mitev, N.N., Venner, G.M. and Walker, S. *Designing an online public access catalogue: Okapi, a catalogue on a local area network.* British Library, 1985. (Library and Information Research Report 39.)

[3] Walker, S. and Jones, R.M. *Improving subject retrieval in online catalogues: 1. Stemming, automatic spelling correction and cross-reference tables.* British Library, 1987. (British Library Research Paper 24.)

[4] Walker, S. and De Vere, R. *Improving subject retrieval in online catalogues: 2. Relevance feedback and query expansion.* British Library, 1990. (British Library Research Paper 72.) ISBN 0-7123-3219-7

[5] Robertson, S.E. *et al.* Okapi at TREC. In: [19], p21–30.

[6] Robertson, S.E. and Sparck Jones K. Relevance weighting of search terms. *Journal of the American Society for Information Science* 27, May–June 1976, p129–146.

[7] Robertson, S.E. *et al.* Okapi at TREC–2. In: [20], p21–34.

[8] Robertson, S.E. *et al.* Okapi at TREC–3. In: [21], p109–126.

[9] Robertson, S.E. *et al.* Okapi at TREC–4. In: [22], p73–96.

[10] Beaulieu, M.M. *et al.* Okapi at TREC–5. In: [11], p143–165.

[11] *The Fifth Text REtrieval Conference (TREC–5).* Edited by E.M. Voorhees and D.K. Harman. Gaithersburg, MD: NIST, 1997.

[12] Walker S. *et al.* Okapi at TREC–6: Automatic ad hoc, VLC, routing, filtering and QSDR. In: [13], p125–136.

[13] *The Sixth Text REtrieval Conference (TREC–6).* Edited by E.M. Voorhees and D.K. Harman. Gaithersburg, MD: NIST, 1998.

[14] Robertson, S.E. and Walker S. On relevance weights with little relevance information. In *Proceedings of the 20th annual international ACM SIGIR conference on research and development in information retrieval.* Edited by Nicholas J Belkin, A Desai Narasimhalu and Peter Willett. ACM Press, 1997. p16–24.

[15] Robertson, S.E. On term selection for query expansion. *Journal of Documentation* 46, Dec 1990, p359–364.

[16] Porter, M.F. An algorithm for suffix stripping. *Program* 14 (3), Jul 1980, p130-137.

[17] Cooper, W.S. and Gey, F.C. Experiments in the Probabilistic Retrieval of Full Text Documents. In: [21], p127–134.

[18] Robertson, S.E. Unpublished note on phrase weighting. 1996.

[19] *The First Text REtrieval Conference (TREC–1).* Edited by D.K. Harman. Gaithersburg, MD: NIST, 1993.

[20] *The Second Text REtrieval Conference (TREC–2).* Edited by D.K. Harman. Gaithersburg, MD: NIST, 1994.

[21] *Overview of the Third Text REtrieval Conference (TREC–3).* Edited by D.K. Harman. Gaithersburg, MD: NIST, 1995.

[22] *The Fourth Text REtrieval Conference (TREC–4).* Edited by D.K. Harman. Gaithersburg, MD: NIST, 1996.

Cluster-Based Adaptive and Batch Filtering

David Eichmann, Miguel Ruiz, Padmini Srinivasan
School of Library and Information Science
University of Iowa

1 – Introduction

Information filtering is increasingly critical to knowledge workers drowning in a growing flood of byte streams [6, 8, 9]. Our interest in the filtering track for TREC-7 grew out of work originally designed for information retrieval on the Web, using both 'traditional' search engine [5] and agent-based techniques [6, 7]. In particular, the work by Cutter, et. al. in clustering [3, 4] has great appeal in the potential for synergistic interaction between user and retrieval system.

Our efforts for TREC-7 included two distinct filtering architectures, as well as a more traditional approach to the adhoc track (which used SMART 11.3). The filtering work was done using TRECcer – our Java-based clustering environment – alone for adaptive filtering and a combination of TRECcer and SMART for batch filtering.

2 – Adhoc Track

2.1 – Adhoc Methodology

Our overall approach is to apply Rocchio-based retrieval feedback [11] for query expansion. The second run submitted (Iowacuhk2) is simply such a run with the top 10 documents from an initial retrieval run assumed relevant and the best 350 terms extracted from these documents. Documents and queries were weighted using the Lnu.ltu scheme [12] which had yielded good results in previous TREC runs [e.g., 2]. For our primary run (Iowacuhk1), we focussed on improving the initial retrieved set that is assumed relevant during retrieval feedback. The following steps describe our approach.

1. Retrieve 10 documents using the initial query (Lnu.ltu) weights. Call this set A.
2. Identify the top 3 documents for each query.
3. Treat these top 3 documents as pseudo-queries, index them against the same database and retrieve 100 documents for each pseudo query (Lnu.ltu weights with pivot average document size of 126 and a slope of 0.2).
4. Merge the 100 documents from the three pseudo-queries and eliminate duplicates. Call this set B.
5. Find the intersection of sets A and B for each topic. Use this set for retrieval feedback to expand the query. Call this set C.
6. Expand the original query by 350 terms using alpha = 8, beta = 8 and gamma = 8 with set C and using Rocchio's algorithm.
7. Retrieve the final set of 1000 documents using the expanded query (Lnu.ltu weights using the above parameters).

2.2 – Results and Analysis

Figures 1 through 3 compare the performance of the two Iowa runs against the minimum, maximum and median scores for the adhoc track. Table 1 below summarizes this performance. It shows the number of topics in which the corresponding Iowa run performs at or better than the median value.

Table 1: Adhoc Performance

	>= Median, Top 100 Retrieved	>= Median, Top 1000 Retrieved	>= Median, Avg. Precision	Avg. Precision (non-interpolated)	Exact Precision
Iowacukh1	37	32	32	0.2221	0.2680
Iowacuhk2	39	34	35	0.2260	0.2754

Thus in 64 to 78% of the topics, the Iowa runs are at or above the median performance. It is also seen that our second run, i.e., the straight Lnu.ltu and Rocchio-based retrieval feedback approach is slightly better for each measure than our primary run in which we tried to refine the set of documents used for retrieval feedback. Although somewhat

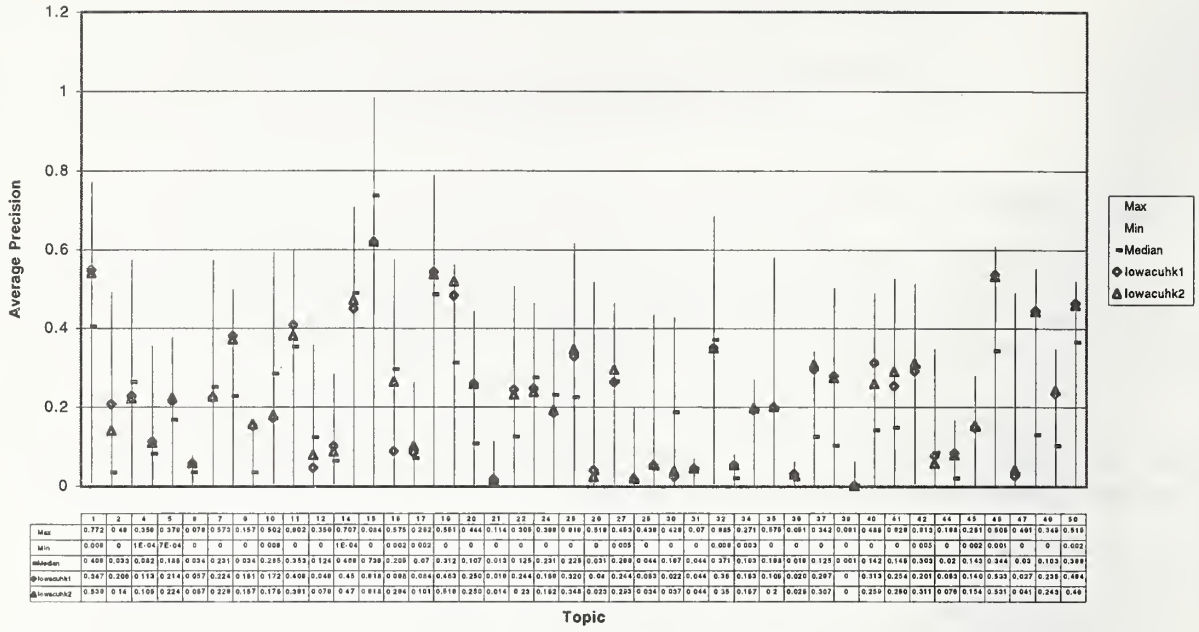


Figure 1: Adhoc Retrieval, Average Precision

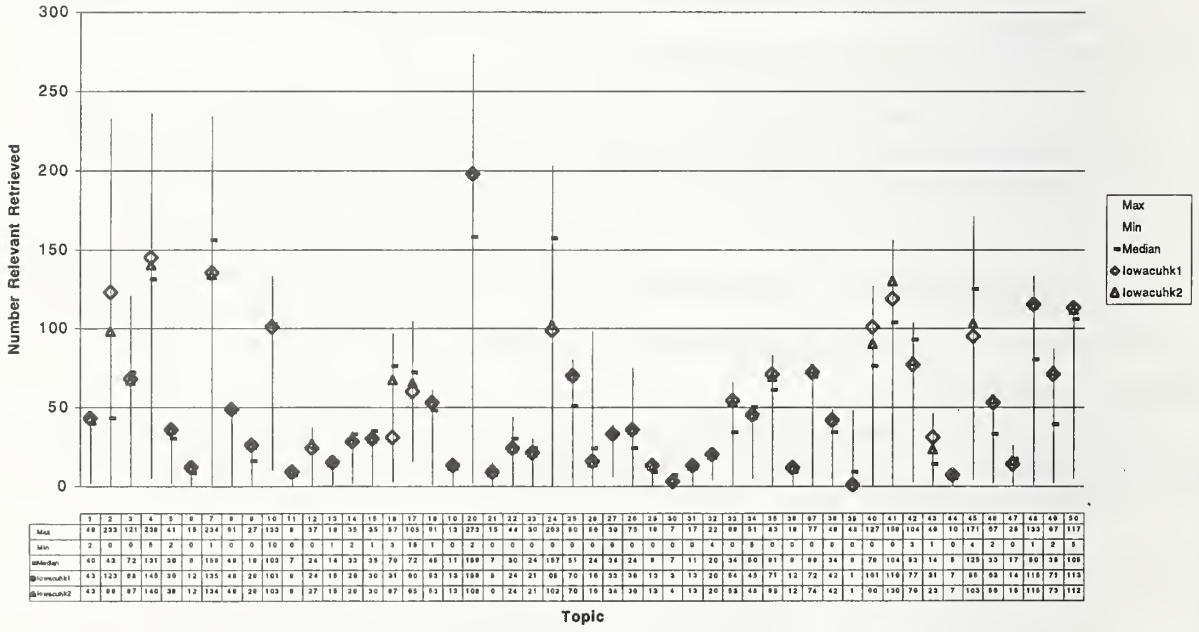


Figure 2: Adhoc Retrieval, # Relevant Retrieved in Top 1000 Documents

disappointing, this performance sets an internal baseline against which we hope to show improvements in our future TREC efforts.

3 – Adaptive Filtering Track

Our existing approach to Web search/filtering involves a dynamic clustering technique where the threshold for formation of new clusters and the threshold for visibility of 'sufficiently important' clusters can be specified by the

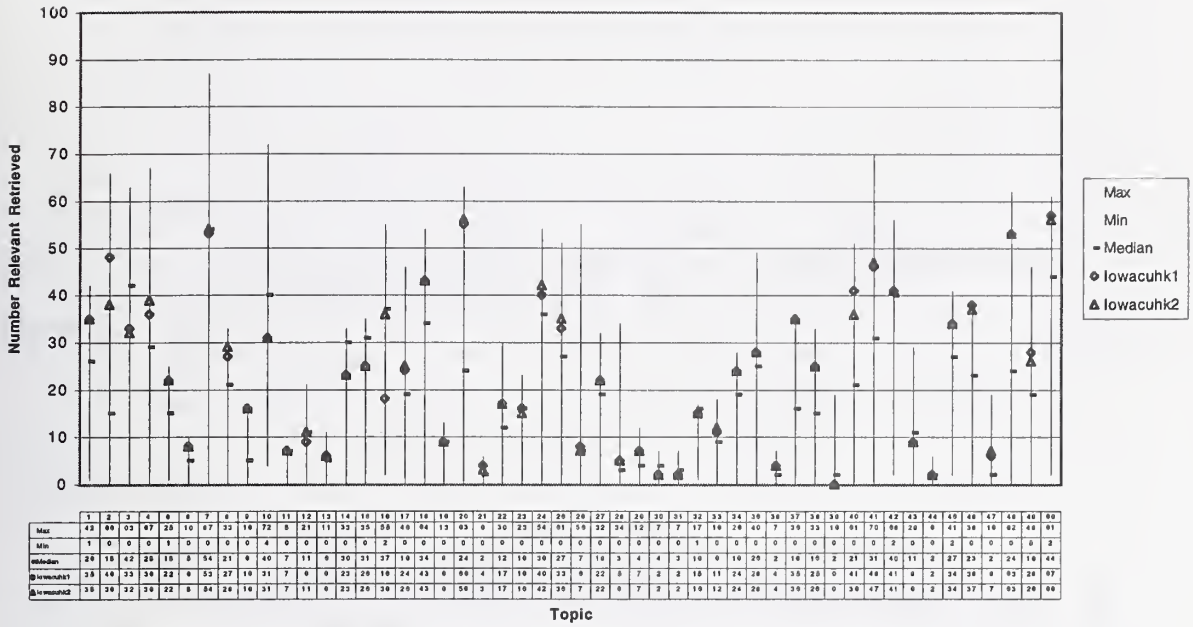


Figure 3: Adhoc Retrieval, # Relevant Retrieved in Top 100 Documents

user when the topic is presented to the system. As documents are retrieved and clusters form, the user can select interesting clusters for further exploration through the links contained in member documents [7]. The TREC requirements for multi-query support and simulation of user judgment responses led us to modify the single set-of-clusters model, creating a two-level scheme.

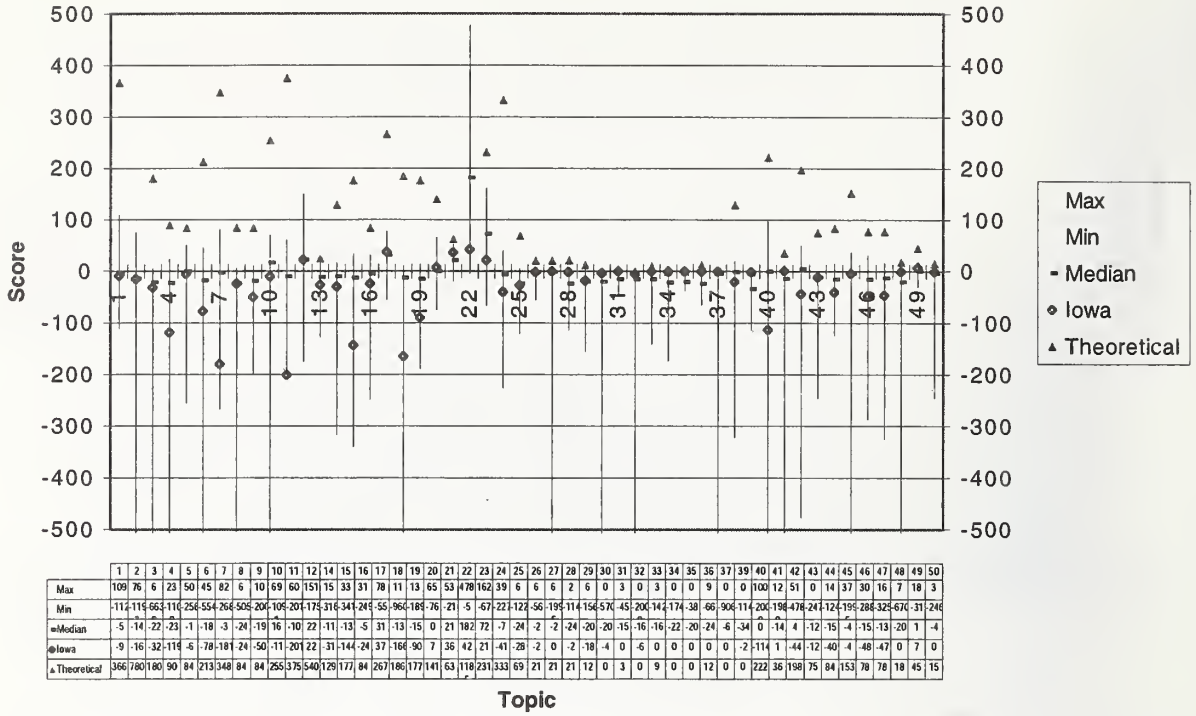
Similarity between documents and clusters is measured using a straight-forward vector cosine measure:

$$sim(d, c) = \frac{\sum_{i=1}^{N_d} \sum_{j=1}^{N_c} (TF(W_i) \cdot TF(W_j))}{\sqrt{\sum_{i=1}^{N_d} TF(W_i) \cdot \sum_{j=1}^{N_c} TF(W_i)}}$$

Term frequencies are built up incrementally as a given run progresses and cluster term weights are adjusted every ten input files. This approach is therefore somewhat inaccurate in the initial phases of a run, but quickly reaches a point of reasonable stability with respect to term frequencies and has the added benefit of requiring no fore-knowledge of the vocabulary. All vocabulary is stemmed using Porter's algorithm [10]. We prune document term vectors to the 100 most weighty terms and cluster vectors to the 200 most weighty terms. This proves to have no significant effect on the accuracy of our results, but a significant effect on both memory requirements and execution time, the latter due to a corresponding reduction in the cost of dot product calculations.

The primary cluster level corresponds to the internal representation of a topic definition. The original threshold specifications were retained here to allow specification of the first-order recall of the system. We experimented with a variety of means of generating a primary similarity measure, but settled on one based upon the text of the topic's concept definitions for the submitted runs.

The secondary level is where the adaptive portion of the system functions and where we found the most benefit in parameter tuning. Each primary cluster (and hence, each topic) has a private set of zero or more secondary clusters. When a document clears the threshold for a primary cluster, it either joins an existing secondary cluster or forms a new one, based upon a membership threshold. The shift from a single membership threshold to a primary/secondary pair



	1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	39	40	41	42	43	44	45	46	47	48	49	50
Max	109	76	6	23	50	45	82	6	10	69	60	151	15	33	31	78	11	13	65	53	478	162	39	6	6	6	2	6	0	3	0	3	0	0	9	0	0	100	12	51	0	14	37	30	16	7	18	3
Min	-112	-115	-66	-110	-256	-554	-266	-505	-200	-109	-201	-175	-316	-341	-249	-55	-966	-189	-76	-21	-5	-67	-22	-122	-56	-195	-114	-156	-570	-45	-200	-142	-174	-38	-66	-906	-114	-200	-198	-478	-241	-124	-195	-288	-325	-670	-31	-240
▪Median	-5	-14	-22	-23	-1	-18	-3	-24	-19	16	-10	22	-11	-13	-5	31	-13	-15	0	21	182	72	-7	-24	-2	-2	-24	-20	-20	-15	-16	-16	-22	-20	-24	-6	-34	0	-14	4	-12	-15	-4	-15	-13	-20	1	-4
◆Iowa	-9	-16	-32	-119	-6	-78	-181	-24	-50	-11	-201	22	-31	-144	-24	37	-166	-90	7	36	42	21	-41	-28	-2	0	-2	-18	-4	0	-6	0	0	0	0	0	-2	-114	1	-44	-12	-40	-4	-48	-47	0	7	0
▲Theoretical	366	780	180	90	84	213	348	84	84	255	375	540	129	177	84	267	186	177	141	63	118	231	333	69	21	21	21	12	0	3	0	9	0	0	12	0	0	222	36	198	75	84	153	78	78	18	45	15

Figure 4: Adaptive Filtering, AP'88 F1 Results

allowed us to achieve a tunable level of recall (by using a lower primary threshold, as mentioned above) while teasing out distinctions between candidate document clusters through use of a higher secondary threshold.

Introduction of a visibility threshold for secondary cluster similarity to the primary then gives us a means for adaptation. When a secondary cluster's similarity first exceeds the visibility threshold, its member documents are declared to the user and relevance judgments are obtained. The secondary is then colored appropriately. Secondary clusters containing relevant (and unjudged, if any) documents are colored green and have all subsequent members declared as relevant. Secondaries containing non-relevant (and potentially, unjudged) documents are colored red and declare no further members. Adaptation then occurs over time as secondary clusters exceed the visibility threshold and are colored, with red secondary clusters mitigating the lack of precision provided by the recall-centric primary threshold.

Secondary clusters exceeding the visibility threshold potentially contain a mix of different document types (relevant, non-relevant and unjudged). We currently address this in the following, conservative manner: if a secondary cluster contains

- one or more relevant documents, no non-relevant documents and zero or more unjudged documents, color it green;
- one or more non-relevant documents, no relevant documents and zero or more unjudged documents, color it red;
- one or more relevant documents, one or more non-relevant documents and zero or more unjudged documents, color it gray, but treat it as green;
- fewer than a specific number (currently 10) of unjudged documents and no relevant or non-relevant documents, leave it uncolored until the first relevant or non-relevant document is added, then color it appropriately (note that this optimistic stance has a distinct effect w.r.t. false positives); and finally,
- more than a specific number of unjudged documents and no relevant or non-relevant documents, color it red (we do this pessimistically due to the low density of judged documents in the corpus).

We selected a primary similarity threshold of 0.18, secondary similarity threshold of 0.5 and visibility threshold of 3 in our preliminary experiments with the Wall Street Journal corpus, but used a primary similarity threshold of 0.15, secondary similarity threshold of 0.4 and visibility threshold of 2 based upon an assumption that the WSJ corpus involved a more restricted vocabulary than the AP vocabulary. Figures 4, 5, and 6 show our results for AP88, AP89,

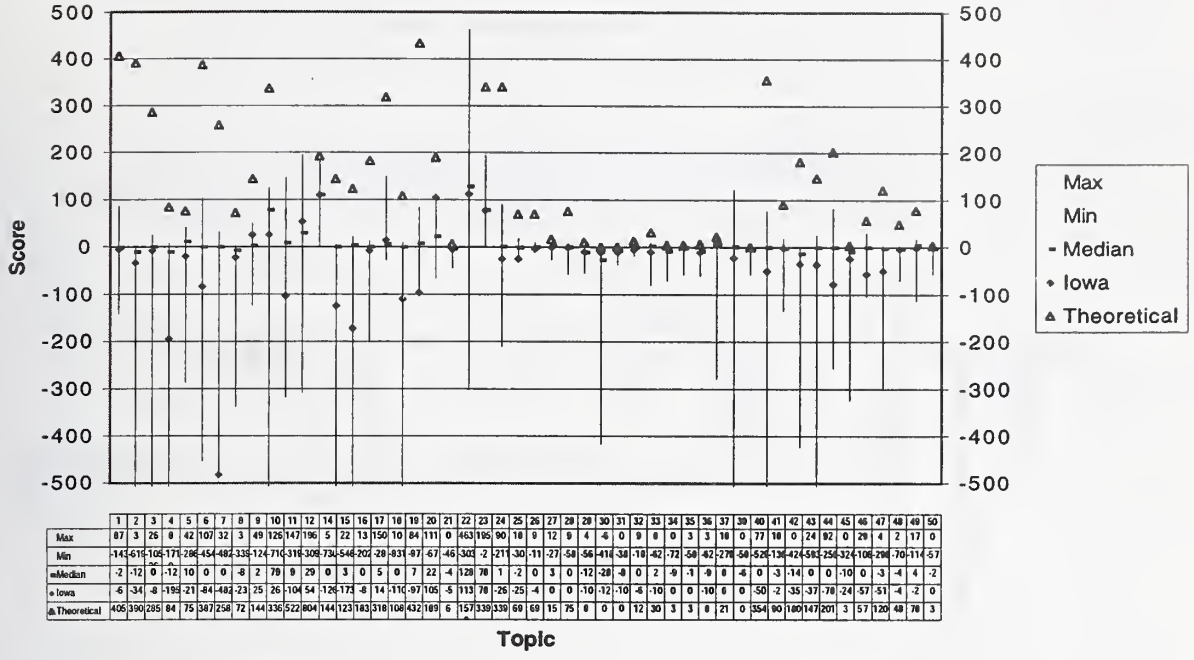


Figure 5: Adaptive Filtering, AP'89 F1 Results

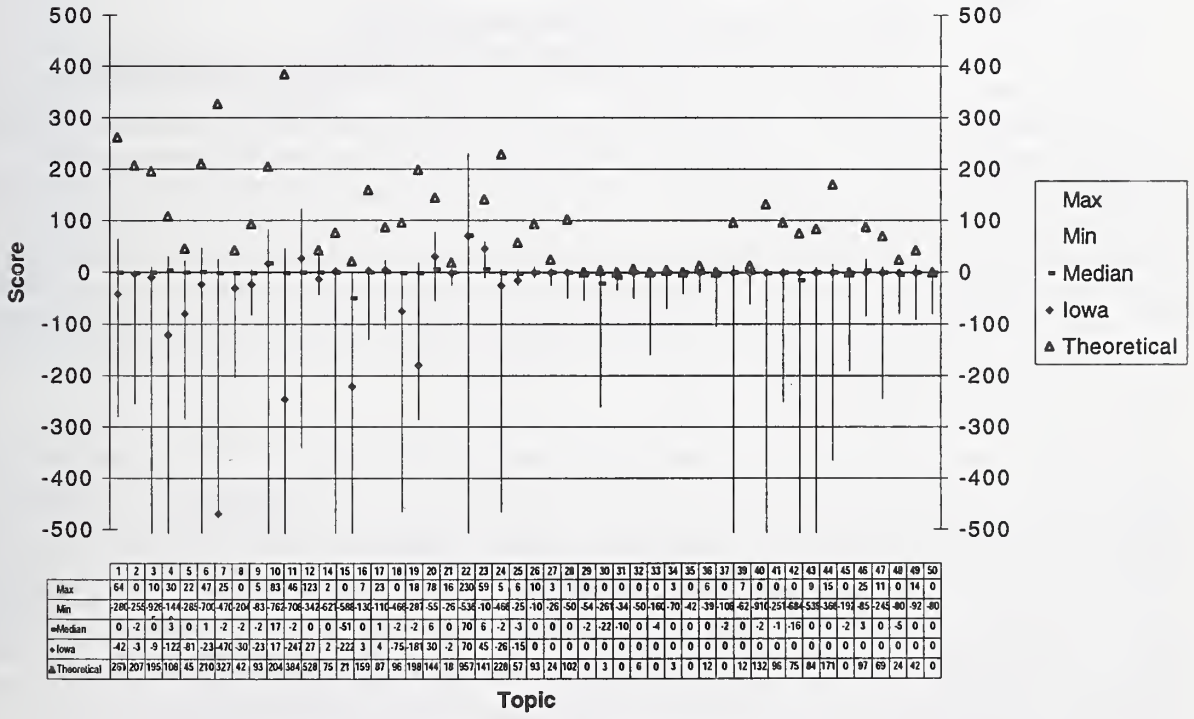


Figure 6: Adaptive Filtering, AP'90 F1 Results

and AP90, respectively for the F1 measure. The vertical bars indicate the min, max and median scores for a given topic, the circles our score, and the diamonds the theoretically possible score. Table 2 shows the number of topics for which we score at or above the median for each of the three years.

Table 2: Adaptive Filtering Result Comparison

Year	# of scores above median
AP88	23
AP89	15
AP90	31

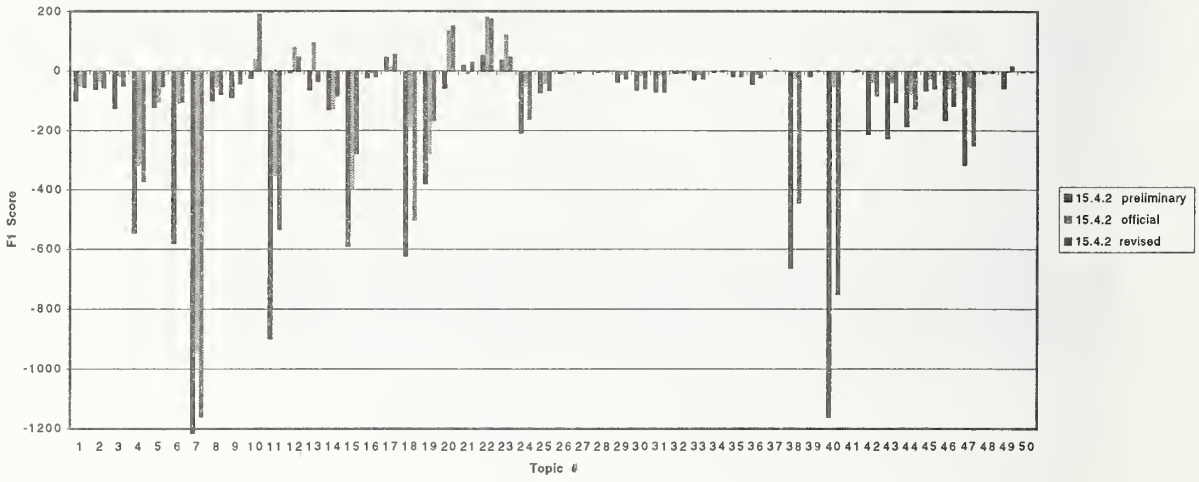


Figure 7: F1 Scoring Shifts Due to QRel Coverage

With a few notable exceptions (e.g., Topic 7), our approach scores very near the best performance recorded in the official runs. This includes the topics with extremely low density of relevant documents. Our performance in suppressing the negatively judged documents in topics 26-50 appears to train well in '88-'89, but at the cost of missing relevant documents in '90 in topics 40-50. This could well be an artifact of the trade-off between the number of secondary clusters formed and their resulting specificity. A smaller number of more general clusters suffers from lack of focus, but also declares fewer negatively judged documents in coloring a cluster. A larger number of more specific clusters improves cluster specificity, but can cause the declaration of a greater number of negatively judged documents as clusters corresponding to fine discriminations in concepts present in the corpus are colored. These effects will require substantially more experimentation.

The points in time at which feedback occurs has significant effect upon the performance of the clustering algorithm. Figure 7 shows the results for our preliminary run (with original qrels for '88 and '89 only), the official run (with original qrels for all years) and a revised run (with revised qrels for all years). In virtually all topics the cumulative F1 score is higher for both the official runs and the revised qrel run over the preliminary run with qrels for only '88 and '89. Note however, that there are numerous cases where the official run outperforms the revised qrel run. We suspect that this is due to shifts in cluster declaration patterns across the changing qrel patterns. As an example of this, consider the pattern of black cluster declaration for Topic 5 as shown in Figures 8 and 9. As the preliminary run exhausts its qrels at snapshot 69 (the end of '89), no additional black clusters are declared until the end of '90. The revised run, with additional judgments for '90, continues to declare a substantial number of negative clusters, but at the same time, scores better than the preliminary run. The growth in the number of negative clusters is due to the relatively high density of negative judgements compared to positive judgements in '90.

Figures 10 and 11 show a somewhat different situation for Topic 12. While the density of all judgements varies more substantially in '90, the number of both positive and negative clusters declared does not substantially differ from the preliminary run to the revised run. The cumulative F1 score, however, shows a distinct improvement, capitalizing on already declared clusters to suppress non-relevant documents and declare relevant documents.

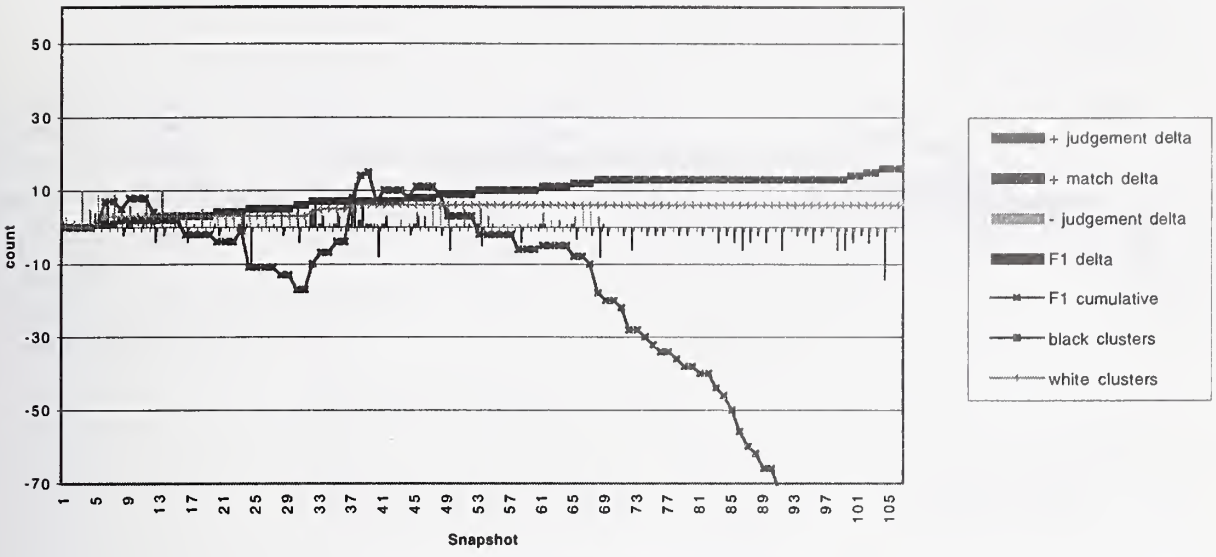


Figure 8: Topic 5, Preliminary QRels

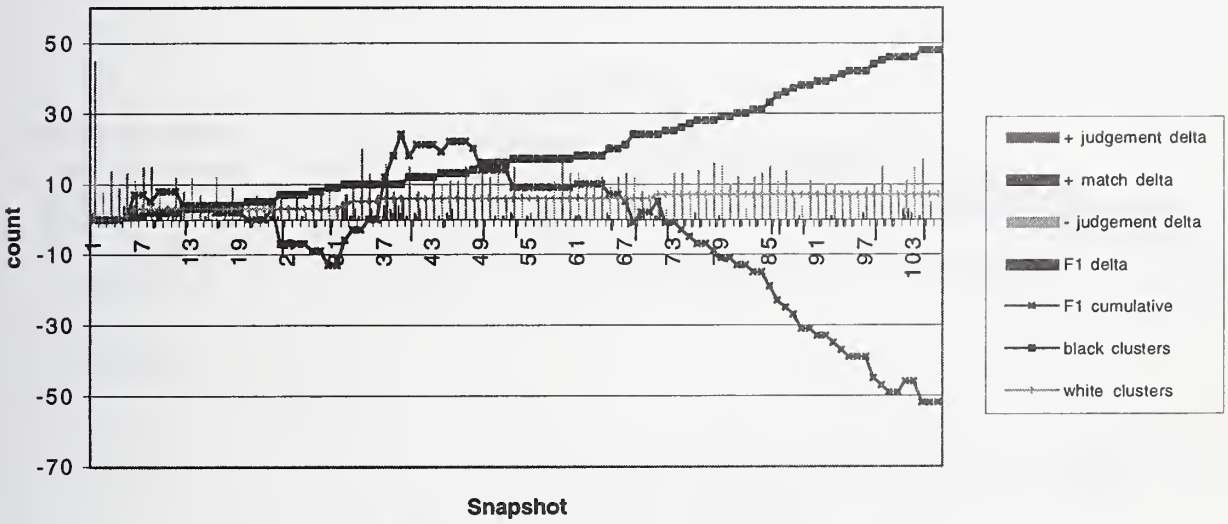


Figure 9: Topic 5, Revised QRels

4 – Batch Filtering Track

For this subtrack we decided to use Rocchio's query expansion method to build an initial profile for each topic, and then let the adaptive system learn on the training before processing the test set. The following steps describe our methodology:

1. Index the training set (AP88, topics, and relevance judgments of AP88) using SMART. We used the pivoted normalization weighting scheme (Lnu.ltu), stop words, and no stemming.
2. Expand the topics using relevance feedback on the training dataset. For this, the initial retrieval run extracted the top 100 documents. Rocchio's method was used with parameters alpha=8, beta=8, and gamma=8. The top 200 terms were used for expansion. These expanded topics were input to the TRECcer program in step 3.
3. Run the TRECcer program. Originally we intended to generate IDF statistics on all the AP88 train set for the TRECcer program. However, due to lack of time we selected a subset of files and used only those files for IDF statistics and initial cluster formation with the TRECcer. For each topic, the first file in the training collection that

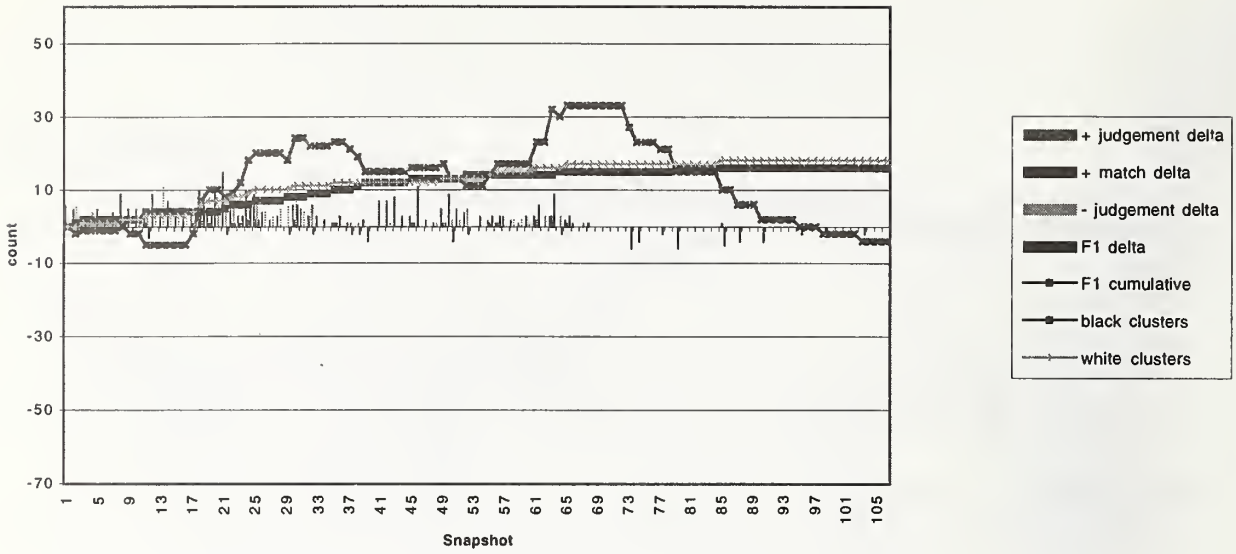


Figure 10: Topic 12, Preliminary QRels

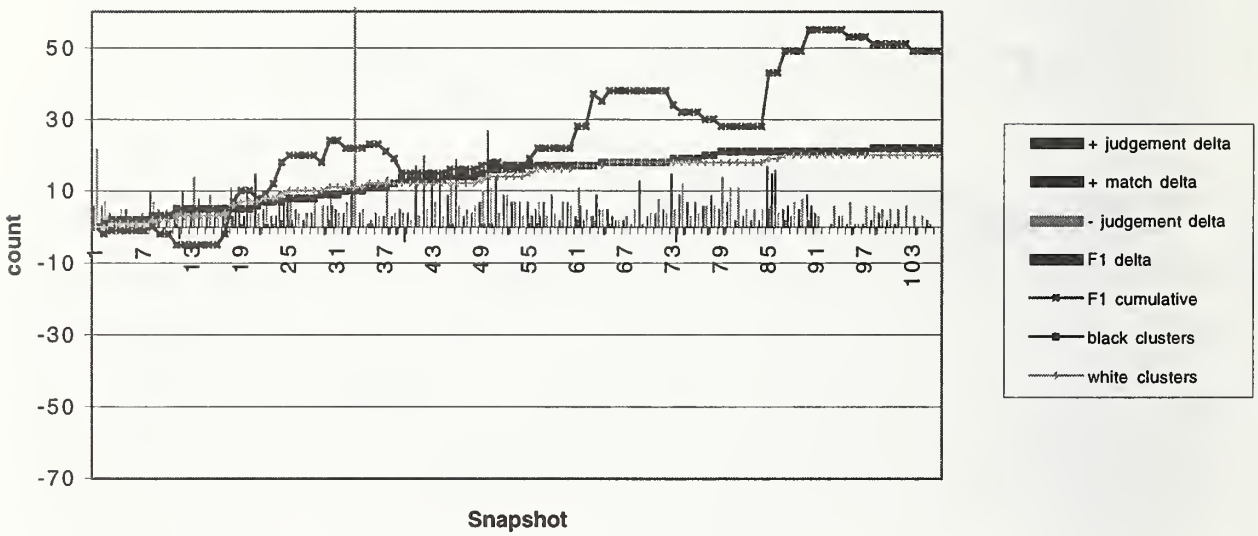


Figure 11: Topic 12, Revised QRels

contains a relevant document was identified. This procedure generated a subset of 26 files.

The parameters used in the batch filtering TRECcer runs were tuned using the training set AP88. We obtained the following settings:

- F1 run (shown in Figure 12): primary cluster membership= 0.25, secondary membership threshold= 0.5, visibility threshold = 0.25.
- F3 run (shown in Figure 13): primary cluster membership= 0.2, secondary membership threshold= 0.5, visibility threshold = 0.25.

We ran the TRECcer program on 20 machines (a mix of HP and SGI workstations), assigning 5 topics to each.

4.1 – Results

Unfortunately, the official results included partial results for many of the queries (only queries 36-50 for the F1 run had been completed). We did complete the runs later using the entire AP88 subset for training the system before starting to process the test set (AP89-90).

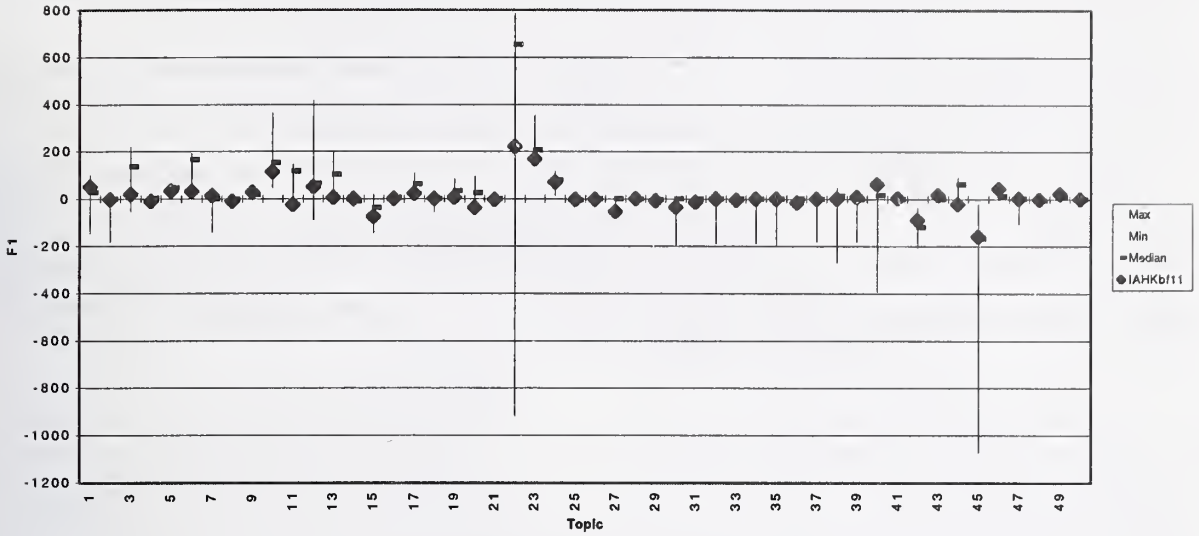


Figure 12: Batch Filtering, F1 Results

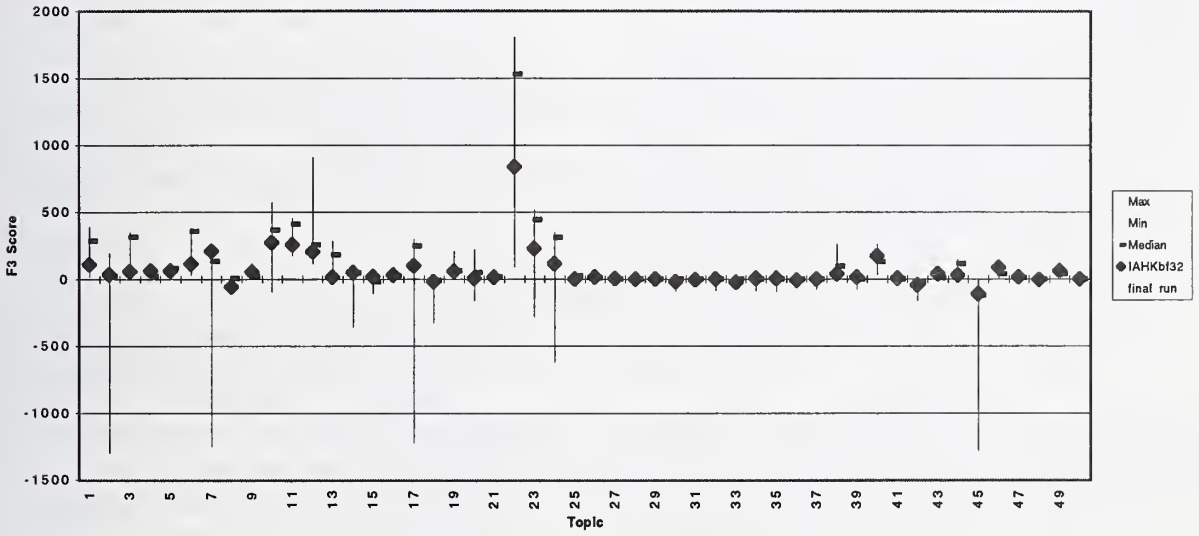


Figure 13: Batch Filtering, F3 Results

Table 3 summarizes the results of our F1 runs. In the official run 17 of the topics were above or equal to the median(12 of them the maximum), while in the unofficial but complete run this improved to 23 (13 of the maximum).

Table 3: Performance of the Batch Filtering Runs

	>= Median	Maximum in
F1 (official)	17	12
F1 (complete)	22	13
F3 (official)	12	7
F3 (complete)	23	7

In the official F3 runs 12 of the topics were above the median (7 of them the maximum), whereas in the unofficial run 23 topics were above the median (7 of the maximum).

The results obtained in our F3 runs indicate that the unofficial run significantly outperforms our official run. This improvement is justified in part by the fact that the unofficial runs include more declared documents that eventually will increase the chance of retrieving relevant documents.

In the case of the F1 runs the unofficial run is better but the improvement is smaller. We attribute this to the fact that our official F1 run already had 15 completed topics. There is also a difference in the training examples used. The Official run uses a subset of 26 files, while the unofficial run uses the entire training set. We compared the results obtained in the subset of queries 36-40. In the official run 10 of these topics are above the median (6 at the maximum), while in the unofficial run 12 of the topics are above the median (8 at the maximum). We also observe that there are differences in the number of declarations – 7 topics increase the number of declarations while 2 decrease. This is because a different training set induces a different secondary cluster structure.

5 – Conclusions and Future Plans

Our preliminary experience with two-level clustering and a mixed architecture of TRECcer and SMART have been encouraging. We expect that with further tuning of primary/secondary cluster interaction we will achieve significantly better results. Our performance on the Wall Street Journal corpus during our earlier experiments with clustering lead us to believe that similarity thresholds are sensitive to vocabulary diversity, particularly compared to the more diverse vocabulary of the AP corpus. We are quite interested in exploring a blend of lower primary thresholds and higher secondary thresholds. This should improve our recall, but only at the cost of early training of negative examples.

6 – References

[1] Baclace, P. E., "Competitive Agents for Information Filtering," *Communications of the ACM,* v. 35, n. 12, December 1992, p. 50.

[2] Buckley, C., M. Mitra, J. Walz, and C. Cardie, "Using Clustering and SuperConcepts within SMART," *Proc. of Sixth Text REtrieval Conference (TREC-6),* Gaithersburg, Maryland, November 19-21, 1997. http://trec.nist.gov/pubs/trec6/papers/index.track.html

[3] Cutting, D. R., D R. Karger and J. O. Pedersen, "Constant interaction-time scatter/gather browsing of very large document collections," *Proc. of SIGIR'93,* June 1993.

[4] Cutting, D., D. Karger, J. Pedersen, and J. W. Tukey, "Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections," *Proceedings of the 15th Annual International ACM/SIGIR Conference,* Copenhagen, 1992.

[5] Eichmann, D. "The RBSE Spider – Balancing Effective Search Against Web Load," *First International Conference on the World Wide Web,* Geneva, Switzerland, May 25-27, 1994, pages 113-120.

[6] Eichmann, D., "Search and Metasearch on a Diverse Web," *First W3C Workshop on Distributed Indexing and Search,* Boston, MA, May 28-29, 1996.

[7] Eichmann, D., "Ontology-Based Information Fusion," *Workshop on Real-Time Intelligent User Interfaces for Decision Support and Information Visualization, 1998 International Conference on Intelligent User Interfaces,* San Francisco, CA, January 6-9, 1998.

[8] Lieberman, H., "Letizia: An Agent That Assists Web Browsing," *Proceedings of the International Joint Conference on Artificial Intelligence,* Montreal, August 1995.

[9] Lieberman, H., "Autonomous Interface Agents," *Proceedings of the ACM Conference on Computers and Human Interfaces (CHI'97),* Atlanta, GA, March 1997.

[10] Porter, M. F., "An Algorithm for Suffix Stripping," *Program,* v. 14, no. 3, 1980, p. 130-137.

[11] Rocchio, J., "Relevance feedback in information retrieval," in *The SMART Retrieval System: Experiments in Automatic Document Processing,* G. Salton (ed.), Prentice-Hall, p. 313-323.

[12] Singhal, A, G. Salton, M. Mitra and C. Buckley, "Document Length Normalization," *Information Processing & Management,* v. 32, no. 5, Sept. 1996, p. 619-633.

Rutgers' TREC-7 Interactive Track Experience

N.J. Belkin, J. Perez Carballo, C. Cool*, D. Kelly, S. Lin,
S.Y. Park, S.Y. Rieh, P. Savage-Knepshield, and C. Sikora

School of Communication, Information & Library Studies
Rutgers University
4 Huntington Street
New Brunswick, NJ 08901-1071
*GSLIS, Queens College, CUNY

Abstract

We present results of a study comparing two different interactive information retrieval systems: one which supports positive relevance feedback as a term-suggestion device; the other which supports both positive and negative relevance feedback in this same context. The purpose of the study was to investigate the effectiveness and usability of a specific implementation of negative relevance feedback in interactive information retrieval. A second purpose was to investigate the effectiveness and usability of relevance feedback implemented as a term-suggestion device. The results suggest that, although there was no benefit in terms of performance for the system with negative and positive relevance feedback, this might be due to specific implementation issues.

1.0 Introduction

As in TREC-7, we continued the work begun in our TREC-6 experiments (Belkin, et al., 1998), investigating the effectiveness and usability of negative relevance feedback (RF) in interactive information retrieval (IR). One reason for considering negative RF was that subjects in our previous TREC experiments had expressed the desire to be able to make negative judgments on retrieved documents which would subsequently affect retrieval and ranking. Another was our belief that a particular way of implementing negative RF would lead to identifying documents in which "good" query terms appear in inappropriate contexts (Belkin, et al., 1998).

Our TREC-6 results seemed rather inconclusive, primarily because of the small number of subjects taking part in the experiment, the small number of searches that they conducted, and because of what seemed to be problems with our interface design. The TREC-7 experimental protocol gave us the opportunity to compare directly, rather than indirectly, our two conditions (positive RF only, versus positive plus negative RF), thereby also increasing the number of subjects in each condition, as well as the number of searches by each subject. In addition, we redesigned our interface to take account of problems that were made evident in TREC-6.

Following the results of Koenemann (1996), we implemented RF as a term-suggestion device for user-controlled query expansion. In TREC-7, we attempted to investigate the effectiveness and usability of this implementation of RF, or at least of its effect on our main questions concerning negative and positive RF.

Below, we first discuss the systems that we used to investigate our research questions, and the methods that we applied. We follow with an overview of the results, divided into three sections: characteristics of the subjects; *effectiveness* of the positive RF system (RUINQ-G) versus the positive plus negative RF system (RUINQ-R), and of term suggestion; and, *usability* of the two systems and of term suggestion. We then discuss some relationships amongst these results, and some interpretations of them, and conclude with some suggestions about where to go next.

2.0 Methods

This section describes our subjects, experimental IR systems, and the procedures that we followed while conducting our TREC-7 Interactive Track experiment.

2.1 Searchers

Sixteen volunteer searchers were recruited to participate in this study from the population of students in the School of Communication, Information and Library Studies at Rutgers University. None of our subjects had taken part in previous TREC studies and none had prior experience with our RU-INQUERY system. Demographic characteristics of the searchers and their experiences with IR systems are described in Section 3.1.

2.2 Experimental IR Systems

We used Inquery 3.1p1 including its default values for indexing, retrieval, and RF. The major difference between our implementation and the standard version

of Inquery, apart from the interface, was in RF implementation. We modified Inquery's RF function so that it produced a list of 50 terms, for both positively and negatively judged documents. As users made RF judgments about documents, the top *n* terms were presented in a term suggestion window. At the user's discretion, these terms could be added to the existing query. The term ranking algorithm was default Inquery.

Positive and negative RF were both implemented using the standard Inquery method for positive relevance judgments. However, terms that co-occurred in highly-ranked negatively judged documents and in either the original query or in positively judged documents were excluded from the suggested "bad" terms list. The number of terms suggested was determined by the formula: $n = 5i + 5$ in which i is the number of judged documents, and n is no greater than 25.

Appendix A contains a screen dump of the positive and negative RF system (RUINQ-R). The positive only RF system (RUINQ-G) was identical, except that there were no Bad Terms to Avoid window, no Clear Bad Docs button, and no Bad RF radio buttons. Our interface offered the following features and functionality:

- Query terms window - used to input a free-form query, with minimal structure (phrases and negative terms).
- Results Summary window - displayed the titles of ten documents and provided radio buttons for marking documents as good, bad (in the positive and negative RF condition), and saved.
- Document window - displayed text of a selected document.
- Pop-up Instance Labeling window - used to label saved documents according to the "instances" that they represented.
- Documents Saved window - listed the saved document's title and its associated instance label.
- Good Terms to Add window -- displayed suggested terms which could be added to the query by clicking on them.
- Bad Terms to Avoid window -- displayed suggested terms which could be added to the query by clicking on them (this window was only presented in the positive and negative RF condition).
- Search Button - used to retrieve a list of documents.
- Clear Query button - used to remove all terms in the query terms window.
- Clear Good Documents and Clear Bad Documents Buttons -- used to "unmark" previously marked good and bad documents, respectively.
- Show Next Keyword, Show Best Passage, Show Next, and Show Prev buttons - used to quickly navigate through the full text of a document.
- Exit button - used to end a search session.

Both systems ran on a SUN Ultra 140 with 64MG memory and 9GB disk under Solaris 2.5.1 with a 17" color monitor.

2.3 Procedure

Each searcher conducted eight searches in accordance with the TREC-7 Interactive Track experimental guidelines. Searchers were alternately assigned to one of two experimental conditions. In one condition, searchers conducted four searches using RUINQ-R with positive and negative relevance feedback (RF) and then conducted their next four searches using RUINQ-G with positive only relevance feedback. In the second condition, system order was reversed so that searchers used the system with positive only RF followed by the system with positive and negative RF. Within each condition, topic block presentation was counterbalanced so that half of the subjects searched on topic block B1 (365i, 357i, 362i, 352i) first, while the other half searched on block B2 (366i, 392i, 387i, 353i) first.

On arrival, the subjects read and signed a consent form explaining their rights and the potential risks associated with participation in the experiment. Next, they completed a demographic questionnaire that gathered background information and probed their previous searching experience. After completing the Controlled Associations Test FA-1, they were ready to begin their first tutorial. The tutorial familiarized them with the features and usage of the RU-INQUERY system that they would be using for the first half of the experiment. They received written instructions for the task in general and specific instructions for the current search topic. They were allotted 15 minutes to complete each search. As they searched, they specified "instances" of the topic as they identified them and "thought aloud." A videotape recorded the computer monitor during their searches and captured their "thinking aloud" utterances and the entire search interaction was logged. After completing each search, a brief questionnaire was completed in which participants assessed their topic familiarity, search difficulty, search result satisfaction, aspect identification confidence, and satisfaction with the amount of time allotted for the search. This process was completed for three more searches and then participants were given the opportunity to take a break. After the break, the same process was repeated

for the second system that they used to conduct the remaining four searches. After completing all eight searches, the participants completed an exit questionnaire and an exit interview. We added the exit interview and some additional survey questions to the standard Interactive Track data collection instruments, to better understand users' perceptions of the usefulness of both positive and negative RF, of the usefulness of term suggestion, and of the usability of the two systems' interfaces. All subjects were tested individually and required approximately 3-1/4 hours to complete the study.

3.0 Results

3.1 Characteristics of the Subjects

The subject group included 11 females and 5 males, whose ages ranged from 23 to 58. Thirteen of the subjects either had, or were pursuing a graduate degree in library science. Of these 13 subjects, 1 was a Ph.D. student in library science and one had already earned a Ph.D. in a subject area outside of library science. Three of the subjects either had, or were candidates for Master's level degrees in areas outside of library science. One subject had only a high school diploma. The occupations of the subjects did not vary greatly. Eight subjects reported being students, 4 reported being librarians, 2 reported being members of academic faculty and 2 were in neither of these categories. None of the subjects reported having previously participated in any TREC experiments.

The median number of years reported for overall experience doing online searching was 3.00 (M=3.59, SD=3.76). The minimum amount of experience reported was 1 and the maximum was 17. The average amount of previous search experience on different types of systems did not vary by much. This previous search experience was accessed using a five-point scale where 0=no experience and 5=great deal of experience. Subjects reported the most experience using a point and click interface (M=4.75, SD=0.68) and the least experience using CD-ROMs (M=3.06, SD=1.48). The average rating and standard deviations for subjects' search experience with library catalogs, commercial online systems, and the World Wide Web were, respectively: M=4.18, SD=0.83; M=3.46, SD=0.99; M=4.37, SD=0.80. Subjects reported conducting searches an average of 4.43 (SD=0.62) on a five-point scale where 1=never; 2=once or twice a year; 3=once or twice a month; 4=once or twice a week; and 5=once or twice a day. Subjects also reported the extent to which they enjoy carrying out information searches. A different five-point scale was used where 1=strongly disagree; 3=neutral; and 5=strongly agree. The average response was 4.31 (SD=0.70).

3.2 Effectiveness

3.2.1 Effectiveness of Positive versus Positive Plus Negative Relevance Feedback

The precision and instance recall for all subjects in both systems were 0.64 and 0.37 respectively. Overall, the subjects saved 6.16 documents in average within 892.22 seconds (14.74 minutes). Table 1 presents these results, comparing mean performance in RUINQ-G (positive RF only) and RUINQ-R (positive plus negative RF). The differences in peformance between the two systems are insignificant on all four measures; for the basic performance measure of *instance recall* the relevant figure is t(125) = 0.925.

In order to check for interaction effects, we compared performance by system order and by topic block order. The results showed that the subjects who used the RUINQ-R system first performed a little bit better in terms of recall (M= .38, SD= .23) than those who used RUINQ-G system first (M= .35, SD=24). However, t-test results revealed that the difference was not statistically significant: t(125)= -.635. The means of instance recall between two different block order are almost the same, M= .37 (SD= .25) in Block1 and M= .36 (SD= .22) in Block2.

A *cycle* in our analysis is defined as the number of invocations of the "Search" button plus one. Transaction logs saved during the searches revealed that the subjects engaged in 6.8 cycles per search. Overall, they identified 5.02 instances per search. For

	RUINQ-G Mean (SD)	RUINQ-R Mean (SD)	Total Mean (SD)
Time (seconds)	892.55 (206.47)	891.89 (179.86)	892.22 (192.75)
Documents saved	6.46 (3.78)	5.87 (3.86)	6.16 (3.81)
Precision	.64 (.27)	.63 (.30)	.64 (.29)
Instance Recall	.39 (.24)	.35 (.23)	.37 (.23)

Table 1. Comparison of Performance between RUINQ-G (positive RF) and RUINQ-R (positive & negative RF).

	RUINQ-G Mean (SD)	RUINQ-R Mean (SD)	Total Mean (SD)
Cycles	7.19 (5.30)	6.41 (3.77)	6.79 (4.59)
Instances entered	5.49 (3.74)	4.56 (3.45)	5.02 (3.61)
Terms in the first query	3.52 (3.41)	3.69 (2.98)	3.60 (3.19)
Terms entered by user in the last query	4.02 (4.22)	4.62 (4.16)	4.32 (4.18)
Full documents displayed	27.24 (16.30)	24.80 (12.15)	26.01 (14.35)
Titles shown	304.02 (339.69)	203.33 (250.65)	253.28 (301.24)

Table 2. Comparison of Searching Behavior between RUINQ-G (positive RF) and RUINQ-R (positive & negative RF)

each search, 253.28 unique titles were shown to the subjects, and 26.01 full documents were displayed. Table 2 summarizes the results of transaction logs according to the type of system used.

We analyzed the performance data with respect to the searching behaviors identified in Table 2. The only significant relationship was that greater user terms in the last query resulted in lower performance in terms of instance recall ($r = -.212$, $p<0.05$). The rest of searching behaviors were not correlated with performance results.

We compared the searching behaviors between high-performance subjects and low-performance subjects. A "high performance" subject is one whose mean instance recall is above the mean for all subjects; a "low performance subject is one whose mean instance recall is below that of the mean for all subjects. By this categorization, eight subjects were high-performance, and eight were low-performance.

Low-performance subjects entered more terms in their first query (M=4.21, SD=3.92), and entered more terms in the last query (M=5.25, SD=5.15) than high-performance subjects (M=2.98, SD=2.07 in the first query and M=3.38, SD=2.61 in the last query). These differences were statistically significant with $t(125)=2.21$, $p< .05$ in the number of terms entered in the first query, and $t(125)=2.57$, $p< .05$ in the number of terms entered by users (not chosen from term suggestion features) in the last query respectively. On the other hand, high-performance subjects saved more instances, (M=5.81, SD=4.35), and displayed more full documents (M=29.53, SD=15.40) than low-performance subjects (4.25 (SD=2.50) instances, 22.53 (SD=12.40) full documents). These differences were significant according to the t-test results at the level of 0.05: number of instances, $t(125)= -2.48$; number of full documents displayed, $t(125)= -2.82$. In addition, high-performance subjects saw more document titles (M=321.27, SD=369.14) than low-performance subjects (M=186.34, SD=195.50). This result was also significant, with $t(125) =-2.58$, $p<.01$.

3.2.2 Effectiveness of Term Suggestion

In RUINQ-G, which has positive RF only, the subjects chose 2.31 (SD=3.34) terms per search on average from the positive terms suggested by system. In approximately half (31 searches, 49.2%) of the total 63 searches using RUINQ-G, the subjects didn't choose any term from the positive terms suggested.

In RUINQ-R, which has positive and negative RF, the subjects chose 1.76 terms on average from the positive terms suggested and 1.97 terms from the negative terms suggested. In 30 searches (46.9%) out of 64 searches, no positive terms were chosen. In 39 searches (60.9%) out of 64 searches, the subjects didn't choose any negative terms for their queries.

Neither number of positive terms selected nor number of negative terms selected was significantly correlated with instance recall. In the RUINQ-G system, high-performance subjects chose 2.06 terms, and low-performance subjects chose 2.56 terms. On the RUINQ-R system, the high-performance subjects selected 1.72 positive terms and 2.03 negative terms while low-performance subjects selected 1.81 positive terms and 1.91 negative terms. In the number of terms chosen, either negative terms or positive terms, there was no significant difference between high-performance subjects and low-performance subjects.

The effectiveness of the term suggestion feature was also investigated by analyzing data from subjects' self reports during the exit interview. The subjects were asked, "To what extent did you find the term suggestion feature useful during your searches? Why is that?" and "To what extent did the term suggestion feature improve your ability to identify different aspects of the topics? Why is that?" The usefulness of the term suggestion feature yielded an overall mean rating of 3.19, where 1 represented not at all useful

and 5 indicated completely useful. The extent to which the term suggestion feature was indicated to improve their ability to identify different aspects of the topics had a mean rating of 3.25, where 1 represented no improvement of ability and 5 indicated complete improvement of ability. Overall, the term suggestion feature was not viewed as highly useful.

The subjects who had the positive only system first seemed to be less positive about the term suggestion feature's usefulness than those subjects having the system with both positive and negative term suggestions first. The positive only first subjects complained about the type of words provided and the cognitive overhead in analyzing the terms. Although the positive only second group was more positive about the usefulness in general, it was mostly based on unintended advantages. These included not having to type the word in and providing a summary of the document. Most felt they could think of better words than those provided.

The comments from the positive only first group regarding the feature's contribution to improving their ability to identify aspects were generally high or low with few in between. The subjects either did not like the feature at all or thought it was really useful. The ones who discussed it as useful generally mentioned the bad term suggestion advantages or the summary overview the good terms provide. The positive only second group were generally more positive, but several just did not want to use it for the intended purpose. As with the other group, it helped some get more items, but others used it as summary information.

The comparison of subjects based on performance indicated few differences in effectiveness of the term suggestion feature. The effectiveness of the system in terms of being useful during the searches had a mean ratings of 3.13 and 3.25 for the high performers and low performers respectively (on a 5-point scale where 5 is highest). The effectiveness of the system was also not different for the two groups on the question of the feature improving their ability to identify aspects. Both high performers and low performers had a mean ratings of 3.25 (on a 5-point scale where 5 is highest).

3.3 Usability

3.3.1 Usability of the systems in general

Subjects were asked to consider their search experience following four searches with each system. Using a five-point scale where 1=not at all; 3=somewhat; and 5=extremely, subjects were asked to answer 3 questions: how easy it was to learn to use the system; how easy it was to use the system; and how well they understood how to use the system. The results yielded no significant difference between subjects' mean ratings on these questions for RUINQ-G and for RUINQ-R. These ratings are displayed in Table 3.

These same questions were also looked at according to system order, block order and performance status. Again, there was no significant difference for any of these conditions. However, for system order, it does appear that subjects in system order RUINQ-G/RUINQ-R consistently prefer system RUINQ-G to system RUINQ-R while subjects in system order

	RUINQ-G Mean (SD)	RUINQ-R Mean (SD)
Easy to learn to use	4.00 (0.65)	3.81 (0.98)
Easy to use	3.86 (0.74)	3.56 (1.09)
Understand how to use	3.66 (0.72)	3.37 (1.08)

Table 3. Usability of RUINQ-G (positive RF) versus RUINQ-R (positive & negative RF)

	System Order			
	RUINQ-G/ RUINQ-R		RUINQ-R/ RUINQ-G	
	RUINQ-G Mean (SD)	RUINQ-R Mean (SD)	RUINQ-G Mean (SD)	RUINQ-R Mean (SD)
Easy to learn to use	4.0 (.53)	3.37 (1.18)	4.0 (.81)	4.25 (.46)
Easy to use	4.0 (.53)	3.25 (1.28)	3.7 (.95)	3.87 (.83)
Understand how to use	4.0 (.53)	3.12 (1.13)	3.2 (.76)	3.62 (1.06)

Table 4. Usability of RUINQ-G (positive RF) versus RUINQ-R (positive & negative RF) according to System Order

Preferred System:	RUINQ-G N (%)	RUINQ-R N (%)	Total[a] N
Easier to learn to use	12 (80%)	3 (20%)	15
Easier to use	10 (71%)	4 (29%)	14[b]
Liked best	7 (50%)	7 (50%)	14[b]

Table 5. Number (percentage) of subjects preferring one system over the other
Note: [a] one missing data point in each case; [b] one subject rated systems equal

RUINQ-R/RUINQ-G consistently prefer system RUINQ-R. Unfortunately, the low number of subjects did not permit us to do further analysis on this data. These results are displayed in Table 4.

fter subjects had completed all searching, they were asked to compare RUINQ-G and RUINQ-R in an exit questionnaire. Questions included: how easy they were to learn to use; how easy they were to use; and which system the subjects liked best. Subjects were asked to place a "1" next to the easier/liked best system and a "2" next to the more difficult/liked least system. The results are displayed in Table 5. There is a significant difference between which system was easier to learn to use. RUINQ-G was rated as significantly easier to learn to use χ^2 (1, N=15)=5.4, $p < .05$. Although RUINQ-G also appears to be the easier system to use, there was no significant difference in subject ratings. For system preference, or which system subjects *liked best*, RUINQ-G and RUINQ-R each received an equal number of preferences. Although not part of our instructions, one subject rated the two systems as equally likable and equally easy to use.

Subject ratings from the exit questionnaire were then analyzed in terms of system order, block order and high/low status. The results of these analyses are displayed in Tables 6, 7 and 8, respectively. The results for system order indicate that RUINQ-G was easier to learn to use and easier to use. Subjects' rankings of which system they liked best with respect to system order revealed that the majority of subjects in system order RUINQ-G/RUINQ-R ranked RUINQ-R as the most likable system and the majority of subjects in system order RUINQ-R/RUINQ-G ranked RUINQ-G as the most likable system. These results indicate that subjects may have a preference for whichever system that they used last.

When subjects' rankings of which system was easier to learn to use were examined in regard to block order, the results indicate that RUINQ-G was easier to learn to use regardless of block order. For ease of use rankings and block order, the results show that the majority of subjects in both block orders ranked RUINQ-G as the easier system to use. The likeness rankings for RUINQ-G and RUINQ-R revealed that while the majority of the subjects in Block Order 1 ranked RUINQ-G as being the most likable system, the majority of subjects in Block Order 2 ranked RUINQ-R as being the most likable system. The equal and missing data in this category prevented us from concluding that this finding was significant.

When the subjects' rankings of which system was easier to learn to use were examined in regard to performance status, the results indicate that RUINQ-G was easier to learn to use regardless of performance status. The ease of use ratings were similar across both high and low performers: 5 of the high performers ranked RUINQ-G as being the easier system to use and 5 of low performers ranked RUINQ-G as the easier system to use. However, the majority of the high performers liked RUINQ-R best, while the majority of the low performers liked RUING-G best.

System Order:	RUINQ-G/RUINQ-R			RUINQ-R/RUINQ-G		
Preferred System:	RUINQ-G N=	RUINQ-R N=	Total	RUINQ-G N=	RUINQ-R N=	Total[a]
Easier to learn to use	6 (75%)	2 (25%)	8	6 (86%)	1 (14%)	7
Easier to use	5 (63%)	3 (37%)	8	5 (83%)	1 (17%)	6[b]
Liked best	3 (37%)	5 (63%)	8	4 (67%)	2 (33%)	6[b]

Table 6. Number (percentage) of system preferences according to System Order
Note: [a] one missing data point in each case; [b] one subject rated systems equal

Block Order:	Block Order 1			Block Order 2		
Preferred System:	RUINQ-G N=	RUINQ-R N=	Total	RUINQ-G N=	RUINQ-R N=	Total[a]
Easier to learn to use	7 (88%)	1 (12%)	8	5 (71%)	2 (29%)	7
Easier to use	6 (75%)	2 (25%)	8	4 (67%)	2 (33%)	6[b]
Liked best	6 (75%)	2 (25%)	8	1 (17%)	5 (83%)	6[b]

Table 7. Number (percentage) of system preferences according to Block Order
Note: [a] one missing data point in each case; [b] one subject rated systems equal

Performance Status:	High Performers			Low Performers		
Preferred System:	RUINQ-G N=	RUINQ-R N=	Total	RUINQ-G N=	RUINQ-R N=	Total[a]
Easier to learn to use	7 (88%)	1 (12%)	8	5 (71%)	2 (29%)	7
Easier to use	5 (71%)	2 (29%)	7[b]	5 (71%)	2 (29%)	7
Liked best	2 (29%)	5 (71%)	7[b]	5 (71%)	2 (29%)	7

Table 8. Number (percentage) of system preference according to Performance Status
Note: [a] one missing data point in each case; [b] one subject rated systems equal

3.3.2 Use and Usability of Term Suggestion

Subjects were asked to rate their understanding of the term suggestion feature and their use of the term suggestion feature to modify their searches using a five-point scale where 1=not at all; 3=somewhat; and 5=extremely. The results from these questions are displayed in Table 9. Overall, subjects rated their understanding of term suggestion with mean = 3.78. Subjects in system order RUINQ-G/RUINQ-R rated their understanding of term suggestion significantly higher (M=4.25) than those subjects in system order RUINQ-R/RUINQ-G (M=3.17), with $t(12)=2.16$, $p<0.05$. Block order and performance status had no significant effect with respect to understanding of term suggestion.

Overall, subjects rated their use of the term suggestion feature to modify their searches with a mean of 3.32. Subjects in system order RUINQ-G/RUINQ-R did not respond much differently than those in system order RUINQ-R/RUINQ-G (G: M=3.5; R: M=3.1). Subjects in Block Order 1 rated this question significantly lower (M=2.81) than those subjects in Block Order 2 (M=4.0), with $t(12)=.019$, $p<0.05$. The ratings on this question for high and low performers were very similar (M=3.42; M=3.21).

The usability of the term suggestion feature was specifically addressed in the exit interview. The subjects were asked, "To what extent did you understand how to use the term suggestion feature? Why is that?" and "To what extent did you use the term suggestion feature to modify your searches? Why is that?" Overall, the mean rating for understanding the feature was 3.84 where 1 represented no understanding and 5 indicated complete understanding. The overall mean rating for using the feature was 3.25, where 1 represented no use and 5 indicated complete use. Generally, subjects described their understanding of the feature in terms of a synonym suggestion tool. The use of the feature varied in the way it was used. The negative term suggestion feature was generally used to constrain the documents retrieved, however the positive term suggestion feature was often used to get an overview of the document or to find synonyms.

The subjects who used the feature with positive only term suggestion before using it with both positive and negative, had a higher mean rating for their understanding of the system than those who used the systems in the reverse order, 4.25 and 3.44 respectively. The subjects in the positive only first system order who gave the lowest ratings indicated they had a tendency to mark documents as good or bad randomly looking for certain words or ideas, which were not found. The higher ratings within this subject group were given by subjects who discussed the term suggestion feature as synonyms or words to refine the query. The subjects using the positive only system first had a higher mean rating for the amount of feature use than those using that system second, 3.5 and 2.9 respectively. The lower ratings from this group came from subjects who expressed a distrust of the bad term suggestion feature, because they suggested it was unclear how it worked. Higher ratings came from subjects who indicated that they used negative terms to eliminate documents already seen.

	Overall	System Order		Block Order		Performance Status	
		GR	G	Block 1	Block 2	High	Low
Understanding of term suggestion feature	3.78	4.25	3.17	3.5	4.17	3.42	4.14
Use of term suggestion feature to modify searches	3.32	3.5	3.1	2.81	4.0	3.42	3.21

Table 9. Usability/Use of Term Suggestion Feature

The subject group using the positive only system after the system with both positive and negative term suggestions, mostly complained about the use of the feature rather than addressing how well they understood it. When the use of the feature was discussed it was as providing alternate terms and further direction for honing the search. The group indicated a moderate use of the feature. They discussed the bad term suggestion feature as being used in the way intended. However, the good term suggestion was not viewed as particularly helpful, but the highest rater used it to scan the content of the documents.

The usability of the system in terms of being able to understand how to use the terms suggestion feature had a mean rating of 3.38 for the high performers and 4.13 for the low performers (on a 5-point scale where 5 is highest). The better performers purported to understand the feature less well. However, there was no difference in usability ratings in terms of how much they indicated using the term suggestion feature. High performers had a mean rating of 3.25 and low performers had a mean rating of 3.25 (on a 5-point scale where 5 is highest). Interestingly, high performers generally had more comments than low performers.

3.4 Subject Characteristics and Effectiveness and Usability

None of the demographic characteristics that were observed, including performance on the FA-1 Controlled Associations Test, was significantly related to any of the explicit performance or usability measures discussed above. But we should note that this was a relatively homogeneous population.

4.0 Discussion

Our initial hypothesis that a system providing both positive and negative RF would perform better than one that offered positive RF only was not supported by our experiment. In our study, the two systems performed no differently on instance recall. We might conclude, therefore, that there appears to be no benefit of negative RF, on the task presented to our subjects.

However, the self-report and interview data provide us with a broader picture of subjects' uses and understandings of the systems and the features offered, which helps to explain the performance results we observed. These data suggest that subjects had difficulty conceptualizing how to use the *term – suggestion* feature, in particular. Because of their unfamiliarity with negative RF, many subjects distrusted the "bad term" suggestion feature and hesitated to use it. Furthermore, when subjects directly compared the two systems, they rated the system with positive only RF as being easier to learn, and easier to use. But at the same time, the subjects also rated both systems evenly in terms of overall preference, with respect to the task.

A significant finding of our study is the inverse relationship between number of query terms and performance. At the moment we have no explanation of this result, nor do we have any idea about causal direction. Clearly, further investigation of the actual interactive behavior of the searchers is needed in order to understand this result.

5.0 Conclusions

Once again, it appears that we have demonstrated that incorporation of negative RF in a system which offers RF as a term-suggestion device, does no harm, with respect to performance on the "instances" task. On the face of it, this is not a terribly exciting result. However, this result, taken in combination with the usability results, does suggest that making negative RF more learnable and more usable than in our current system could lead to a performance advantage over positive RF only systems. This further suggests that much more research on appropriate conceptual models of RF, and on interfaces to support interaction with RF needs to be done before the more general issue can be resolved.

RF as term suggestion appeared also not to be well

understood by our subjects, nor effectively used, in this implementation. Although some subjects understood it, and used it, as a query enhancement (i.e. new term finding) device, the fact that many understood it as a synonym device is unfortunate. This result could be explained by the nature of our subject pool, and also (more whimsically) by the speculation that the FA-1 test might have conditioned them to think in terms of synonyms, it is clear that a better conceptual model of RF as term suggestion needs to be developed.

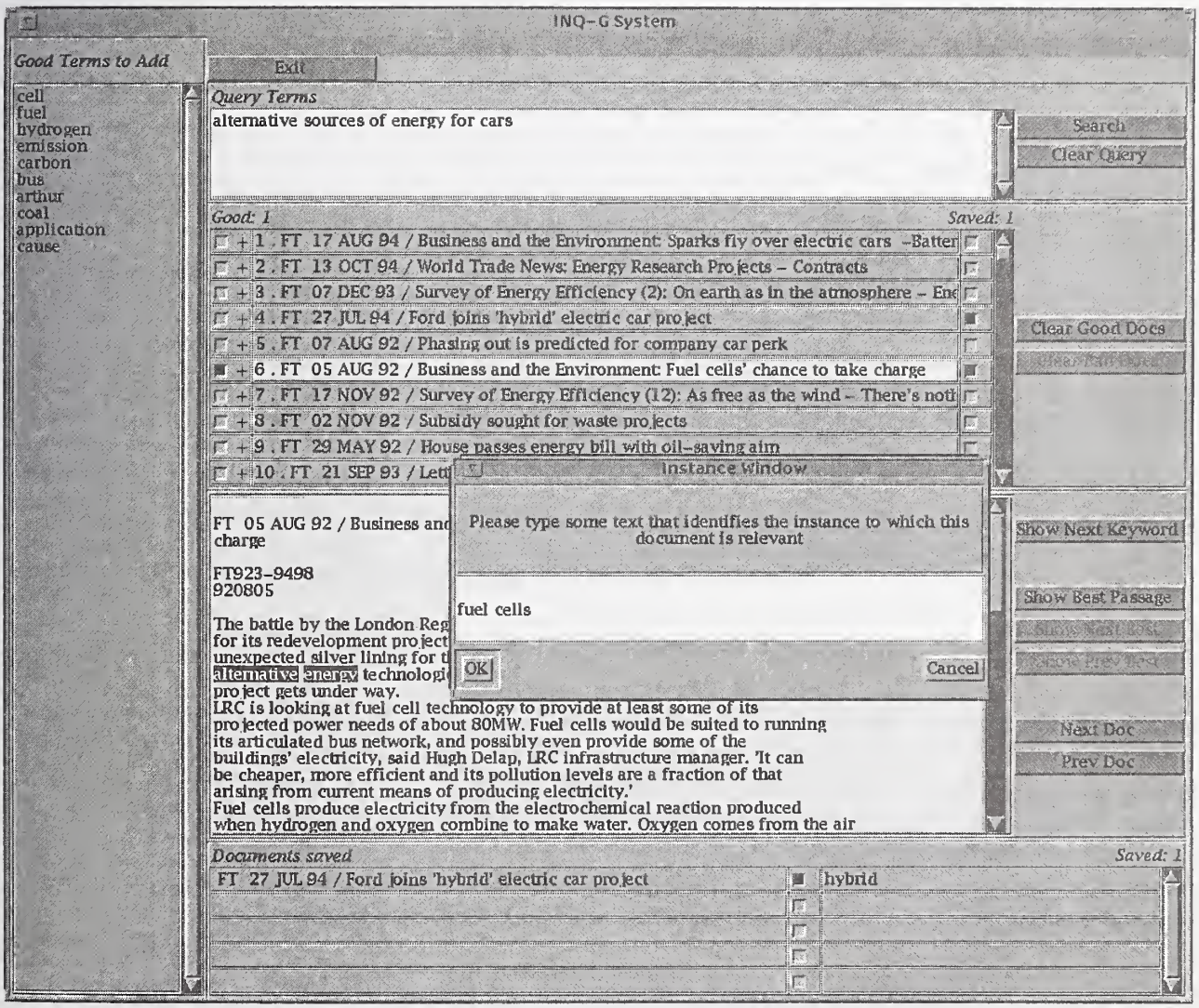
Finally, the counter-intuitive result that fewer query terms led to better performance needs to be investigated in much more detail. The result could be an artifact of the task itself, which would in itself be of considerable interest. But to understand this result will require detailed analysis of the interactions themselves, from a variety of points of view. An intriguing possibility that this result suggests is that interactive IR may work in much different ways than automatic, and that we may need to reconceptualize our understandings of what constitute "good" ways to do IR. We hope that our further analyses of these data, and related experiments, will shed light on this question.

6.0 References

Belkin, N.J. et al. (1998) Rutgers' TREC-6 interactive track experience. In D. Harman and E. Voorhees, eds. TREC-6. Proceedings of the sixth Text Retrieval Conference. Washington, D.C.: GPO.

Koenemann, J. (1996) *Relevance feedback: Usage, usability, utility.* Unpublished PhD dissertation, Department of Psychology, Rutgers University.

APPENDIX A. Screen Dump of RUINQ-R



SMART High Precision : TREC 7

Chris Buckley*, Mandar Mitra†, Janet Walz*, Claire Cardie†

Abstract

The Smart information retrieval project emphasizes completely automatic approaches to the understanding and retrieval of large quantities of text. We continue our work in TREC 7, concentrating on high precision retrieval. In particular, we present an in-depth analysis of our High-Precision Track results, including offering evaluation approaches and measures for time dependent evaluation. We participated in the Query Track, making initial efforts at analyzing query variability, one of the major obstacles for improving retrieval effectiveness.

Basic Indexing and Retrieval

In the Smart system, the vector-processing model of retrieval is used to transform both the available information requests as well as the stored documents into vectors of the form:

$$D_i = (w_{i1}, w_{i2}, \ldots, w_{it})$$

where D_i represents a document (or query) text and w_{ik} is the weight of term T_k in document D_i. A weight of zero is used for terms that are absent from a particular document, and positive weights characterize terms actually assigned. The assumption is that t terms in all are available for the representation of the information.

The basic "tf*idf" weighting schemes used within SMART have been discussed many times. For TREC 7 we use the same basic weights and document length normalization as were developed at Cornell by Amit Singhal for TREC 4([3, 5]. Tests on various collections show that this indexing is reasonably collection independent and thus should be valid across a wide range of new collections. No human expertise in the subject matter is required for either the initial collection creation, or the actual query formulation.

The same phrase strategy (and phrases) used in all previous TRECs (for example [2, 3, 4, 1]) are used for TREC 7. Any pair of adjacent non-stopwords is regarded as a potential phrase. The final list of phrases is composed of those pairs of words occurring in 25 or more documents of the initial TREC 1 document set. Phrases are weighted with the same scheme as single terms.

When the text of document D_i is represented by a vector of the form $(d_{i1}, d_{i2}, \ldots, d_{it})$ and query Q_j by the vector $(q_{j1}, q_{j2}, \ldots, q_{jt})$, a similarity (S) computation between the two items can conveniently be obtained as the inner product between corresponding weighted term vectors as follows:

$$S(D_i, Q_j) = \sum_{k=1}^{t} (d_{ik} * q_{jk}) \quad (1)$$

Thus, the similarity between two texts (whether query or document) depends on the weights of coinciding terms in the two vectors.

The Cornell TREC experiments use the SMART Information Retrieval System, Version 13.2, and most were run on a dedicated Intel dual 200 Mhz Pentium Pro running Solaris, with 512 Megabytes of memory and 49 Gigabytes of local disk (some runs were made on a Sun UltraSparc 1/140 with 512 Megabytes of memory).

SMART Version 13 is the latest in a long line of experimental information retrieval systems, dating back over 30 years, developed under the guidance of G. Salton. The new version is approximately 44,000 lines of C code and documentation.

*SabIR Research

†Department of Computer Science, Cornell University, Ithaca, NY 14853-7501

SMART is highly flexible and very fast, thus providing an ideal platform for information retrieval experimentation. Documents for TREC 7 are indexed at a rate of about 2 Gigabytes an hour, on hardware costing under $10,000 new. Retrieval speed is similarly fast, with basic simple searches taking much less than a second a query.

High-Precision Track

Track Overview

TREC 7 is the second year the High-Precision (HP) track been run. It is an attempt to perform a task that is much more closely related to real-world user interactions than the ad-hoc or routing task. The goal is simple: a user is asked to find 15 relevant documents in 5 minutes. No other restrictions are put on the user (other than no prior knowledge of the query, and no asking other users for help). Official evaluation is simply how many actual relevant documents were found among the 15 documents supplied by the user, modified slightly for those queries with fewer than 15 relevant documents in the collection (Relative Precision at 15 documents).

There are no restrictions on the type of resources the user may use during this task other than

- Only one user per query per run (no human collaboration).
- The user and system can have no previous information about the query (eg, the system cannot have previously built a query dependent data structure.)

In particular, the users are allowed to make multiple retrieval runs, allowed to look at documents, allowed to use whatever visualization tools the system has, and allowed to use system or collection-dependent thesauruses, as long as they stay within the 5 minute clock time.

This track tests (at least) the effectiveness, efficiency, and user interface of the systems. The task provides a forum for testing many of the neat ideas in user interface and visualization that have been suggested over the years.

Unlike other interactive evaluations (for example, the TREC 6 Interactive task), no attempt is made to factor out user differences when comparing across systems. All users are assumed to be experts and equally proficient in use of their own system. This allows for fair comparison of systems, but implies that the absolute level of performance within the track will be better than the level obtainable from casual users. These are upper-bound interactive experiments.

The only changes in the rules from the TREC 6 track are to raise the number of relevant documents required to 15 instead of 10, and to forbid cutting and pasting of the original query. This latter change requires the participants to type in the query, and makes the task fairer for those groups for whom cutting and pasting would not give a query in the proper form. It also has the side effect of making the task more difficult since reading and typing the query might take 30 seconds (10% of the available time).

High-Precision Methodology

Our methodology for the TREC 7 HP task is very similar to those we've used in the past 3 TRECs. [3, 4, 1]. The user's main task is to provide relevance judgements to be fed to our standard Rocchio relevance feedback algorithm. Direct modification of the query (adding/deleting terms to/from the query or directly modifying weights) was also occasionally (rarely) used by the searchers. The other principal component of our technique is the use of pipelining or "parallel" processing so that expensive retrieval techniques can be executing while the user continues to make judgements. The details of the method are given below:

1. The current time is noted. The user views the topic supplied by NIST and types a query into the system.
2. The query entered by the user is indexed and a set of documents is retrieved using a simple vector match.
3. The top-ranked documents are presented to the user.
4. The user starts viewing the documents and judging them 'relevant', 'non-relevant' or 'possibly relevant'.

In parallel, a child process is forked to retrieve additional documents using a more sophisticated retrieval algorithm: the initial query is used to retrieve 1000 documents, the top 20 are assumed to be relevant, documents ranked 501-1000 are assumed to be non-relevant, and automatic feedback is used to expand the query by 25 single terms and 5 phrases, using $\alpha = 8, \beta = 8$ and $\gamma = 8$.

5. After every judgement, the current time is noted. All documents retrieved so far are sorted such that the documents judged relevant come first, followed by all documents judged possibly relevant, followed by all unjudged documents, and the top 15 documents in this ranking are saved in a file and time-stamped.
6. After every 5 categorical judgements (i.e. 'relevant' or 'non-relevant'), a relevance feedback process is started in parallel if the child process is idle. For this process, documents marked relevant by the searcher are assumed to be relevant, and documents marked non-relevant as well as those retrieved at ranks 501-1000 by the initial user query are assumed to be non-relevant. Documents marked "possibly relevant" are not used in the feedback process. The query is expanded by 25 words and 5 phrases. $\alpha = 8, \beta = 8$ and $\gamma = 8$ are used. While this feedback process is running in the background, the user continues to judge more documents.
7. When the child process is done (i.e. retrieval or feedback completes), and the new retrieval results are available, these results are merged into the current list of top-ranked documents being shown to the user.
8. The final top 15 documents for the query will be the last set of 15 documents saved with a timestamp under the 5-minute limit

User Interface. The user interface for the TREC 7 high-precision runs is a simple GUI using Tk/Tcl. The display has 4 main windows

1. Text of user query
2. Vector form of user query
3. Current titles being judged
4. Current document, with query terms optionally highlighted

The GUI is used to view documents and mark documents 'relevant', 'non-relevant', or 'possibly relevant'. As soon as one document is judged, the next document is displayed. The user can go back and re-judge previously judged documents if needed, though in practice this was done mostly to correct errors of clicking the wrong judgement button. The interface may also be used to modify the user query statement by either modifying the text, or by modifying the term weights. After modification, the new query (or query vector) is used as the user query and combined with the existing relevance judgements in a relevance feedback retrieval. As an aid to pacing the query session, the interface displays the time elapsed since the beginning of the search.

Users and Settings. Three runs are presented; each the result of one user running all 50 queries. The users and some environmental characteristics are:

1. User 1 - Run HP1 : SMART System designer (HP interface designer) using Pentium Pro 200 dual processor with Solaris
2. User 2 - Run HP2 : SMART System implementer using UltraSparc 1/140
3. User 3 - Run HP3 : SMART System designer using UltraSparc 1/140

All three users should be considered experts and were running on comparable machines, though the Pentium was slightly faster. Unlike last year, all 3 users used highlighting of terms.

Effectiveness Results.

The effectiveness evaluation results are presented in Table 1. The base case is the official run Cor7A3rff which gives the precision at 15 documents of that automatic run. All three runs do very well and are amazingly

Run	Precision	Relative Precision	Num queries Best	Num queries ≥ Median
Base	.4760	-	-	-
Cor7HP1	.5787	.5909	12	38
Cor7HP2	.5813	.5920	19	37
Cor7HP3	.5853	.5967	16	43

Table 1: High-Precision comparison (50 queries)

close to each other. Less than 1% separates the top run from the bottom run. The top run is greater than or equal to the median on 86% of the queries, though the second run is best on more queries.

Agreements with TREC Assessors.

One important question is how the users agree with the official TREC relevance judgements. If the HP track is to have meaning, the disagreement between user interpretation of relevance to a query, and the official assessor interpretation can not dominate the results. Table 2 gives the total number judged relevant, possibly relevant, and non-relevant for each user, for both the TREC-assessor judged relevant documents and the TREC-assessor judged non-relevant documents. For example, User 3 judged 290 documents relevant (159) or iffy (131) that the official assessors had judged non-relevant.

Run	TREC judged Rel			TREC judged NonRel			Overlap
	UserRel	Iffy	NonRel	UserRel	Iffy	NonRel	(Iffy=rel)
Cor7HP1	315	170	51	79	181	448	61%
Cor7HP2	396	73	36	115	128	444	63%
Cor7HP3	374	100	84	159	131	674	56%

Table 2: High-Precision User-assessor consistency (50 queries)

The last column gives the overlap on judgements of relevant documents. If "Iffy" documents are assumed to be relevant, then the overlap for User 1 is 61% (from (315+170) / (315+170+51+79+181)). This is noticeably less than in previous studies, though it is not clear how much of this is due to the task. Often users marked documents as "Iffy" just because they were the only documents seen that were close to being relevant. Note that if we define "Iffy" documents as non-relevant when calculating overlap, the values are even lower: User 3 would have an overlap of only 52%.

The great majority of the disagreements are the users considering documents relevant that the assessor considered non-relevant. In fact, consider the 15 queries with lowest overlap for each of the three users; for all 45 queries the user has looser criteria than the assessor. This is to be expected, since the assessor as the originator of the query can easily have in mind a stricter query than made it to the topic description. For example, in query 375 "hydrogen energy", the assessor obviously did not want hydrogen fuel for car engines, though that wasn't clear from the topic. The three users marked a total 50 documents as relevant or iffy that were not relevant. Query 363 "tunnel disasters" was another with major disagreements (36 documents).

The disagreements in the other direction are rarer and a bit less obvious. For example, query 377, "cigar smoking", had the most disagreements, with 15 total assessor relevant documents being marked non-relevant by the three users.

The overall level of disagreement between assessor and users is unfortunately high. The overall level of performance is being strongly affected by agreement with assessor, rather than intrinsic performance.

Difficulty of Task.

One of the ways of telling how easy or difficult the TREC 7 HP task is, is to look at the queries for which the users did not find 15 documents that they thought were relevant. Table 3 gives the number of documents that are included in the final submitted retrieval without being judged. There will be unjudged documents only if the user did not find 15 relevant or iffy documents after 5 minutes.

According to the logs, it is obvious the users simply ran out of time on several queries. For example, for query 397, User 3 had just focused in on a set of relevant documents. User 3 had found 8 relevant or iffy

documents in 5 minutes, so 7 documents were filled in. 6 out of those 7 were relevant. Similarly, for Query 377, User 2 had only found 6 relevant or iffy documents by the end of 5 minutes, but 4 out of the next 9 documents were relevant. For these few queries, it is clear the 5 minute limit was effective and stressed the system. These queries account for the comparatively high number of relevant documents among the unjudged (ranging from 10% to 16%).

Run	num docs unjudged	num queries with unjudged	num unjudged rel docs
Cor7HP1	122	24	12
Cor7HP2	139	25	20
Cor7HP3	84	17	13

Table 3: Unjudged Retrieved Documents

However, half or less of the 50 queries have any unjudged documents at all for all three users. This includes queries for which there were fewer than 15 relevant documents in the collection. This implies for the majority of the queries, the only evaluation differences are due to disagreement with assessors rather than effectiveness of system. Combined with the high disagreement between users and assessors, the conclusion must be reached that the task is too easy.

Query Analysis.

Table 4 gives some facts and timings for query construction and retrieval runs. User 2 constructed shorter initial queries and used 10 seconds less time doing so. After initial queries were constructed, the initial simple run took less than 1 second to run (timings for these runs were measured in seconds so we do not have exact figures for the initial run). While the user was perusing the initial returned documents, a complex run was taking between 11 and 16 seconds. Then there were an average of 5 feedback runs made per query, each one taking from 7 to 12 seconds.

Run	num query terms	Construct query time	Complex run time	Num runs Feedback	Feedback run time	Num Judged
Cor7HP1	5.34	49.2	11.1	5.06	7.6	24.9
Cor7HP2	3.44	39.7	14.1	4.7	12.2	23.8
Cor7HP3	6.82	50.5	16.2	5.3	12.4	30.4

Table 4: Query Timing and Stats

Unlike our TREC 6 experiments, the complex and feedback runs took a reasonably short time to complete. The user typically only had time to judge one or two documents during these runs before the new documents would become available. It would have been possible to have had many more feedback runs; perhaps next time we will do so.

As can be inferred from Table 4 and Table 2, User 3 took an approach of judging as many documents as possible, as fast as possible. If the document wasn't obviously relevant on the first page, it was generally judged non-relevant, with the idea that there would be other more obviously relevant documents later. This allowed User 3 to judge an extra 5 to 6 documents per query as compared to the other two users. However, User 3 also had the lowest overlap with assessors, undoubtedly due to hasty judgements. User 3 looked at more relevant documents, but the inaccuracies in judgement meant the overall results remained the same as the other two users.

Timing Evaluation.

As has been indicated above, we kept track of not only what each user document judgement was, but when it occurred. Thus we can analyze the time performance of each user, and hopefully develop time-based evaluation measures that reflect the power and efficiency of systems.

The most obvious fact to look at is when the relevant documents were retrieved. Figure 1 gives the number of relevant documents retrieved during each 5 second timeslice for User 1, on average for 50 queries. The number of retrieved relevant starts off at 0 for the first 20 to 50 seconds as the user reads and types in the query. Then

it steadily increases for the next minute or so and then starts slowly decreasing up until the 5 minute point is reached. There's a big hump at 300 seconds as the 15 documents to be returned get filled in with unjudged documents. In the normal course, these documents would be judged over the next few buckets.

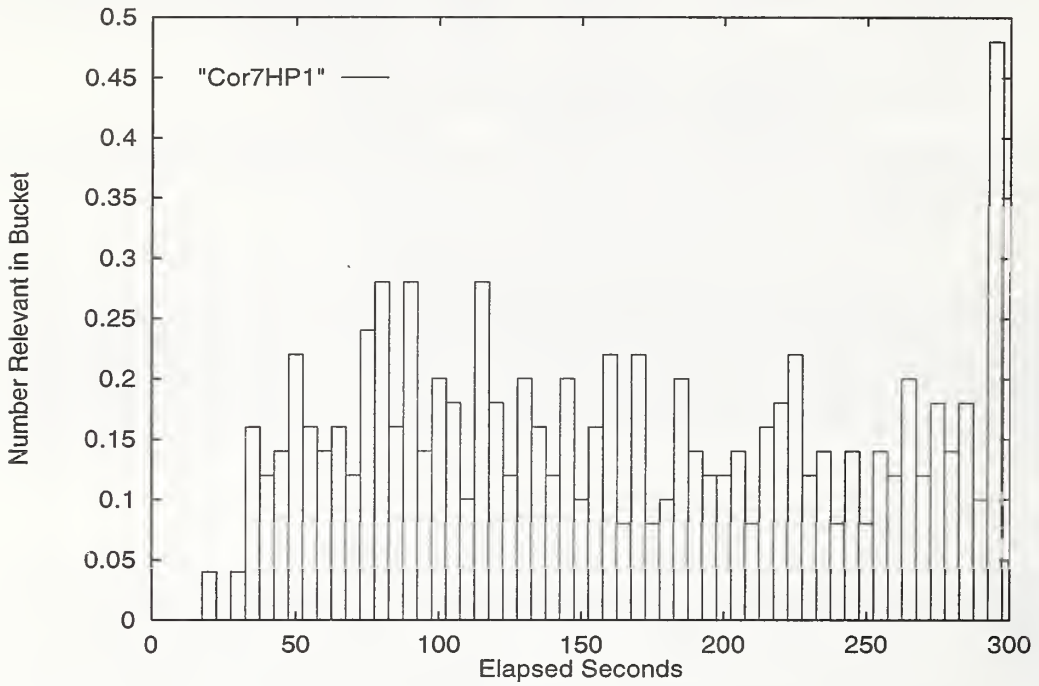


Figure 1: Average Relevant Retrieved per 5 second Timeslice over 50 Queries

This graph is actually evidence against the conclusion reached earlier that the task was too easy. The rate at which relevant documents are being added close to 300 seconds is still substantial. The previous evidence indicates it can't go on for much longer, and that less than half of the queries are still active. However, there is no sudden drop-off as there would be if this particular run finds too many relevant documents.

Figure 2 compares all three users on a typical single query, Query 366. The measure being plotted is precision at 15 documents. As was discussed earlier, User 2 typed in shorter queries so started judging documents earlier than the others. User 2 maintains a lead up until 180 seconds, when User 1 takes over. Then at 240 seconds, User 3 takes the lead for the last minute.

For this particular query, it is clear that User 3 has the best end result (precision after 5 minutes). But it is also clear that User 2 and possibly User 1 have better sessions: they find relevant documents sooner during the first 4 minutes.

Figure 3 gives the same comparison except on the average of all 50 queries. Once again, User 2 has the lead for most of the session up until the very end when User 3 takes over. For most of the session, User 2 is about 10 seconds ahead of User 3 and 20 seconds ahead of User 1. Again, User 3 has the best end result, but User 2 had the best session.

Other evaluation measures give the same overall results. For example, Unranked Average Precision at 15 documents is given in Figure 4. The curve is almost identical.

One different evaluation measure is Utility(1,-1,0,0) in Figure 5. This measures increases by 1 when a relevant document is retrieved and decreases by 1 when a non-relevant document is retrieved. It is a poor evaluation measure for the HP task. It is dominated by the retrieved non-relevant documents; i.e., those documents for which user and assessor disagree on relevance. None-the-less, the results are informative.

User 2's lead is even more substantial (remember User 2 has the most accurate judgements as measured by agreement with assessors). But what is very interesting is how the plots for User 2 and User 3 flatten out over

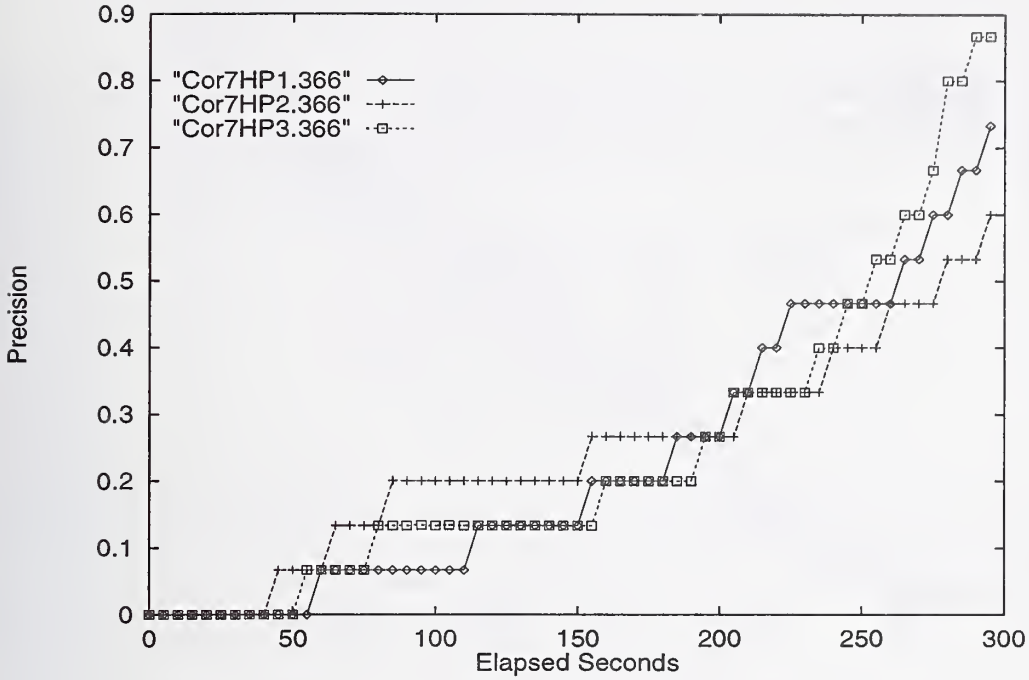


Figure 2: Precision (at 15 Documents) vs. Time for Query 366

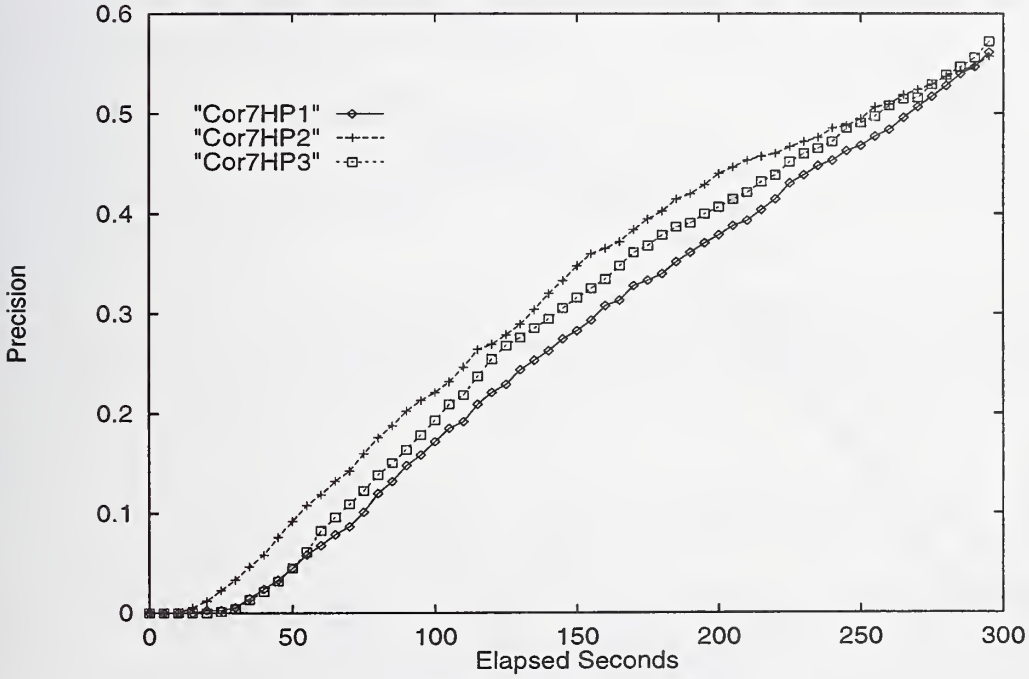


Figure 3: Precision (at 15 Documents) vs. Time over 50 Queries

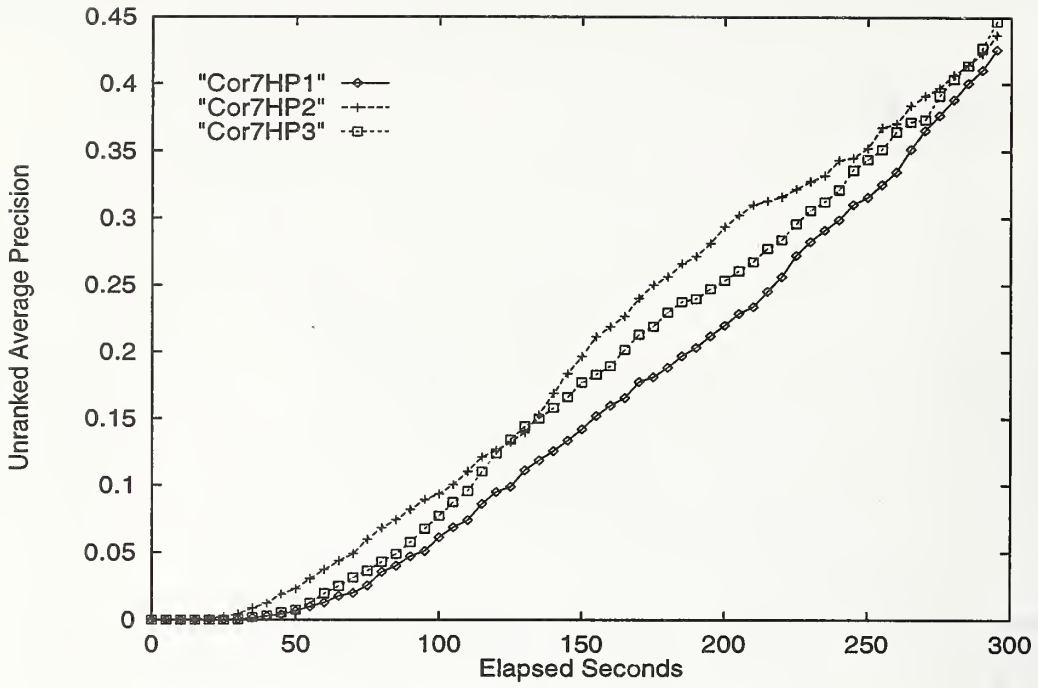


Figure 4: Unranked Average Precision vs. Time over 50 queries

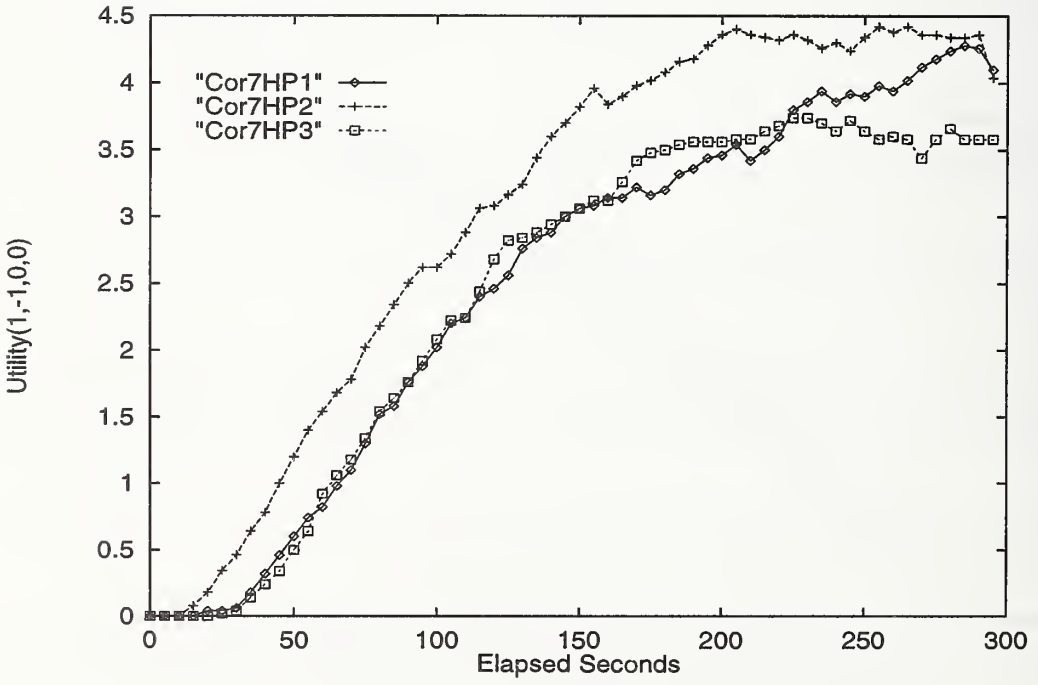


Figure 5: Utility(1,-1,0,0) vs. Time over 50 queries

the last 2 minutes. For every relevant document being added, a non-relevant document is being added. This may indicate more disagreements occur late, or maybe there is a natural stopping spot late. Further study is needed, especially since the handling of "iffy" documents may be partly responsible for the effect.

All of these time-based measures and graphs suggest that a reasonable evaluation measure for an entire session is the area under each plot, much in the same way as the area under the recall-precision curve is a good single measure (this is "average precision"). Table 5 gives three such measures, corresponding to the three different plots seen above. As expected, for all 3 session measures, User 2 has a substantial (6% – 8%) lead over User 3 and even more over User 1.

Run	Average Precis	Average UAP	Average Utility(1,-1)
Cor7HP1	.2726	.1590	3.997
Cor7HP2	.3104	.1934	4.606
Cor7HP3	.2901	.1780	4.287

Table 5: Timing Evaluation

These session evaluation measures can be extended to work on any time-based retrieval. It would be very interesting to apply these measures to the standard Manual portion of the ad-hoc task. Perhaps for TREC 8, we can request that timing figures be optionally supplied, perhaps as the iteration field, to Manual submissions. There are still open questions regarding these measures. A couple that immediately spring to mind is how sensitive they are to starting time, and to size of time-slice. However, they still seem to offer a hope at bringing efficiency into evaluation of manual systems and sessions.

Note that the latest copy of trec_eval is in pub/smart/trec_eval.7.0beta.tar.gz on ftp.cs.cornell.edu and includes all the measures discussed here plus many others, though perhaps not in their final form (for instance, the timing information is assumed to be in the "sim" field but will probably be moved.)

Examples and Failure Analysis.

After the HP results were received back from NIST, the three users were asked to write a sentence or two about each query. The following comments (paraphrased in some cases) give some insights into weaknesses of the system. There were a fair number of comments about disagreements with assessors, but those are ignored here.

Query 353: Antarctica exploration

- User 3: I misspelled query as "Antartica" and didn't notice for 2 minutes (though I noticed something was wrong and revamped weights)!

Query 354: journalist risks

- User 2: I think I tried "journalist hostage" first; I got onto a single case of a journalist kidnapped by Colombian drug lords, got a bunch of non-relevant Colombian documents, and then got a few more relevant and ran out of time (first relevant document had way too many other Colombian details for the feedback to work on)

Query 376: world court

- User 2: the first relevant documents, about the World Court refusing to hear Libya's case, pulled up voluminous Libyan stuff that I couldn't get past

Query 381: alternative medicine

- User 1: unexpectedly difficult to get "alternative medicine"
- User 2: I probably tried "alternative medicine" first, then apparently added "nontraditional", "acupuncture" (where my first relevant document brought up all sorts of stuff on drug treatment), ...
- User 3: Couldn't find specific examples (that were judged relevant).

Query 383: mental illness drugs

- User 2: lots of articles on treating drug abuse by the mentally ill, and for some reason I didn't seem to get through as many articles as usual.
- User 3: Extremely frustrating. Never able to find that first relevant document though there are lots out there.

Query 389: illegal technology transfer

- User 1: tough because query is high-level (concept not well-represented by keywords)
- User 2: once I got a few articles, I got stuff directly on COCOM as well as violations of it and what new rules might be adopted
- User 3: Never got any relevant documents.

Overall, the comments indicate there are two system weaknesses that we may want to address in the future. The first is that for a number of queries, it is very difficult to find any relevant documents. Instead, the user spends their time plowing through piles of very similar non-relevant documents. Perhaps the user should be offered the option of "Find different documents" after a couple of iterations of normal search. The system should use the same query but come up with documents that are different from each other and from previously examined documents.

The second observed weakness is that the system occasionally becomes too focused on one sort of relevant document, and is unable to find any other sort. The "Find different documents" option should help here also. The system should emphasize the original query, and should retrieve documents different from the relevant documents seen before.

The question of whether either or both of these uses of "Find different documents" can be decided upon automatically by the system is an interesting one, and deserving of further study. It suggests a slightly different sort of negative relevance feedback based upon avoiding previously seen *clusters* of either relevant or non-relevant documents.

Ad-hoc Task

Over the past year since TREC 6, we tried a number of different variations of our algorithms in order to improve performance. We looked at, or re-visited, stemming, phrasing, alternative clusterings, emphasizing titles, and emphasizing beginnings of documents. Unfortunately, none of these minor variations improved performance enough to be worth adopting. This suggests we need to go back to some of our more radical variations of the past (e.g., ITL or SuperConcepts) to improve effectiveness. We ran out of time to do that this year.

Ad-hoc Methodology

The basic approach we used for this year's TREC ad-hoc task is almost identical to our TREC 6 clustering approach. Unlike in previous years, we only used one algorithm (no experimental algorithm this year!), and ran it on different topic lengths. Our TREC 6 paper [1] gives the details and rationale for the approach. The basic algorithm is

1. Retrieve 1000 documents using the initial query (using *Lnu.ltu* weights).
2. Generate cooccurrence information about the query terms from the top 1000 documents.
3. Rerank the top 50 documents as in TREC 5 (using correlation and proximity information).
4. Assume the top 20 documents relevant, documents ranked 501–1000 non-relevant.
5. Generate clusters for the top 30 documents and save the best (most heavily weighted) terms from each cluster vector.

6. Rank the cluster vectors according to their similarity to the original query (using *bnn* weights for the clusters) and select the best 2 clusters.
7. Expand the query by 25 words and 5 phrases using Rocchio expansion with $\alpha = 8$, $\beta = 8$, and $\gamma = 8$. The expansion terms are selected from among the saved terms for both clusters and the actual number of terms selected from a cluster is proportional to its similarity to the original query.
8. Retrieve the final set of 1000 documents using the expanded query.

Ad-Hoc experiments and analysis

We submitted three runs in the ad-hoc category, all using the same algorithm. Cor7A1clt uses only the title field of the topics, Cor7A2rrd uses only the description field of the topics, and Cor7A3rrf uses the entire topic description. (Note that Cor7A1clt should really be named Cor7A1rrt for consistency's sake.)

Table 6 shows the results for the various runs across 50 queries. Unlike last year, we get a very pleasing performance improvement as we increase the amount of the query text we use. As always, though, the averages hide a great deal of variation at the query-by-query level. For example, the title only run scores higher than the full topic run for 19 queries; almost half! Most of those differences are small, but the full text can on occasion help immensely. For example, for query 398, "dismantling Europe's arsenal", the title only query scored .0011 in average precision but the full text query scored .5051 (and the description only query actually scored .5523).

The absolute level of performance is considerably higher than last year. Even the title only run this year beat all of our official runs last year. Given the lack of change with the system, it is obvious that the task this year was considerably easier for us.

Run	Average precision	Total rel retrieved	R precision	Precision @100 docs
Cor7A1clt	.2329	2621	.2564	.2106
Cor7A1rrd	.2543	2894	.2782	.2338
Cor7A1rrf	.2674	3198	.2953	.2584

Table 6: Ad-Hoc results (50 queries)

Table 7 shows that our runs compare reasonably with other runs. It is hard to tell much about relative performance since all automatic ad-hoc runs enter the same comparison pool this year, unlike in previous years where they were sorted by length. Even the title only run was above the median for the majority of the queries, which is impressive since it is being compared against many runs using the full topics. There are 86 runs in the comparison pool; having 5 best queries is quite respectable.

Run	Task pool	Best	$\geq$ median
Cor7A1clt	automatic	0	27
Cor7A1rrd	automatic	3	37
Cor7A1rrf	automatic	2	42

Table 7: Comparative automatic ad-hoc results (50 queries)

Query Track

General IR research is being held up because we don't have enough queries of various types to investigate advanced retrieval techniques that are query dependent. There's no way we can get enough relevance judgements on new queries to form a good query pool. The Query track looks at multiple query variations of past TREC topics to get a large number of query formulations.

The track guideline states four goals:

1. Start investigating the split between query formation/analysis and back-end engines. Evaluating what makes a good general query formation approach.
2. Get many variations of the same topic so we can start analyzing (including with strong NLP approaches) queries, and determining what sorts of things we want to pull out of queries.
3. Get a collection of mixed fact/content queries. For decades we've had systems (eg Pnorm) that can handle these, but haven't been able to evaluate and compare due to lack of a query collection.
4. Get a collection of reasonable very short queries, more typical of real-life ad-hoc queries.

Each group forms variations of each of the 50 topics in some subsets of the following categories (as defined in the guidelines):

1. Very short: (2-3 words) based on topic.
2. Sentence: NL (natural Language), based on topic and judgements
3. Manual Feedback: Manual NL sentence based on reading 5 or so relevant documents without reference to the topic (done by someone who doesn't have the topics memorized and who might use different vocabulary than the topic). An attempt to get a sentence which might use different vocabulary than the topic.
4. Manual structured query: based on topics and judgements. Perhaps mixed fact and content queries. Perhaps result of manual NL analysis.
5. Automatic structured query: based on topics and judgements (Note that "structure" could be just a list of words, or could be very complicated based on semantics.) Perhaps the result of automatic NL analysis.

Then all groups run everybody's queries for some subset of the categories above (whatever categories their system can be made to support). The names of the submitted runs consist of 7-8 letters/digits. The first 3 letters identify the group running the query. The last 4-5 letters are the queryset id, including category. Thus, "CorAPL5a" would be Cornell running the first Category 5 query set that was constructed by APL.

Query Track Methodology

This was the first year for the query track. As it ended up, only two groups participated in the track. Thus it is impossible to come up with as many conclusions as we had wanted.

The two groups are us (Cornell/SabIR) and the APL Labs at Johns Hopkins. We constructed one set of queries in each of the 5 categories; pretty much directly using the definitions of the categories. APL constructed 4 query sets, skipping category 3 and 4, but having two versions of category 5. For the first two categories, APL deliberately tried to construct different queries than the obvious choice of words. This increased query variability, though at a cost of overall effectiveness as we will see later.

All 5 sets of queries were easy to construct. Our category 4 queries do not have much detailed structure; they are basically a weighted sum of a vector query and a pnorm query. Our category 5 queries are straight weighted relevance feedback vectors. The most difficult part of category 4 and 5 queries was reverse engineering the stemming of terms, so that we could supply weighted unstemmed terms to other groups.

The queries are all constructed in DN2 format. DN2 is a quite complicated query language, but luckily very few features needed to be known for the queries the two groups constructed. We did not run directly on the DN2 queries but translated them back and forth from normal SMART queries.

Query Track Results

Table 8 gives our results on running the 9 query set variations (5 variations from Cornell and 4 from APL). The runs all strongly differ from each other in results; depending on the evaluation measure, the differences go up to 430%. In general, the Cornell queries performed better for us than the APL queries. Part of that is that goals of the APL queries were explicitly to use different, possibly non-optimal, vocabulary. But part of it could be that

Run	Ave Prec	R Prec	NumRelRet
CorCor1	.2457	.3066	6877
CorCor2	.3367	.3901	9056
CorCor3	.2020	.2774	6690
CorCor4	.3282	.3743	8674
CorCor5	.4586	.4861	10476
CorAPL1a	.1051	.1583	4438
CorAPL2a	.1142	.1633	4239
CorAPL5a	.1971	.2600	6119
CorAPL5b	.3219	.3727	8748

Table 8: Results of Cornell Runs on Different Query Sets

we constructed queries to suit our system. In particular, the query set Cor5 was constructed using relevance feedback based on Cornell document weights. How well these weights suit other systems remains to be seen.

As normal, even with the very strong overall differences in results between query sets, large numbers of individual queries of the weaker query set do better than the corresponding query in the stronger set. Table 9 gives the number of queries (out of 50) for which one query set beats another. For instance, APL5b beat Cor2 on 38 queries, despite having weaker overall evaluation averages.

>	Cor1	Cor2	Cor3	Cor4	Cor5	APL1a	APL2a	APL5a	APL5b
Cor1	0	7	32	11	2	43	39	30	18
Cor2	43	0	46	23	4	48	47	43	22
Cor3	18	4	0	5	1	38	36	26	12
Cor4	39	27	45	0	8	48	47	41	23
Cor5	48	46	49	42	0	50	49	48	46
APL1a	6	2	11	2	0	0	27	9	2
APL2a	11	3	14	3	1	22	0	16	3
APL5a	20	7	24	9	2	40	32	0	15
APL5b	32	28	38	27	4	48	47	35	0

Table 9: Comparative Query (row better than column for X queries)

There is a tremendous amount of query variability hidden in the comparative averages. We need to understand this variability. It is not clear that 9 query variations is enough to get a handle on variability; but at least it is a start.

Comparison with past TREC's

It is difficult to determine how much systems are improving from TREC to TREC since the queries and the documents are changing. For example, in TREC 3 the "Concept" field of the queries was removed. These terms proved to be very good terms for retrieval effectiveness in TREC 1 and TREC 2; thus the TREC 3 task without them is a harder task than previous TRECs. The TREC 4 task was more difficult since so much more of the text was removed from the queries. TREC 5, TREC 6, and TREC 7 continued using short queries which seem more difficult. Also, the average number of relevant documents per query has been steadily reduced every year, going from 328 in TREC 1 to 92 or 93 for the past two years. Very broad (and easy) queries have been eliminated.

To examine both how much SMART has improved over the years of TREC, and how much harder the TREC ad-hoc tasks have gotten, we ran our 7 TREC SMART systems against each of the 7 TREC ad-hoc tasks. Actually, we present two versions of the TREC 7 task. In the first version, we use the description field only; in the second version we use the title plus the description field. This emphasizes the core concepts of each topic.

Table 10 gives the results. Note that the indexing of the collections has changed slightly over the years so results may not be exactly what got reported in previous years. In the interest of speed, we ran our current implementation of the query and document indexing and weighting.

Methodology and Run	TREC 1 Task	TREC 2 Task	TREC 3 Task	TREC 4 Task	TREC 5 Short	TREC 6 DESC	TREC 7 DESC
TREC 1: ntc.ntc	.2442	.2615	.2099	.1533	.1048	.0997	.1137
TREC 2: lnc.ltc	.3056	.3344	.2828	.1762	.1111	.1125	.1258
TREC 3: lnc.ltc-Exp	.3400	.3512	.3219	.2124	.1287	.1242	.1679
TREC 4: Lnu.ltu-Exp	.3628	.3718	.3812	.2773	.1342	.1807	.2262
TREC 5: Exp-rerank	.3759	.3832	.3992	.3127	.2046	.1844	.2547
TREC 6: Rrk-clust	.3765	.3835	.4011	.3073	.1978	.1768	.2510
TREC 7: Rrk-clust	.3778	.3839	.4003	.3142	.2116	.1804	.2543
% Change from ntc.ntc	+55	+47	+91	+105	+102	+89	+124

Table 10: Comparisons of past SMART approaches with present

Comparing the columns of Table 10 gives an indication of how much harder the TREC task has gotten during the 7 years of TREC. Five quite different versions of the same system all do from 45% to 65% worse, in absolute numbers, on the TREC 7 task as compared to the TREC 1 task. The TREC 1 and TREC 2 figures are about the same. Performance starts to drop in TREC 3 and 4 when the queries get progressively shorter. The short high-level queries of the last 3 TRECs prove very difficult for all versions of SMART.

Comparing the rows of Table 10, it is obvious that our results with our TREC 7 approach are not noticeably different from our TREC 5 or TREC 6 approach.

Conclusion

This year, Cornell and SabIR Research participated in the High-Precision and Query tracks, as well as doing the base ad-hoc task. Once again we did very well in all the tracks, ahead of the median in all tracks. (Though that does not mean all that much in the Query Track with 2 participants.)

In the High-Precision area, we looked in-depth at methods of analyzing and evaluating time-dependent retrieval sessions. We came up with several new evaluation measures that seem to capture the essentials of what a session evaluation of manual retrieval should capture. These approaches may be quite useful outside of the High-Precision track, perhaps to evaluate timed Manual retrieval.

References

[1] Chris Buckley, Mandar Mitra, Janet Walz, and Claire Cardie. Using clustering and superconcepts within SMART : TREC 6. In E. M. Voorhees and D. K. Harman, editors, *The Sixth Text REtrieval Conference (TREC-6)*. NIST Special Publication 500-240, 1998.

[2] Chris Buckley, Gerard Salton, and James Allan. Automatic retrieval with locality information using SMART. In D. K. Harman, editor, *Proceedings of the First Text REtrieval Conference (TREC-1)*, pages 59–72. NIST Special Publication 500-207, March 1993.

[3] Chris Buckley, Amit Singhal, and Mandar Mitra. New retrieval approaches using SMART : TREC 4. In D. K. Harman, editor, *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*. NIST Special Publication 500-236, 1996.

[4] Chris Buckley, Amit Singhal, and Mandar Mitra. Using query zoning and correlation within SMART : TREC 5. In D. K. Harman, editor, *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*. NIST Special Publication 500-238, 1997.

[5] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In Hans-Peter Frei, Donna Harman, Peter Schauble, and Ross Wilkinson, editors, *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29. Association for Computing Machinery, 1996.

ACSys TREC-7 Experiments*

David Hawking
CSIRO Mathematics and Information Sciences,
Canberra, Australia
Nick Craswell and Paul Thistlewaite
Department of Computer Science, ANU
Canberra, Australia
David.Hawking@cmis.csiro.au, {pbt,nick}@cs.anu.edu.au

January 27, 1999

Abstract

Experiments relating to TREC-7 Ad Hoc, HP and VLC tasks are described and results reported. Minor refinements of last year's Ad Hoc methods do not appear to have resulted in worthwile improvements in performance. However, larger benefits were gained from automatic feedback than last year and concept scoring was very beneficial in the Manual Ad Hoc category. In the Automatic Ad Hoc category title-only performance seems to have suffered more severely than long-topic from a number of lexical scanning shortcomings and from an excessive stopword list. The HP track was used to validate the usibility of the combination of PADRE and the Quokka GUI. In the VLC track, the 100 gigabyte collection was indexed in under eight hours and moderately effective queries were processed in less than two seconds.

1 Introduction

The work reported here comprises a number of text retrieval experiments conducted within the framework of TREC-7 and addressing questions of interest in the following research areas: Scalable information retrieval; Query term weighting; Concept-based relevance scoring; User-efficient retrieval interfaces and Automatic query generation.

ACSys completed Automatic and Manual Adhoc, High Precision and VLC tasks.

2 Method

2.1 Relevance Scoring Methods Employed

As in TREC-6 [Hawking et al. 1997], the basic relevance scoring method used in official ACSys adhoc runs was the Cornell variant of the Okapi BM25 weighting function [Singhal et al. 1995; Robertson et al. 1994]

$$w_t = q_t \times tf_d \times \frac{\log(\frac{N-n+0.5}{n+0.5})}{2 \times (0.25 + 0.75 \times \frac{dl}{avdl}) + tf_d} \quad (1)$$

*The authors wish to acknowledge that this work was carried out within the Cooperative Research Centre for Advanced Computational Systems established under the Australian Government's Cooperative Research Centres Program.

where w_t is the relevance weight assigned to a document due to query term t, q_t is the weight attached to the term by the query, tf_d is the number of times t occurs in the document, N is the total number of documents, n is the number of documents containing at least one occurrence of t, dl is the length of the document and $avdl$ is the average document length (both measured in bytes).

On average, the Okapi probabilistic model performs well but there are cases where it does not. For example, a query derived from the topic "vitamins and health" is likely to comprise a set of vitamin words and a set of health words. For a document to be relevant, it should contain instances of words from both sets, but a sum-of-weights formula such as Okapi does not recognise this.

ACSys/ANU TREC submissions have focussed on scoring methods which tend to reward the conjunction of multiple concepts (e.g. the *vitamin* and *health* concepts in the example above). Our TREC-4 manual adhoc approach used *distance-based scoring* [Hawking and Thistlewaite 1995; Hawking and Thistlewaite 1996]. In TREC-5, ACSys/ANU manual queries were scored in the same way and a largely unsuccessful effort was made to provide semi-automatic assistance in query generation.

Neither TREC-4 nor TREC-5 manual runs involved any interaction, a fact which probably led to easily avoidable failures on some topics. As might be expected, distance-based queries perform more successfully in an interactive environment. [Cormack et al. 1997] In TREC-6, ANU/ACSys [Hawking et al. 1997] introduced a weaker method for rewarding concept co-occurrence *concept scoring* in the manual categories which was shown to produce a small but worthwhile benefit, on average.

This year, concept scoring was used in the title-only Automatic Adhoc submission, where each non-stop title word was assumed to represent a concept, and in Manual Adhoc, where concepts generated in the same way were modified by the human searcher. No means were to hand for automatically assigning terms from the description and narrative fields to concepts and, consequently, concept scoring was not used in the longer Automatic Adhoc categories. However, in these runs, a *Term Coordination* measure was used to reward documents with a wide spread of query terms. (See Section 2.1.3.)

2.1.1 Frequency Scoring

The basic Okapi relevance scoring method defined in Equation 1 will from now on be referred to as *frequency scoring* to distinguish it from the other methods.

2.1.2 Concept Scoring

As previously mentioned, groups of related terms in a query are called concepts. Documents are scored against each concept and the results are recorded in separate accumulators. The final score s for a document is derived from the concept scores $c_1, \ldots c_n$ using $s = (k_c c_1 + 1) \times \ldots \times (k_c c_n + 1)$. A value of $k_c = 1$ was used in concept scoring experiments reported here.

2.1.3 Term Coordination

When term coordination was in force, scores derived from the basic Okapi formula were multiplied by a term-coordination factor as follows:

$$S' = \frac{k_t + num\_qterms\_present}{k_t + num\_qterms} . S$$

A value of $k_t = 10.0$ was used in term coordination runs reported here.

2.2 Run-naming Convention

All ACSys TREC-7 runs consist of the string `acsys7` followed by a suffix which indicates the task category. Additional suffixes may be appended to differentiate runs in the same category. Table 1 lists the runids of official runs.

Table 1: Runids for the official ACSys submissions. Runids of submissions in the VLC track give the size of the collection, and the number of terms in the queries used. Thus **acsys7_100_2** is a VLC run over 100 gigabytes of data using 2-term queries.

Name	Category
acsys7as	Automatic Ad Hoc, short topic (T)
acsys7am	Automatic Ad Hoc, medium topic (T+D)
acsys7al	Automatic Ad Hoc, long topic (T+D+N)
acsys7mi	Manual Ad Hoc, interactive
acsys7hp	High Precision Track
acsys7_g_n	VLC Track, n-term query over g gigabytes of data.

2.3 Training

Training was carried out using the TREC-7 data and TREC-6 topics after removing the Congressional Record documents from the TREC-6 qrels.

2.4 Hardware and Software Employed

The PADRE retrieval system used in previous TRECs has undergone further evolution, mainly to improve efficiency. The current version is known as PADRE98 and it was used in all experiments reported here. A Dell Latitude laptop PC running Linux and a Sun Ultra-1 workstation were used in the Adhoc runs. In the VLC track, a cluster of eight DEC Alphas was employed.

Interactive query modification was carried out using the **quokka** graphical user interface to PADRE.

2.5 Statistical Testing of Differences Between Runs

Throughout this paper, wherever comparisons are made between pairs of runs, apparent differences between means have been tested for statistical significance using two-tailed t-tests with $\alpha = 0.05$.

2.6 Automatic Query Generation

The basic approach to automatic query generation was as described in last year's TREC paper, except for the changes described in this section. [Hawking et al. 1997]

The goal of experiments using automatic query generation was to provide preliminary answers to the following questions:

1. How can the Concept scoring method be used with automatically generated queries?
2. Can the use of frequency within the topic as a query-term weight be improved upon?

The basic relevance scoring method used by ACSys (see Equation 1), makes use of a query term weight q_t. In TREC-6, q_t was simply a count of how many times the query term occurred in the part of the topic statement being used.

In TREC-7, an attempt was made to calculate better query term weights by using a modified Okapi formula, in which the accessible part of the query statement was treated as a document and in which *df* values were derived from the body of text represented by TREC topics 1-400. The main effect of this

is to reduce the weight of terms occurring in many topic statements (such as "document", "identify", "relevant", etc.) and to reduce the boost given to terms occurring many times in the topic statement.

A simplistic attempt was also made to identify terms which were more central to the topic, by artificially boosting the frequency of occurrence for non-stopwords in the title field by a constant (k_e). A value of $k_e = 1.5$ was found to be effective.

As previously mentioned, in the title-only run it was assumed that each non-stop title word represented a separate concept. In addition, phrases were not generated for the title-only task, as training suggested that they harmed performance in this category (only).

2.7 Relevance Feedback

The pseudo-relevance feedback mechanism based on hotspots (passages) in the retrieved documents was used almost unchanged from TREC-6. However, the changed method for calculating query-term weights necessitated a change in the interpretation of the w_0 parameter (the query-term weight of the best term selected by feedback). In TREC-6, it was a constant ($w_0 = 0.75$) but in TREC-7 it was interpreted as a fraction to be multiplied by the maximum query-term weight of any of the original terms.

2.8 Parameters Used in Official TREC-7 Runs

Table 2: Parameter settings used in the official Automatic Adhoc runs. k_e - emphasis added to terms occurring in the title field; $avlen_q$ - value used for average topic length in calculating query-term weights; k_t - term coordination constant; k_c - concept scoring constant; T - Number of top-ranked documents examined during relevance feedback (RF); p - Proximity range (in characters) used to define RF hotspots; n - Number of RF terms extracted; w_0 - multiplied by maximum query-term weight to give weight of best RF term. Times per query are elapsed times in seconds measured on a 167 MHz,256 Mbyte Sun Ultra-1. Figures in parentheses are the corresponding figures for a 266 MHz Pentium-2 Dell Latitude laptop.

	acsys7as	acsys7am	acsys7al
Topic fields	T	T + D	T + D + N
k_e	0	1.5	1.5
$avlen_q$	20	1	15
k_t	NA	10.0	10.0
k_c	1.0	NA	NA
T	20	20	20
p	500	500	500
n	20	20	20
w_0	0.4	0.3	0.2
#terms	2.4	6.2	17.0
#phrases	0	3.8	10.5
Time/query	10.9(17.0)	16.5(23.1)	33.3(41.9)

Parameters used in Automatic Adhoc runs are detailed in Table 2.

2.9 Query Optimisation

In various TREC tasks, such as HP and VLC, query processing was optimised by the simple expedient of ranking the query terms in order of decreasing q_t and processing only the top k of them. This behaviour

is controlled by including `MAXTERMS = k` directives in the query stream.

3 Results

3.1 Automatic Adhoc Results

Table 3: Average Precision performance of ANU/ACSys Automatic Adhoc runs relative to official runs in the same category. The number of topics for which the run achieved best (possibly equal best) performance and the number achieving median or better (in both cases relative to *all* automatic runs, not just those using the same topic fields) are tabulated in the rightmost two columns. The rank is relative to automatic runs of the same topic length. The starred run was an unofficial run performed under official conditions and prior to the deadline. There were 50 topics.

Run-id	Category	Mean	Rank	#best	#≥med.
acsys7as	T	.2045	8	2	26
acsys7am*	T+D	.2230	6*		
acsys7al	T+D+N	.2659	7	2	42

Table 4: The same runs as in table 3, compared on the basis of overall recall (percentage of known relevant documents retrieved, averaged across 50 topics).

Run-id	Category	Percent	#best	#≥med.
acsys7as	T	61.7	2	37
acsys7am*	T+D	68.1	6*	40
acsys7al	T+D+N	72.7	4	44

Table 5: The same runs as in table 3, compared on the basis of P@20. Best and median data was not available.

Run-id	Category	Mean
acsys7as	T	0.337
acsys7am*	T+D	0.359
acsys7al	T+D+N	0.439

Results for Automatic Adhoc runs are summarised in Tables 3, 4 and 5. The long topic run outperformed the title-only run by considerable margins (30%, 30%, and 18% for average precision, P@20 and recall respectively). The gaps between long topic and title-plus-description are smaller (19%, 22% and 7% respectively) but still statistically significant. In training runs (see Section 2.3), these differences had been much smaller.

3.2 Automatic Adhoc Discussion and Conclusions

This year's attempted improvements in automatic query generation appear to have been founded on too little training data. Relative to comparable runs performed using last year's methods, this year' runs achieved "gains" of +6%, -1% and -3% in average precision for long, medium and short topics. None of these differences were statistically significant. This was disappointing, as training had been particularly focussed on the shorter length topics.

With the assistance of the Okapi group, an analysis was undertaken of why ACSys short-topic queries were relatively poor. The following reasons are believed to explain the major part of the differences:

1. Lexical and stop-word issues had large effects on certain topics. In the topic *R&D drug prices*, R&D was eliminated in lexical scanning and prices was (for some peculiar reason) on the stop list. The resulting query was the ineffective *drug*. The version of PADRE in use treated only the first 12 characters of a word as significant, but the same restriction did not apply to the query generator. Consequently, results for *oceanographic vessels* (including a 13-letter word) were quite poor.

2. Okapi made use of a small number of pre-defined synonym classes. On certain topics, the addition of synonyms (such as *Malvinas* to *Falkland* and *channel tunnel* to *chunnel*, caused a considerable gain.

3. Passage retrieval. The Okapi group reported a small overall gain from use of passages prior to pseudo relevance feedback. ACSys did not use passage retrieval.

Without removal of these differences, it is not possible to compare the relative effectiveness of the two variants of the BM25 formula used, nor to compare the results of the relevance feedback methods.

4 Manual Query Generation

Manual AdHoc, Official Run `acsys7mi`, Corrected Runs `acsys7mi2` and `acsys7mi2rf`

4.1 Manual Query Generation Process

A reasonably experienced user (the first author) interactively generated a set of manual queries starting with an initial automatic set. The initial queries were generated from the full topic descriptions and were similar to the queries used in `acsys7al` but used neither feedback nor phrases.

Concepts corresponding to the title words were generated automatically, with non-title words being initially assigned to the last concept. The Quokka GUI allowed user-efficient deletion of terms and assignment of terms to concepts.

The task was approached in a similar fashion to the High Precision track except that a) more time was allowed and b) the goal was to produce a generally-useful single query which could then be run to achieve good results, particularly on the early precision measure.

Query modification proceeded in a number of phases:

1. Blind (no reference to documents) refinement of the query by: a) restructuring (if necessary) the concepts, b) removing terms thought to be of low value, c) assigning terms to the appropriate concepts, and d) adding new terms thought to be useful. The goal was to produce an initial document ranking with as many relevant documents as possible in the first ten.

2. The query was then run by PADRE with a restriction on the number of active document accumulators and light query optimisation (MAXTERMS=15). These steps were intended to reduce user waiting time. The titles of the top 100 documents were displayed in a scrolling window, each with coloured squares indicating the presence of concepts within the document.

3. The user then examined documents to evaluate the performance of the query. Sometimes they were chosen from the top of the ranking; at other times documents flagged as containing evidence for all of the concepts were chosed from further down the list. Query terms occurring within displayed documents were automatically highlighted in the colour assigned to the concept for which they provided evidence. A decision was made as to the document's relevance and recorded as a red or green square beside the document title.

4. After reading a few documents, it was usually clear that one of the following applied:

   (a) The query was performing very well and needed no further modification.

   (b) The query was performing well, but a list of additional terms had been identified which could improve it. These were added and the query re-run.

   (c) Irrelevant documents were being returned due to distractor (eg. ambiguous) terms. In this case, terms might be removed (sometimes with addition of new terms) or negative-weight terms added and the query re-run.

   (d) The query was unbalanced and gave too much weight to some concepts. An indication of this appears in the pattern of coloured concept squares in the document list. This was addressed by some combination of: adjusting the query-term weights to increase the emphasis on neglected concepts; and adding new terms to neglected concepts.

5. Steps 2-4 were repeated until the user was satisfied with the proportion of relevant documents in the first ten or so documents or gave up.

6. The last-run version of the query and the top 1000 ranking resulting from running it were saved for submission.

The median time taken per topic was 10.6 minutes. A total of 533 minutes was required for all 50 topics. After completion of the task, any unjudged douments within the top 10 for each ranking were judged (time not recorded) in order to self-assess the performance of the queries. On average, 7.6 documents in the first ten were self-judged to be *appropriate*, which was felt to be a good result. Again based on self-assessment, there were only six topics for which less than half of the top 10 were appropriate. The definition of *appropriate* was weaker than the definition of *relevant*. Appropriate documents basically matched the query but not necessarily the topic definition of relevant. They may not, for example, have provided the specific examples or specific information demanded by the topic but not by the resulting query.

It was observed that long documents occurred very rarely among the first ten retrieved. Almost none of the documents examined were longer than two or three screenfuls.

4.2 Manual Adhoc Results

Table 6 records the results of various manual runs with two long-topic runs for comparison.

Contrary to previous experience, the results for the frequency-scored version of the official manual run (`acsys7mif`) seem to show that concept scoring actually caused harm. Subsequent investigation revealed that this was a case of death by misadventure! It transpired that MAXTERMS=15 directives had been inadvertently left in the official run. Removal of the offending directives showed that this mistake had caused a 20% drop in average precision, a 7% drop in P@20 and a 9% drop in recall. All these differences were statistically significant, but may be underestimates because the effect of the MAXTERMS directive during the interactive phase is likely to have been more deleterious than intended or expected.

The harm caused by the term limit to the frequency-scored version of the official run was minimal because, in that case, only low-value terms were not processed. A frequency scored version of the corrected manual run showed that, in fact, concept scoring achieved a gain of 13% (significant) on average precison, a gain of 11% on P@20 at the expense of an apparent loss of 4% on recall (not significant).

Table 6: Results for Manual runs. Averages across 50 topics. Data for the automatic queries from which the manual queries were derived is also included. As mentioned in the text, the official `acsys7mi` run inadvertently included an optimisation directive which harmed performance. The corrected run `acsys7mi2` is identical but for the removal of this directive. Other runs included here show the benefit of concept scoring and pseudo relevance feedback. The suffix `nf` appended to a runid indicates that that the new run was identical except that relevance feedback was not used. Similarly, `rf` indicates a run variant in which relevance feedback was used and `f` indicates a variant in which frequency scoring was used instead of concept scoring.

Run-id	Description	Ave. Prec.	P@20	Recall
`acsys7alnf`	`acsys7al` minus rel. feedback	.2260	.400	69.3%
`acsys7mi`	Official manual ad hoc	.2669	.486	63.0%
`acsys7mif*`	`acsys7mi` minus concept scoring	.2786	.472	71.5%
`acsys7mi2`	Corrected `acsys7mi`	.3196	.525	69.0%
`acsys7mi2f*`	`acsys7mi2` minus concept scoring	.2816	.473	71.8%
`acsys7mi2rf`	`acsys7mi2` with rel. feedback	.3401	.530	74.2%

4.3 Manual Adhoc Discussion and Conclusions

The effect of the manual intervention can be gauged by comparing the corrected manual run with the `acsys7al` run a) without feedback, and b) with it. In the no feedback case, the former performed 41% better on average precision (significant) and 31% better on P@20 (significant). Recall levels were effectively identical. In the feedback case, the manual run performed 28% better on average precision (significant) and 21% better on P@20 (significant). Recall levels were again effectively identical.

In the non-feedback case, query processing times were substantially reduced in the (corrected) manual run, from an average of 17.3 sec. to 4.55 sec.

From the results of the corrected manual run, it is clear that a relatively small amount of manual editing can create quite dramatic improvements in precision and query processing speed.

Concept scoring again produced a worthwile gain in performance.

4.4 Benefits of Automatic Feedback

Table 7: The effect of automatic feedback on various query sets. Scores shown for the various runs apply to the no-feedback case. The percentage gain due to feedback is given in parentheses. Asterisks indicate statistical significance.

Run-id	Ave. Prec.	P@20	Recall
`acsys7as`	.1677(+22%*)	.317(+6%)	54.1%(+14%*)
`acsys7am`	.1940(+15%*)	.344(+4%)	61.8 %(+10%*)
`acsys7al`	.2260(+18%*)	.400(+10%*)	69.3%(+5%)
`acsys7mi2`	.3196(+6%*)	.525(+1%)	69.0%(+7%*)

Table 7 shows that relevance feedback has worked consistently well for all three automatically generated query sets and for the (corrected) manually modified set. All differences in average precision

were statistically significant. Three of the four query sets also showed significant gains in recall. It is interesting that the only query set which did not show a significant change in recall was also the only one to show a significant improvement in P@20.

5 High Speed (... sorry, Precision) Track

The High Precision task is probably the best overall test of a retrieval system's performance (provided that comparisons are not confounded by user variation.) Good performance on the HP task requires acceptable speed and effectiveness of the retrieval system and also an effective user interface but may be unaffected by small advantages on any single dimension. The HP task is also a good framework in which to evaluate the benefit or otherwise of bells and whistles. If users can't take advantage of them on the HP task, how practically useful are bells and whistles?

For this year's HP track, PADRE and the Quokka were used unaltered, except for the addition of a timer to the Quokka. All 50 topics were done in sequence by one user over the course of a single day. To ensure all judgements were entered within the five minutes, a time stamp was entered with each judgement according to the Quokka's timer. The timer was started immediately a new topic was visited, and the final sumbission was generated by taking all judgements with a timestamp of less than 300 seconds.

No attempt was made to overlap query processing with human reading of documents.

PADRE was run several times per query, probably two or three times on average. Three or four word queries were usually used on the first run, then four or five extra terms would be added over the course of the subsequent runs. The colour coded highlighting of query terms in the Quokka was useful for making fast relevance judgements on viewed documents.

Subsequent work on the HP task is likely to focus on analysing the amount of time spent by the user on the various activities such as: a) composing, editing and typing queries; b) waiting for PADRE to process queries; c) waiting for Quokka to display selected documents; d) reading relevant documents; e) reading irrelevant documents. Such an analysis should identify the most profitable areas to speed up, and how to set the balance between speed and effectiveness.

Our impression is that further improvement to PADRE query response time and reduction in the time taken to load a document in the Quokka are likely to be most beneficial. Adding new query expansion mechanisms such as automatic or interactive relevance feedback could both improve the quality of results and the amount of useful highlighting in each document. However, PADRE feedback is currently very slow and would need to be accelerated significantly for these mechanisms to deliver a nett benefit.

6 VLC Track

Although ineligible to win an ACSys medal, it was nonetheless our goal to meet the criteria. This constituted a considerable challenge, as PADRE's speed in last year's track was far lower than the required level.

6.1 Hardware Employed

Experiments were conducted using eight 266 MHz, 128MB EV5 DEC Alpha workstations connected by a 10 Mb/sec Ethernet network. An elderly SPARC 10 (60 MHz, 96MB) connected by a shared 10 Mb/sec network was used as the user interface. This was essentially the same hardware configuration as last year, but this time no use was made either of the RAID box connected to one of the workstations or of the faster ATM network connecting the machines. However, this time, a 9 gbyte external SCSI disk was connected to each node. The small internal disk on each node was used by the operating system and for swap space.

6.2 Indexing

As in the past, the data was divided approximately evenly across the available workstations. If the data allocated to a workstation was much more than one gigabyte, the data was divided into separately indexed *chunks*.

The indexing process for a chunk involves building a compressed inverted file, then post-processing it to produce an uncompressed but reasonably compact inverted file in which each posting is a (docno, score) pair. The score is a quantised version of the Okapi weight taking into account *df*, *tf* and document length.

During indexing (and during query processing), the various dictionary, document and index files are mapped into virtual memory using the `mmap()` call and treated as arrays. This reduces system overhead, and avoids both the need to manage buffers and the need for explicit I/O calls.

A reasonably efficient algorithm has been implemented to merge several separate chunk indexes into a single index.

6.3 Indexing Results

Last year, PADRE required 15.6 hours to index the 20 gB VLC. The advent of local disks on each of the workstations allowed the process to be parallelised and this year it took only 7.38 hours to index 100 gbytes of data. With better load balancing, this figure would have been 5.7 hours. These figures include the time to pre-compute relevance score contributions for every possible term-document combination.

Table 8 gives the indexing times for BASE1, BASE10 and VLC2 collections using all eight workstations. The time for BASE10 appears anomalous. This could be because of poor load balance caused chunks which were excessively large on one or more nodes, resulting in excessive virtual memory system activity.

Indexing of BASE1 was also carried out on one workstation. The time taken was 0.800 hours, a factor of 18.4 longer than in the eight workstation case. This is very likely to be due to virtual memory effects.

6.4 Query Processing Speed Results

Last year, PADRE required an average of 50.6 seconds to process queries over the 20 gbyte VLC. With more heavily optimised queries and considerable tuning, it was found possible to reduce average query processing times over 100 gbytes of data to 2.74 seconds, using the indexes and hardware described above.

Further improvement was hampered by the fact that the original data was divided into 12 (in one case 13) separately-indexed chunks per workstation. Consequently, there were 12 (in one case 13) separate term dictionaries and indexes on each machine. This organisation resulted in large numbers of page faults and consequent delay. Only a very small proportion of the elapsed query processing time is CPU time.

To address the problem, separately created data structures were merged. Groups of three (in one case four) data structure sets were merged, reducing the number of components per workstation to four. Merging was done in parallel across the nodes and the longest elapsed time was 0.35 hours, bringing total indexing time to **7.73 hours**. Having merged in this way, average query processing time dropped to 0.887 seconds for 2-term queries. This is a factor of about 280 better than last year, taking into account the data scale-up, on similar, even less-expensive, hardware. Average query processing time rose to 1.47 seconds for 5-term queries.

It appears that presentation of rankings (mostly looking up the docid) are taking about 0.3 sec per query and there is probably at least 0.1 sec of fixed overhead. It is therefore expected that completing the merging process to the point where each node has but one set of structures might result in reducing average query times down to about 0.5 sec but no further with present hardware. On the other hand, increasing the degree of merging still further is expected to allow the data size per node to be increased substantially while keeping query processing time below 1 second.

Additional memory or parallel disk I/O on each workstation could bring considerable benefit to both indexing and to query processing.

6.5 Speed Ups

There are a number of reasons why this year's processing speed is better:

1. Each workstation has its own local disk,
2. Queries are more heavily optimised,
3. Using a chair (eg. a block index) in front of the term table (to improve locality of reference and reduce page faults),
4. Using reference stats instead of communication between nodes to obtain dfs,
5. Relevance scores are pre-computed during indexing (in other words the inverted file contains the actual score for every term-document combination),
6. Limiting the number of document accumulators,
7. Using radix sorting of results (less than two full passes through the document accumulators are needed),
8. Increasing the size of data represented in each separate index.

6.6 VLC Effectiveness Results

Unfortunately, the dramatic improvements in speed have been achieved at the cost of retrieval effectiveness. Precision at 20 documents retrieved for the two-term (0.298, ranked 18/18) and five-term runs (0.442, ranked equal 12/18) over 100 gigabytes was at the bottom end of the table of participating groups. The median P@20 score for the 18 runs was 0.525.

However, when the six runs corresponding to manually generated or automatic full-topic queries are eliminated, the median average precision over 100 gigabytes drops to 0.442, meaning that, technically, the ACSys 5-term run did in fact (though barely) satisfy the medal conditions.

6.7 VLC Scalability Results

Run `acsys7_1_5` was repeated using only a single Alpha (run `acsys7_1_5S`) instead of eight. As expected, P@20 results were very similar (0.141 v. 0.139). The small difference (involving a total of only two relevant documents) is believed to be due to arbitrary differences in ranking of equal-scoring documents. The average query processing times were quite similar (0.061 sec. and 0.086 sec. respectively) indicating that parallelism introduces almost no query processing benefit in the BASE1 case in the conditions of the experiment.

Note that in the 1 gB, eight-node case, there is sufficient RAM (128MB per node) for all necessary data structures to remain resident, dramatically increasing the chance that document numbers will be memory resident.

The scale-up between the five-term query runs is shown in the three middle columns of Table 8. As may be seen, there is a dramatic increase in P@20 from 1 gB to 10 gB and a less dramatic but still significant jump from 10 gB to 100 gB. In the case of query processing time, the scale-up from 10 gB to 100gB is almost the same as the scale-up in collection size whereas the time scale-up from 1 gB to 10 gB is much less than the scale-up in collection size. It is almost certainly the case that the data-independent costs in query processing (query parsing and broadcast, document identifier look-up, result merging and

Table 8: Performance of PADRE98 on the three different collection sizes, averaged over the 50 Ad Hoc topics. Except for the results in the right-most column, all data relates to the same set of 5-term queries. All runs used eight DEC Alpha workstations. Query processing times are in elapsed seconds. Each is the average of three runs and each group of three runs was preceded by a warm-up query (a 51-term manual query for topic 254). Indexing time is in elapsed hours and is the time taken by the slowest node). Indexing was performed only once. Figures in parentheses give the scale-up factor over either BASE1 or both BASE1 and BASE10 as appropriate.

Measure	BASE1	BASE10	VLC2	VLC2 (2-term queries)
Runid	acsys7_1_5	acsys7_10_5	acsys7_100_5	acsys7_100_2
P@20	0.139	0.321 (2.31)	0.442 (3.18, 1.38)	0.298 (2.14, 0.928)
Indexing time	0.0434	1.71(39.4)	7.73(178,4.52)	-
Time/query	0.061	0.168 (2.75)	1.468 (24.1, 8.74)	0.887 (14.5, 5.28)

presentation) are responsible for the bulk of the time spent in the 1 gigabyte case. If the fixed overheads are assumed to be about 0.05 seconds per query, then the ratios correspond much more closely to the data scale-up factor.

The two-term run over the full collection was made: a) to achieve sub-second query processing times, and b) to investigate whether the expected increase in P@20 due to collection scale-up could be traded for a lower query processing time scale-up factor. As may be seen in Table 8, the two-term queries achieve more than twice the P@20 over 100 gB as do the five-term queries over 1 gB and almost the same P@20 as do the five-term queries over 10 gB, while achieving a significant saving in processing time.

6.8 VLC Discussion and Conclusions

At this stage, it is unclear why precision results were as relatively bad as they were. Possibilities include:

1. Inaccurate *df* estimates derived from last year's VLC track baseline.
2. Errors due to quantisation of pre-computed scores.
3. Too-draconian limit on number of document accumulators with possible bias against particular collections.
4. Query optimisation too aggressive or terms badly ranked.
5. Thresholding of postings too aggressive or completely ill-advised.
6. It is possible (but unlikely given the similarity to methods used by the Okapi group) that PADRE98's basic retrieval methods do not work as well on Web data.
7. Bugs. (Surely not!)

The combination of techniques such as radix sorting, relevance score pre-computation (with uncompressed index files) has the effect of reducing the CPU cost of query processing. At the same time, the combination of index thresholding and the MAXTERMS style of query optimisation ensures that postings lists are kept short and therefore limits the amount of data to be transferred from disk. The result is that disk seek time becomes the dominant factor. In PADRE98, the processing of each query term over each chunk of text is likely to require one I/O request (with seek and rotational latency) to locate the appropriate entry in the term dictionary and another to retrieve the postings list. In addition, a disk I/O is nearly always needed to obtain the name of a ranked document.

7 Conclusions

1. The small innovations introduced in the Automatic AdHoc runs do not appear to have been beneficial on the TREC-7 task. However, the T+D+N run again performed at a quite respectable level. In last year's Automatic Adhoc tasks, the relevance scoring model used by ACSys worked much better for the T+D+N task than for the shorter topics. Attempts to improve relative performance on shorter topics in TREC-7 seem to have been unsuccessful.

2. Once again, we have observed that results obtained in training using only one set of 50 topics do not necessarily generalise. When time permits it is always worth training on multiple topic sets.

3. ACSys Manual AdHoc participation seems to indicate that a small (highly topic-dependent, but averaging 10 minutes) amount of human intervention can significantly improve on automatically generated queries as far as P@20 and average precision are concerned. Automatic query generation may have "hit the wall" but humans are on the other side of it!

4. Participation in Manual AdHoc and High-Precision tasks has confirmed the view of the users (also authors) that the PADRE/quokka combination is quite usable and reasonably effective. Comparison with other systems on these tasks does not seem to be terribly fruitful given that there is no control over individual (human) differences. Future work is likely to focus on identifying sources of delay or inefficiency which might be eliminated.

5. The speed-up and improvement in "bang-per-buck" of the ACSys VLC runs compared to TREC-6 was very pleasing. It remains to be seen whether the loss of effectiveness relative to last year can be reversed without sacrificing the speed gains. It was pleasing to achieve our VLC goal, particularly when, at times, it seemed so far away.

Acknowledgements

We are indebted to Steve Walker of the Okapi group for his cooperation in providing queries, additional results and information about methods. This help was invaluable in analysing the performance of the ACSys runs.

Bibliography

Cormack, G., Palmer, C., To, S., and Clarke, C. 1997. Passage-based refinement: Multitext experiments for TREC-6. In E. M. Voorhees and D. K. Harman Eds., *Proceedings of the Sixth Text Retrieval Conference (TREC-6)* (Gaithersburg MD, November 1997), pp. 303–320. U.S. National Institute of Standards and Technology. NIST special publication 500-240.

Hawking, D. and Thistlewaite, P. 1995. Proximity operators - so near and yet so far. In D. K. Harman Ed., *Proceedings of the Fourth Text Retrieval Conference (TREC-4)* (Gaithersburg MD, November 1995), pp. 131–143. U.S. National Institute of Standards and Technology. NIST special publication 500-236.

Hawking, D. and Thistlewaite, P. 1996. Relevance weighting using distance between term occurrences. Technical Report TR-CS-96-08, Department of Computer Science, The Australian National University, `http://cs.anu.edu.au/techreports/1996/index.html`.

Hawking, D., Thistlewaite, P., and Craswell, N. 1997. ANU/ACSys TREC-6 experiments. In E. M. Voorhees and D. K. Harman Eds., *Proceedings of the Sixth Text Retrieval Conference (TREC-6)* (Gaithersburg MD, November 1997), pp. 275–290. U.S. National Institute of Standards and Technology. NIST special publication 500-240.

ROBERTSON, S. E., WALKER, S., HANCOCK-BEAULIEU, M., AND GATFORD, M. 1994. Okapi at TREC-3. In D. K. HARMAN Ed., *Proceedings of the Third Text Retrieval Conference (TREC-3)* (Gaithersburg MD, November 1994). U.S. National Institute of Standards and Technology. NIST special publication 500-225.

SINGHAL, A., SALTON, G., MITRA, M., AND BUCKLEY, C. 1995. Document length normalization. Technical Report TR95-1529, Department of Computer Science, Cornell University, Ithaca NY.

VOORHEES, E. M. AND HARMAN, D. K. Eds. 1997. *Proceedings of the Sixth Text Retrieval Conference (TREC-6)* (Gaithersburg MD, November 1997). U.S. National Institute of Standards and Technology. NIST special publication 500-240.

Applying SIGMA to the TREC-7 Filtering Track

Grigoris J. Karakoulas

Global Analytics Group
Canadian Imperial Bank of Commerce
161 Bay St., BCE-11, P.O. Box 500,
Toronto ON, Canada M5J 2S8
karakoul@cibc.ca

Innes A. Ferguson

Active On-line Systems Ltd.
13 Grange Park, Flat 4
London W5 3PL
UK
innes@active-online.net

Abstract

This paper presents the research method and results from applying SIGMA (System of Information Gathering Market-based Agents) to the adaptive filtering task of the TREC-7 Filtering Track. Our work in SIGMA is based on the research hypothesis that a multi-agent learning approach, where each agent learns a local model for information filtering, performs better than a single-agent learning approach that approximates the whole input space with a single model. We report on experiments for testing this hypothesis as well as for constructing the feature space model for this task. This has been SIGMA's first application to TREC experiments and to an IF task over a large document collection. The system was able to scale to the complexity of the task. The performance of SIGMA in TREC-7 is encouraging for further work.

1. Introduction

There has been growing interest within the Information Retrieval and Machine Learning communities for developing methods that combine retrieval results from different learning models for text categorization and routing (see for example Belkin et al., 1993; Lewis et al., 1996; Schapire et al., 1998; Vogt et al., 1998). Such work is supported by empirical evidence showing that for the same information need different users may retrieve sets of documents with little overlap amongst them (McGill et al., 1979). In our work, we have developed SIGMA (System of Information Gathering Market-based Agents) for multi-agent incremental learning and applied it to various information filtering (IF) tasks (Karakoulas & Ferguson, 1995;1996a,b). Due to the multi-dimensionality, partial observability and non-stationarity of an IF task, a multi-agent learning approach, where each agent learns a local model, should perform better than a single-agent learning approach that approximates the whole input space with a single model.

Our interest in participating in TREC-7 (our first TREC participation) and particularly in the adaptive filtering task of the Filtering Track is for two reasons: (i) to evaluate the hypothesis that multi-agent learning with SIGMA improves IF performance over a single-agent learning model; and (ii) to test the scalability and robustness of our IF system for this complex task. In the next section we present the method for performing adaptive information filtering with SIGMA. We then describe the experiments for the Filtering Track and present some of the results. In the last section of the paper we discuss future work.

2. Adaptive Information Filtering with SIGMA

Similarly to economic markets, the computational economy of SIGMA can be defined in terms of goods and agents that produce and consume those goods. The *goods* traded in SIGMA are infor-

mation items (i.e. AP news articles) in different representation forms depending on the stage of processing. The agents are of three general categories: (i) the *consumer* agents, (ii) the *producer* agents and (iii) the *broker* agents.

A consumer, Profile Selector (PS), represents a user's goal and preferences for a topic (i.e. one of the 50 TREC-7 topics) over a time period. Producers, Feature Extractors (FE) and Profile Generators (PG), transform goods from an input form into an output form according to their learning techniques. In response to a consumer's demand for goods the producers enter the local market and compete with each other to serve as efficiently as possible the demands for goods from other agents — consumers or other producers — within the market. A producer determines the price that maximizes its profit by learning how to produce goods, that a consumer is expected to buy, and estimating their demand. A broker is responsible for implementing the bidding policy for a particular consumer's demand for goods, namely setting up the auction each time a demand is posed in the market and deciding which producers win the bidding. The broker also maintains the history of the performance of the producers in serving the particular consumer's demand. The producers that succeed in the bidding sell goods to the consumer at their respective prices. The goods are of different quality. The consumer is endowed with a budget. At any time the consumer is allowed to "shop around" by probabilistically assigning its preferences among the producers and allocating its budget for buying one or more goods. Given its choices for goods these stochastic preferences are reinforced by the feedback that the consumer receives from the environment. The feedback for each good is also propagated by the consumer to the producer from which the product was bought. The market agents used in this task were built as follows.

FE Agent. It transforms an article or a topic according to the Vector Space Model (VSM). The outputs of an FE are the VSM representations of articles and document frequency (DF) tables. The VSM representation currently consists of terms only. Three different term weighting schemes were implemented: (i) simple tf-idf (ii) Salton & Buckley's (1987) tf-idf and (iii) a version of SMART similar to the one by AT&T in TREC-6 (Singhal, 1997). More specifically, (iii) is defined as:

$$A = (1 + \log(tf))/(1 + \log(average(tf_d))) \tag{1}$$

where tf is the term frequency in document d;

$$B = \log((N + 1)/(df)) \tag{2}$$

$$C = 1/[0.8 + 0.2 \times (\text{no. of unique words in d / avg no. of unique words per doc})]. \tag{3}$$

and from (1), (2) and (3)

$$w(t, d) = A \times B \times C \tag{4}$$

In the case of a topic w(t,d) = A.

PG Agent. Upon creation it is initialized with the profile from the topic query. It retrieves VSM articles from an FE agent and determines which articles to purchase using the normalized cosine between the article and profile vectors as the similarity measure together with a similarity threshold. When the PG receives relevance feedback it updates its profile through a modified Rocchio formula:

$$P_{t+1} = \begin{cases} P_t + \beta \vec{D_j} & \text{if } D_j \text{ relevant} \\ P_t - \gamma \vec{D_j} & \text{if } D_j \text{ non-relevant} \end{cases} \quad (5)$$

where $P=[w_1,w_2,...,w_N]$. Only positive weights are allowed. New terms from relevant documents are added to the profile by multiplying their weights with a learning rate δ. The length of the profile, N, is fixed. Terms with relatively low weights can be replaced by terms with higher weights.

PS Agent. The decision task of a PS agent at each time *t* amounts to buying from the set of the PG agents that have won the bidding at time *t*, articles such that the cumulative rewards from buying articles in the long term is maximized. The rewards are the F1 and F3 utility measures based on the relevance judgements. Through such feedback the PS agent learns a policy for which PG agents to buy articles from (for more details see (Karakoulas & Ferguson, 1996b)).

3. Experimentation and Results

Each AP document was preprocessed by filtering words using a stopword list and stemming them. In addition, terms with single occurrency were periodically removed from the frequency tables so that noise due to spelling errors is reduced. Each of the topic descriptions was automatically preprocessed similarly to AP documents for forming the initial query to the system. Words in the concepts section preceded by NOT were removed. To eliminate noise we experimented with forming the topic queries using either limited number of keywords (10, 20 or 30) or specific sections of the topic descriptions. Queries based on the title and concepts gave the best performance by training on the data of 1988/02-1989/04 and testing on the data of 1989/05-1989/12. This data split was used in all the experiments reported in this section. For the performance evaluations we used the F1 and F3 TREC-7 utility measures as well as the geometric mean between precision and recall.

Our experiments can be categorized into two classes: (a) feature space modeling and (b) learning. In the first class we addressed the following questions:

(i) does performance improve by limiting the number of terms represented in each document;

(ii) does the addition of more terms from relevant documents to the initial query improve performance;

(iii) how critical is the choice of the specific method for modelling the feature space and creating the VSM representation of a document.

For (i) we experimented with 20, 50 and 100 terms as the document length. The second option gave the best results. In (ii) we found that increasing the initial query length by 20 terms through relevance feedback gave the best performance. For (iii) we compared the three weighting schemes that were incorporated in the FE agent as described in the previous section, i.e. simple tf-idf, Salton&Buckley's tf-idf and a version of SMART. The latter method gave the best performance.

For the second class of experiments we addressed the following two questions:

(iv) how does SIGMA compare with a system that has no learning capabilities and only selects documents based on the initial query

(v) since SIGMA has a two-level learning mechanism how much does SIGMA improve performance over a system that only uses our version of Rocchio's formula to learn a topic profile

For (iv) we developed an IF system that selects documents by first estimating the normalized cosine for the similarity of the document with the initial query profile and then predicting the document to be relevant if its similarity exceeds a predefined threshold. The performance of SIGMA on the testing data over all topics was on average four times better than the performance of this system that used no relevance feedback. For (v) we used SIGMA with only one PG agent. Learning only takes place in the PG agent through our version of Rocchio's formula and relevance feedback. The learning factors in (5) were set as follows: β=0.9, γ=0.1 and δ=0.2. The performance of SIGMA on the testing data over all topics was on average 85% better than the performance of this single agent with one level of learning.

SIGMA's performance with respect to the other participants in the adaptive filtering task of TREC-7 is shown in Tables 1 and 2, one for each utility run. The tables were constructed using the evaluation results from NIST. For each topic we compared the performance of SIGMA with respect to the minimum, median and maximum performance on that topic. The tables show the averages over all topics of the number of times SIGMA's performance ranked equal to the minimum (=min), less than the median (<=median), greater than the median (>median) and equal to the maximum (=max) peformance. SIGMA mostly ranked lower than the median performance in the F1 run and above the median in the F3 run. The reason for this difference in performance between the two measures is because the F3 utility measure is more biased towards recall than the F1 measure. Due to time constraints we did not tune SIGMA for precision which has been the goal of TREC.

	=min	<=median	>median	=max
1988	0.42	0.5	0.08	0
1989	0.56	0.36	0.08	0
1990	0.62	0.34	0.04	0

Table 1: Comparison of SIGMA for the F1 run.

	=min	<=median	>median	=max
1988	0.36	0.48	0.16	0
1989	0.46	0.38	0.16	0
1990	0.46	0.44	0.08	0.02

Table 2: Comparison of SIGMA for the F3 run.

4. Discussion

This has been our first participation in TREC. We were able to perform various experiments for empirically validating research hypotheses and successfully carry out the adaptive filtering task on the AP collection. This has been partly due to the scalability and robustness of SIGMA with respect to the dimensionality and complexity of the task. According to our evaluation, the performance of SIGMA ranked below but close to the median level compared to the rest of the participants in the adaptive filtering task. Given the time constraints that we faced the results are encouraging for our research in SIGMA. The comparisons with two benchmark IF systems have shown that the two-level learning approach of SIGMA can substantially boost the performance of an one-level learning approach.

There is work underway for extending the feature extraction capabilities of the FE agents with the extraction of phrases. Future work will also focus on a method for dynamically learning the similarity threshold in PG agents.

References

Belkin, N.J.; Cool, C.; Croft, W.B.; and Callan, J.P. The effect of multiple query representations on information retrieval performance. In *Proceedings of the Sixteenth International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-93)*. Association for Computing Machinery.

Karakoulas, G.; and Ferguson, I. 1995. A computational market for information navigation in multi-dimensional spaces. In *Proceedings of AAAI 95 Fall Symposium on AI Application in Knowledge Navigation and Retrieval*, AAAI Press.

Karakoulas, G.; and Ferguson, I. 1996a. SIGMA: Integrating learning techniques in computational markets for information filtering. In *Proceedings of AAAI 96 Spring Symposium on Machine Learning and Information Access*, AAAI Press.

Karakoulas, G.; and Ferguson, I. 1996b. A computational market model for multi-agent learning. In *Proceedings of AAAI 96 Fall Symposium on Learning Complex Behaviors in Adaptive Intelligent Systems*, AAAI Press.

Lewis, D.; Schapire, R.; Callan, J.; Papka, R. 1996. Training algorithms for linear text classifiers. In *Proceedings of the Nineteenth International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-96)*. Association for Computing Machinery.

McGill, M.; Koll, M.; and Norreault, T. 1979. An evaluation of factors affecting document ranking by information retrieval systems. Syracuse University School of Information Studies.

Salton, G.; and Buckley, C. 1987. Term weighting approaches in automated text retrieval. Technical Report 87-88, Cornell University, Department of Computer Science.

Schapire, R.; Singer, Y.; and Singhal, A. 1998. Boosting and Rocchio applied to text filtering. In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-98)*. Association for Computing Machinery.

Singhal, A. 1998. AT&T at TREC-6. In *Proceedings of Sixth Text REtrieval Conference (TREC-6)*, NIST.

Vogt, C.; and Cottrell, G. 1998. Fusion via linear combination for the routing problem. In *Proceedings of Sixth Text REtrieval Conference (TREC-6)*, NIST.

Experiments in Spoken Document Retrieval at CMU

M. Siegler, A. Berger, M. Witbrock, A. Hauptmann*
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

*Justsystem Pittsburgh Research Center
4616 Henry St.
Pittsburgh, PA 15213

Abstract

We describe our submission to the TREC-7 Spoken Document Retrieval (SDR) track and the speech recognition and information retrieval engines. We present SDR evaluation results and a brief analysis. A few developments are also described in greater detail including:

- A new, probabilistic retrieval engine based on language models.
- A new, TFIDF-based weighting function that incorporates word error probability.
- The use of a simple confidence estimate for word probability based on speech recognition lattices.

Although improvements over a development test set were promising, the new techniques failed to yield significant gains in the evaluation test set.

1. The SDR Data and Task

The entire set of speech data for the 1998 TREC-7 spoken document retrieval track consisted of 153 hours of broadcast news, approximately 80 for training and 73 for testing. The data had been segmented into stories and manually transcribed. In the test set, there were three "versions" of the data available: A manually generated transcript, speech recognition transcripts based on IBM and CMU recognizers, and the raw audio data, to be transcribed by our own recognizer.

The entire training set was used to train acoustic models for the speech recognition system. The remainder was held out as unseen test data. There were about 3245 stories in the training data set and 2866 in the test set. To develop and debug the system, the TREC-6 evaluation set was used in a *Known-Item Retrieval* system -- where every query has only one document assigned as relevant.

In our experiments on the evaluation test set, the average precision of the retrieval for each of the relevant documents was used to judge the quality of the retrieval. However, since relevance judgements were not available for the development test set, we used the *Average Inverse Rank* from last year's evaluation to judge retrieval quality.

2. System Overview

In this section we give a system description of the actual CMU TREC-7 SDR submission. The speech recognition system is outlined as well as a fully automatic information retrieval weighting scheme suitable for retrieving documents imperfectly transcribed by automatic speech recognition.

The Speech Recognition Component

The Sphinx-III speech recognition system used for this evaluation was configured similarly to that used in the 1997 TREC-7 SDR evaluation [1], although several changes have been made since then. Sphinx-III is a large vocabulary, speaker independent, fully continuous hidden Markov model speech recognizer with separately trained acoustic, language and lexical models.

For the current evaluation a gender-independent HMM with 6000 senonically-tied states and 16 diagonal-covariance Gaussian mixtures was trained on the TREC-7 SDR training set.

The decoder used a Katz-smoothed trigram language model trained on the 1992-1996 Broadcast News Language Modeling (BN LM) corpus and the LDC-provided supplemental newswire (NW) data from 1997-1998. This is a fairly standard language model, much like those that have been used in the DARPA speech recognition community for the past several years. The lexicon was chosen from the most common words in this corpus. For this evaluation, the vocabulary was comprised of the most frequent 64k words in the BN LM + NW corpora.

The Information Retrieval Component

Both documents and queries were processed using the same conditioning tools, namely noise filtering, stopword removal, and term stemming:

- Noise Filtering: The goal of noise filtering was simply to remove non-alphabet ASCII characters, punctuation, and other junk considered irrelevant to IR. All punctuation was removed except for spelled-letter words, e.g. "C.M.U," and the use of the apostrophe for contractions, e.g. "CAN'T." Any changes in case were removed.
- Stopword removal: A set of 811 stopwords was compiled from a combination of the SMART IR engine and several selected by hand based on document frequency. These words were removed entirely.
- Term mappings: A set of 4578 mappings was used to map words with irregular word endings that were not properly covered by an implementation of the Porter algorithm. An on-line Houghton-Mifflin dictionary was used for this lookup of irregular words and their roots.
  An example of this mapping is `APPENDICES→APPENDIX`
- Term stemming: An implementation of the Porter algorithm was applied to map words to their common root.

For this evaluation, we had two different relevance weighting schemes using entirely different approaches. The first was a vector-space model built on the LNU weighting scheme [3], whereas the second used a language model approach to estimate likelihoods.

3. Word Probability In The Relevance Equation: Mutual Information

Some combination of the two factors frequency and selectivity that is used to evaluate the relevance of documents to queries. Many retrieval engines use derivatives of Salton's vector space model, specifically a measure commonly known as TFIDF (Term Frequency by (log) Inverse Document Frequency.)

Given a set of M documents, a word w_i, and a specific document D_m, the IDF is defined as:

$$IDF_i \equiv -\log\left(\frac{\left|\{\forall m \text{ s.t. } w_i \in D_m\}\right|}{M}\right)$$

Although it is obvious that the IDF provides some measure of term selectivity, it is important, for its application in this paper, to derive a theoretical basis for its use. If documents and queries are regarded from a probabilistic point of view, the significance of IDF is readily apparent and motivates the proper use of word probabilities derived from the speech recognition.

Let documents and queries be defined as mappings of words into probabilities:

$$D : w_i \rightarrow P(w_i)$$
$$Q : w_i \rightarrow P(w_i)$$

The space of distinct documents is defined as:

$$\boldsymbol{D} \equiv \{D_1, D_2, \cdots, D_M\}$$

The *a-priori* probabilities of document relevance are equal:

$$P(D_m) = 1/M$$

The probability of a document, given a particular word is:

$$P(D_m \mid w_i) = P(w_i \mid D_m)\frac{P(D_m)}{P(w_i)}$$

And by simple expansion:

$$P(D_m \mid w_i) = \frac{P(w_i \mid D_m)P(D_m)}{\sum_{m'=1}^{M} P(w_i \mid D_{m'})P(D_{m'})}$$

Consider the information content of word w_i to be the mutual information between the document set and the word:

$$I(\boldsymbol{D}; w_i) \equiv H(\boldsymbol{D}) - H(\boldsymbol{D} \mid w_i)$$

Expanding, using the definition of entropy:

$$I(\boldsymbol{D}; w_i) = -\sum_{m=1}^{M} P(D_m)\log_2 P(D_m) + \sum_{m=1}^{M} P(D_m|w_i)\log_2 P(D_m|w_i)$$

The relevance of query Q to document D_m in space $\boldsymbol{D}$ is defined as the expected value of this information content:

$$\mathrm{Rel}(Q, D_m|\boldsymbol{D}) = \sum_{i=i}^{N} E\{I(\boldsymbol{D}; w_i)|Q, D_m)\}$$
$$= \sum_{i=1}^{N} P(w_i|Q, D_m) I(\boldsymbol{D}; w_i)$$

Assuming documents and queries to be independent:

$$\mathrm{Rel}(Q, D_m|\boldsymbol{D}) = \sum_{i=1}^{N} P(w_i|Q)P(w_i|D_m) I(\boldsymbol{D}; w_i) \qquad (1)$$

If, following the usual practice, documents and queries map words to indicator functions (Boolean):

$$I(\boldsymbol{D}; w_i) = \log_2(M) - \sum_{i=1}^{N} 1(w_i|D_m) = \mathrm{idf}(w_i|\boldsymbol{D})$$

And the relevance function reduces to the familiar:

$$\mathrm{Rel}(Q, D_m|\boldsymbol{D}) = \sum_{i=1}^{N} 1(w_i|D_m)1(w_i|Q)\mathrm{idf}(w_i|\boldsymbol{D})$$

Where the indicator functions are the TF values. By this logic, it is seen that the IDF can be supported as a meaningful derivative of information content. In addition, the more general form in Equation 1 can be used when word probabilities are available.

4. Estimating Word Probability From Recognition Lattices

In the course of decoding the incoming speech signal into a word string, the Sphinx III recognizer produces a lattice of words representing the many competing hypotheses. Each hypothesized word in the lattice has a starting time, an ending time, a link to possible following words, and model probabilities for the word. After producing this lattice, the recognizer selects the most probable path after weighing evidence from the different modeling sources available. The best path, also called the *top-1* hypothesis, is generated as the output of the recognizer.

Although the lattice is available, only the best path has typically been used for the purpose of information retrieval. Although the lattice does not contain all possible word sequences, it is a far more detailed representation of what may have been said than can be given in a single transcription. One serendipitous benefit of the lattice is that the presence of a large number of options at any moment in time may indicate an uncertainty in word recognition. This is valuable, since it should be beneficial to predict which words in the top-1 hypothesis are incorrect, and discount them during information retrieval.

One way of measuring the number of competing hypotheses for a specific node in a lattice is the following:

- Count the time span (in frames) of the node: N
- Count the number of frames contained in other nodes that occur simultaneously with this node (partially or completely): M
- The Lattice Occupation Density (LOD) is N/(N+M).

In the example shown in Figure 1, the recognition system is less certain about the presence of "today" than "news" because the latter word has no competing hypotheses.

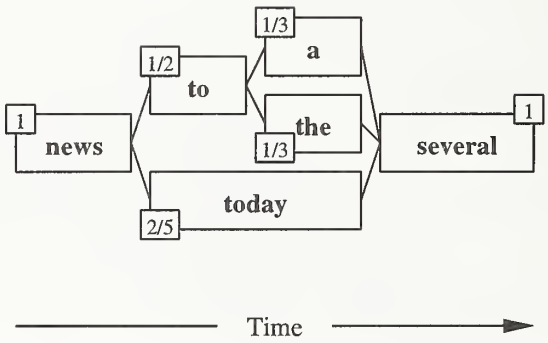


Figure 1: A simple lattice. Numbers show the
Lattice Occupation Density (LOD) values for the various nodes

The TREC-6 training corpus was used to build a probability model by analyzing the lattices created during recognition. The LOD values for each word in the top-1 hypotheses were collected, and the word errors tallied. In Figure 2, the probability that a hypothesized word occurred in the reference transcript is compared with its LOD value from the lattice. From these measurements of the training set, a parameterized model of word probability was derived. The model adopted for this paper was a best fitting exponential of the form:

$$P(w \mid LOD) \approx 1 - 0.2^{LOD}$$

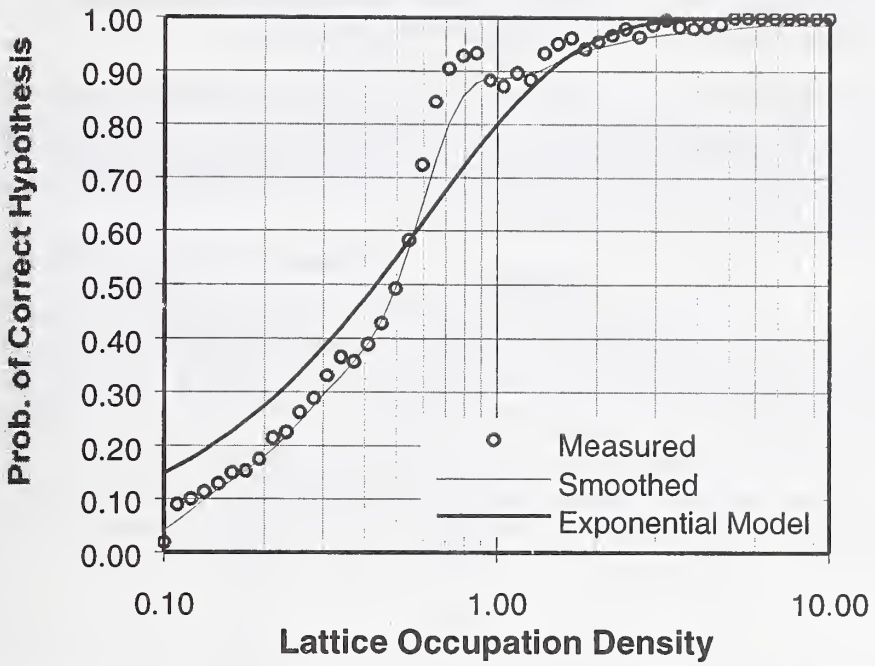


Figure 2: Using LOD to predict word probability in the top-1 hypothesis. For example, hypothesized words with an LOD of 1.0 appeared in the reference text approximately 85% of time.

Performance of Mutual-Information Metric On Development Test Set

As can be seen in Table 1, using error prediction in conjunction with the mutual information metric can reduce the average inverse rank difference between reference texts and the top1 hypotheses by 25%.

Transcription Source	Average Inverse Rank	
	LNU	LNU+Mutual Inf.
REF	0.82	
TOP1	0.74	0.76

Table 1: Performance of the mutual information metric on the development test set. Values are average inverse ranks for the different transcription sources.

5. Finding the Best IR Lattice Path

Although improving the word error rate of the transcription is the primary goal for speech recognition tasks, in the case of indexing and searching audio material, some consideration for the application of the transcription should be made. Since many types of "errors" in transcription are benign (for example a substitution of one stop-word for another) it is apparent that reducing the word error rate blindly does not necessarily result in improved retrieval performance.

In order to clarify this point, the lattices from the speech recognition phase of the evaluation test set were scanned for their lowest word-error paths. This is commonly known as the *Oracle Word Error Rate* of the lattice. In addition to these paths, the paths that generate the lowest word error after the text-processing steps described earlier, *with respect to the text-processed references*, were generated. The distinction of the latter is that paths containing benign errors are not penalized.

Table 2 shows the performance of the oracle path selection using the mutual information retrieval engine on a 2601 document *subset* of the evaluation set. Approximately 10% of the original 2866 documents failed to produce lattices, or were too long to rescore, and these were deleted from the test set.

Filtering after the oracle path is found is tantamount to finding the path that generates the oracle word error rate of the lattice. As can be seen, this is not the optimal path through the lattice with respect to information retrieval. However, filtering before the optimal path is found is a better representative of an error criterion commensurate with the retrieval operation. What is most surprising is that the original performance on the reference transcripts can be completely recovered from the speech-recognition lattices.

	Baseline		Oracle	
	Reference Transcripts	Speech Transcripts	Filter Before Oracle	Filter After Oracle
Training Set	AIR=0.79	AIR=0.75	AIR=0.79	AIR=0.75
Testing Set	P_{AVG}=0.39	P_{AVG} =0.36	P_{AVG} =0.39	P_{AVG} =0.37

Table 2: Baseline and Oracle Annotation on TREC-6 Training and Testing Sets.
Values are Average Inverse Rank (AIR) for the training set, and the average precision (P_{AVG}) for the testing set.

6. Information Retrieval Using Language Models

We also experimented with a variation of the *Language Modeling Approach* to retrieval introduced recently by Ponte and Croft [4]. The idea behind the LM approach is to estimate a separate language model for every document in a collection; fielding queries involves discovering the documents whose models most closely accord with the query.

Specifically, the LM approach associates with each document d a probability distribution $p(t \mid d)$ over terms t. The probability of a query $q = \{q_1, q_2, \cdots, q_n\}$ is given as a product of individual term probabilities,

$$p(q \mid d) = \prod_{i=1}^{n} p(q_i \mid d)$$

Given a query q and a collection of documents d, the LM approach ranks documents by how well they "predict" the query---in other words, by $p(q \mid d)$. Of course, assuming a uniform prior $p(d)$ on documents, ranking documents by $p(q \mid d)$ is equivalent to ranking them by $p(d \mid q)$, which is perhaps a more natural perspective.

Estimating the parameters of these document-specific models is a delicate business. Maximum likelihood estimation, where the probability for a term is proportional to the number of times the term appeared in the document, will not work, since documents are often far too small for robust model estimation. Ponte and Croft's proposed solution is to interpolate or "smooth" each document-specific model with a model estimated on the entire collection of documents.

We adopted a different approach to language model construction. We use a penalized maximum likelihood objective function to select, for each document, the optimal model within a family of exponential models. This approach is more fully described in [5].

Table 3 shows the performance of the language model based retrieval engine on the development test set. In general, the performance is on par with the more finely tuned mutual information retrieval engine. Work is progressing at CMU on a more sophisticated approach which can better handle synonymy and polysemy.

Transcription Source	Average Inverse Rank
REF	0.80
TOP1	0.76

Table 3: Performance of the language model based metric on the development test set. Values are average inverse ranks for the different transcription sources.

7. Official TREC-6 SDR Results

Table 4 shows the official CMU TREC SDR results. Unfortunately, the performance of the mutual information retrieval engine, coupled with the probability estimator did not fare as well as hoped. Upon investigation, the model using the lattice occupation density turned out to be a poor estimate of the word probability for the evaluation test set. However, the language model retrieval engine fared well in either case.

Transcription Source	LNU metric P_{AVG}	LM metric P_{AVG}
REF	0.36	0.39
B1	0.33	0.35
B2	0.26	0.27
CMU	0.32	N/A
CMU+Lattice	0.29	N/A

Table 4: Performance of the CMU TREC-7 SDR Evaluation System

References

[1] M. Siegler, M. Witbrock, S. Slattery, K. Seymore, R. Jones, A. Hauptmann, "Experiments in Spoken Document Retrieval at CMU," *Proceedings of TREC-6*, November 1998, Gaithersburg, MD.

[2] M. Siegler, M. Witbrock, "Improving The Suitability Of Imperfect Transcriptions For Information Retrieval From Spoken Documents," *ICASSP 99* , March 1999, in press.

[3] A. Singhal, C. Buckley, M. Mitra, "Pivoted Document Length Normalization," *SIGIR-1996*, Zurich, Switzerland, August 1996.

[4] J. Ponte, W. Croft, "A Language Modeling Approach to Information Retrieval," *SIGIR-1998*, 1998.

[5] A. Berger, R. Miller, "Just in Time Language Modeling," *ICASSP-1998,* May 1998.

Acknowledgments

This research was supported in part by DARPA under research contract F33615-93-1-1330 and N00039-91-C-0158. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA or the U. S. Government.

PLIERS AT VLC2

*A. MacFarlane, *+S.E.Robertson, *J.A.McCann
*School of Informatics, City University, London EC1V OHB
+Microsoft Research Ltd, Cambridge CB2 3NH

ABSTRACT: This paper describes experiments done on the VLC2 collection at TREC-7. Methods used for indexing text is described together with the results: this includes the official collections BASE1, plus some larger unofficial collections named BASE2 and BASE4. Search times on these collections are described and discussed with a particular emphasis on scaleup: for both weighted term search and passage retrieval. The various configurations for experiments are described.

1. INTRODUCTION

This paper is a description of results gained using the PLIERS system on baselines of the VLC2 collection at TREC-7. The research is part of an ongoing effort to study the effects of different partitioning methods for Inverted files in parallel IR systems. In particular we wish to find out which partitioning method yields the best Indexing and Search results with respect to elapsed time as seen by the user. We present the official results for BASE1 only of VLC2, but include some further results from experiments on larger but unofficial collections labelled BASE2 and BASE4, which are approximately 2 and 4 Gigabytes in size respectively. In section 2 we give details of the experiments such as the hardware and software used. The Indexing results are described in section 3 while section 4 describes the results gained on ordinary term weighting search and section 5 describes results gained using passage retrieval search methods. An overview of the results is provided in section 6. A summary is given in section 7.

2. DESCRIPTION OF THE EXPERIMENTS

In this section we describe various aspects of the experiments, such as software and hardware used, description of the data used, query processing details, the retrieval model used in search and measures used in the experiments.

2.1 SOFTWARE USED

PLIERS (ParaLLeL Information rEtrieval Research System) is a parallel text retrieval system currently being developed at City University as part of the first author's PhD studies. The concepts and design used in the development of this software are heavily influenced by the Okapi system [1]. The message passing system namely MPICH [2] is used for data exchange: this system supports the MPI (Message Passing Interface) standard.

2.2 HARDWARE USED

PLIERS is designed to run on several parallel architectures and is currently implemented on those which use Sun Sparc and DEC Alpha processors. All results presented in this paper were obtained on an 8 node Alpha farm at the Australian National University, Canberra. Each node has its own local disk: the Shared Nothing Architecture [3] is used by PLIERS. Each node is a series 600 266Mhz Digital Alpha workstation with 128 Mbytes of memory running the Digital UNIX 4.0b operating system. One of the nodes has a RAID disk array attached to it and other nodes can access the RAID using NFS. Two types of network interconnects were used: a 155 Mb/s ATM LAN with a Digital GIGASwitch and a 10 Mb/s Ethernet LAN. Search requests were submitted on both types of Networks, but indexing was only done on ATM. Figure 1 shows the architecture of the Alpha farm.

2.3 DATA DESCRIPTION

We use a number of collections in our experiments: BASE1, BASE2 and BASE4. BASE1 is an officially defined sample of the 100 Gigabyte VLC2 collection [4] and is 1 Gigabyte in size. The BASE2 and BASE4 collections are subsets of the official BASE10 collection (another sample of VLC2) and were created by varying the number of BASE10 compressed text files put through the indexing mechanism (130 files per node for BASE2 and 260 for BASE4). The BASE2 collection is 2.1 Gigabytes in size, while the BASE4 collection is 4.2 Gigabytes in size.

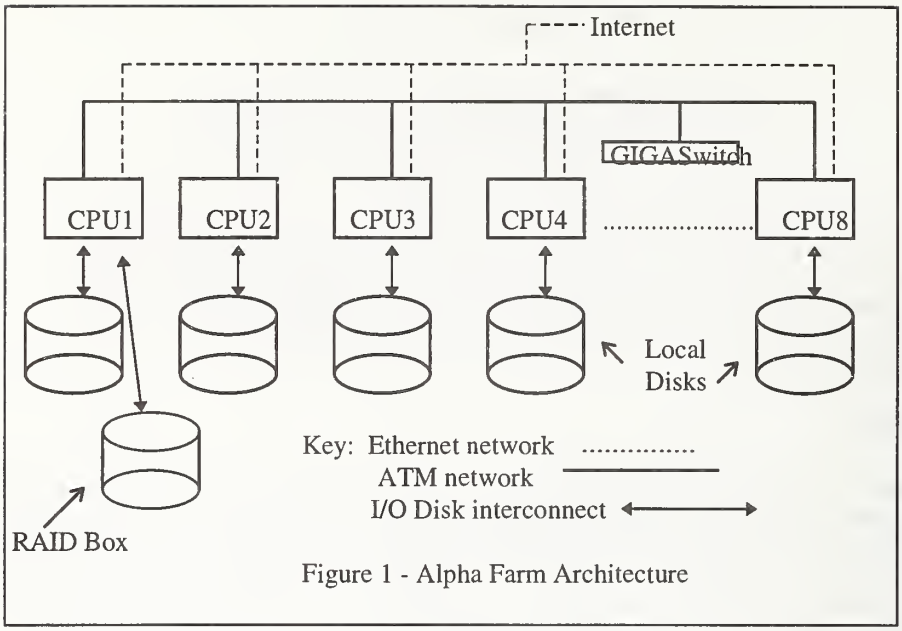


Figure 1 - Alpha Farm Architecture

The strategy used to distribute the BASE1 and BASE10 collections across the nodes was to evenly spread the compressed text file's directories among them as far as possible (an alternative if more time consuming strategy is to do it by file size). The requirement of a distribution strategy is to get the best possible load balance for Indexing as well as Term Weighting and Passage Retrieval search. The distribution was done before the Indexing program was started, and is not included in the timings.

Inverted Files were used with a Dictionary File that contains: keywords, the number of occurrences in the collection of that keyword and a pointer to an Inverted list in the Postings File. Each element of the Postings file contains: a posting with a document identifier, the number of occurrences of a word in that document, plus a list of positions for a word in that document (if such processing is required). Each position element contains field, paragraph, sentence and word position data together with the number of preceding stop words. Only the paragraph data was used in these experiments (for passage retrieval). A Document map file was used to record document length and its start location in the text file as well as the document identifier.

Two types of Inverted files were used for experiments: one type that recorded position information (necessary for passage retrieval and adjacency operations if required) and one that recorded postings only in the Inverted list.

2.4 INDEXING METHODOLOGY

The strategy used for indexing was a parallel one using the document identifier partitioning method, keeping all indexing builds local to the processor. We define document identifier partitioning as Inverted file data that is distributed over a number of partitions, each partition having a unique set of document identifiers [5]. We name the method used for parallel indexing with its data distribution strategy "Local Build Document Identifier Indexing". The method reduces communication to the minimum: after the initial distribution of documents, only control messages are sent. Fragments of the Inverted file are distributed across local disks. Each fragment of the whole Index is an Index of the sub-collection held on that node. We do not keep a super dictionary, which would record all keyword and occurrence data in one central location. Figure 2 shows the Indexer topology.

For each local Index build we used a stop word list of 450 words to filter out unwanted terms and was supplied by Fox [6]. HTML tags were stripped from the text and ignored if not used for specific reasons such as identifying paragraphs <p> and the end of document </DOC>. Each identified word was put through a Lovins stemmer, supplied by the University of Melbourne, and indexed in stem form. Numbers were not indexed. A large amount of in-core memory is pre-allocated in blocks by each indexing process, and documents are analysed until one of several criteria is reached: run out of keyword block, posting block or position block space. When one of the criteria is satisfied, the current analysis is dumped to disk as an intermediate Index, so that the in-core memory can be used for the next set of documents. When all documents have been analysed, the intermediate Indexes are merged together to create the final Index and deleted. We attempted to keep keywords in main memory for the duration of the indexing, but found that we ran out of keyword space fairly quickly on the data sizes we used. Therefore intermediate dictionary files had to be created as well.

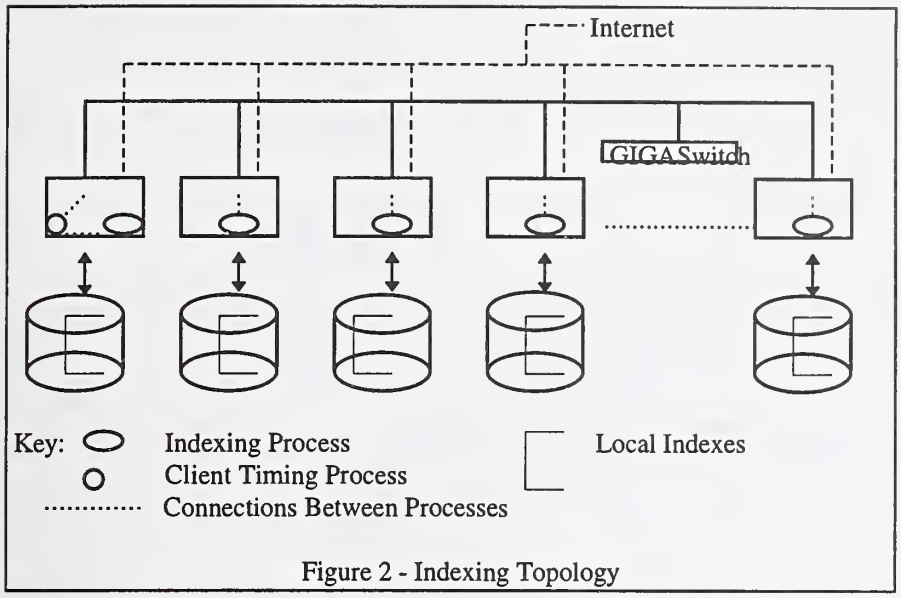


Figure 2 - Indexing Topology

2.5 QUERY PROCESSING

The queries are based on topics 351 to 400 of the TREC-7 ad-hoc track: 50 queries in all. The terms were extracted from TREC-7 topic descriptions using an Okapi query generator utility and put through the Lovins stemmer to produce the final query. The average number of terms per query is 19.58. Queries were processed over the full collections and then subsets (reducing in 1/8th steps to 1/8th of the collections). All queries where submitted sequentially to the system, distributing the queries to all leaf nodes in the system and processing the individual queries in parallel (see figure 3). The batch query client process reads in queries and sends them to the Top Search process. The top search process then sends the query to all leaf processes, awaiting results. The results come back in the inverse direction of the Query. We used three different cases of query processing: term weighting search in the presence and absence of position data in the Inverted file, and Passage Retrieval.

The passage retrieval techniques used here was implemented in Okapi at TREC-3 [7], and is done on arbitrary passages, iterating through contiguous sequences of text atoms in order to find the best weighted passage. The text atoms used were paragraphs. There are two stages in the process. An ordinary term weighting search including a sort identifies the top 1000 top ranked documents in the first stage. The second stage applies the passage retrieval algorithm to the top ranked documents from the first stage. With our method we apply the passage retrieval algorithm to 1000 documents per node, therefore passage retrieval is done on 8000 documents for the full collections, 7000 for 7/8th down to 1000 for 1/8th.

2.6 RETRIEVAL MODEL

The model used for search is the Robertson/Spark Jones Probabilistic model [8] and all Searches, including the ordinary term weighting operation and passage retrieval, use the BM25 weighting function [9]. The tunning constant settings were 2.0 for k1 and 0.6 for B.

2.7 MEASURES USED

For the official results we supply average elapsed times as seen by the user for all query processing cases in search and the total elapsed time for Indexing. For search we give the average elapsed time on the ATM network only as the differences in time with Ethernet elapsed times were found to be small. All search experiments on top of the official submitted results were aimed at measuring scaleup on the chosen method of data partitioning for Inverted files: namely document identifier partitioning. Scaleup is defined as the ability to process a large job in the same time as a smaller job [3]. The ideal scaleup would be 1.0, and any figure higher than this gives the loss due to distribution. Scaleup was measured by obtaining results on the on full BASEx collection and then working backwards getting 7/8th of BASEx to 1/8th of BASEx and comparing the search results on all fragments of the collection. For each query processing case we supply the scaleup from 1 to 8 leaf processes (average elapsed time on 8 leaf processes

divided by average time 1 on leaf process) and comment on the scaleup line (see appendix 1). For Indexing we use a measure of load imbalance LI, defined as being the ratio of the maximum elapsed time divided by the average elapsed time: a perfect load balance would achieve an LI of 1.0 [10].

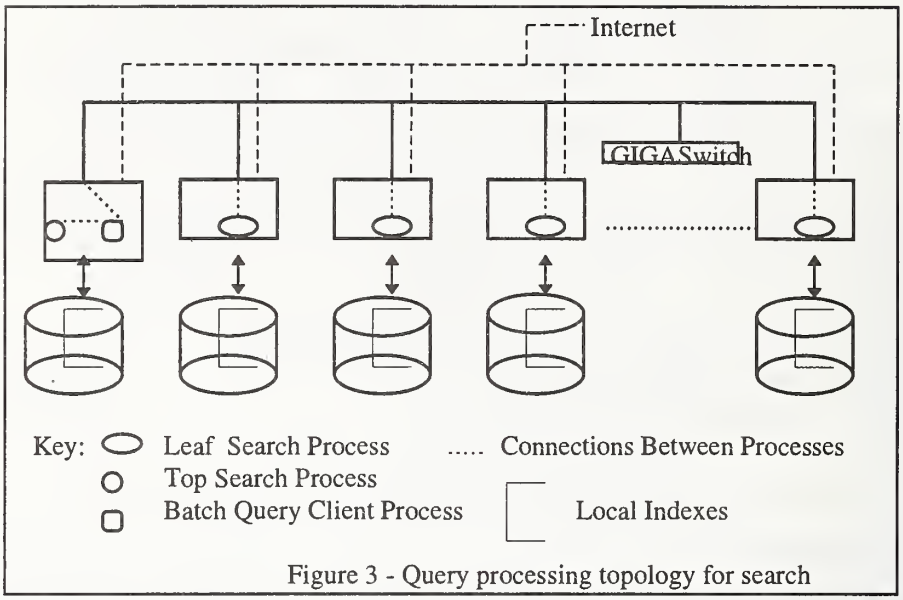


Figure 3 - Query processing topology for search

Inverted file sizes and the overhead they incur with respect to the text are declared. We measure the costs of having position information in the Inverted file for searches that do not require such: this is the query processing case of term weighting search in the presence of position data. It should be noted that document identifier partitioning necessarily incurs a space overhead on the dictionary file as keywords may be on several or all nodes: we do not measure this overhead.

3. INDEXING RESULTS

In this section we give timings for Index building for the BASE1, BASE2 and BASE4 collections and the resultant sizes of the Inverted files for these collections.

3.1 OFFICIAL BASE1 RESULTS

The time taken to index BASE1 was 416 seconds with position data and 286 seconds without position data. The Inverted file with position data was 334 Megabytes in size, while the Inverted file with postings only was 127 Megabytes in size: 35% and 13% of the text respectively. The cost of recording position data in the files was therefore an extra 130 seconds in time and an extra 22% of the text for disk space. The LI for indexing with position data was 1.11, while 1.07 was recorded for indexing without position data: both reasonable measures for load imbalance.

3.2 BASE2 RESULTS

The time to index BASE2 was 5867 seconds with position data and 589 seconds without position data. The Inverted file with position data was 620 Megabytes in size, while the Inverted file with postings only was 275 Megabytes in size. The Inverted file with position data was 29% of the text, while the Inverted file with postings only was 13% of the text. While the extra costs for file space with position data were acceptable for the Inverted files, the extra time needed were not acceptable being nearly 10 times of that required for indexing with postings only. The LI figures reflect this mismatch: an LI of 3.44 was recorded for indexing with position data, while a figure of 1.06 was found for postings only. The reason for this mismatch is discussed in section 3.4.

3.3 BASE4 RESULTS

The results on BASE4 were done on Inversion without position data because of problems with the merging stage of the Indexing: this prevented us from going any further with the experiments (see section 3.4). The time to index the BASE4 collection without position data was 10,044 seconds, with a very poor LI of 3.52: the reason for this is discussed in section 3.4. The Inverted file was 557 Megabytes in size: therefore the Inverted file was 13% of the text file.

3.4 COMPARISON OF INDEXING RESULTS

The times for indexing on BASE1 and BASE2 with no position data were found to be acceptable as was the LI measure, recording figures of just over 1. However the figures for BASE2 with position data and BASE4 with postings only did not meet the required performance. The reason for this is twofold: the merging method used did not scale as times deteriorated with data size and the use of recursion in list processing caused problems by using up the stack space. Elapsed indexing times gradually became worse with the amount of memory needed to process the data and no further progress could be made. The method was unusable beyond BASE4 with postings only data. The official collection BASE10 was not dealt with because of the problems (the full VLC would not have been done because of restrictions on resources). Work has been completed which replaces the merging phase of the indexing in order to make it both scaleable and be able to handle much larger data sets. Iterative techniques for list processing have been introduced to overcome the stack space problem.

Of all the measurements taken with respect to indexing, the Inverted file size cost was found to be good, being around 29/35% when position data is recorded and 13% when postings only are recorded.

4. TERM WEIGHTING SEARCH RESULTS

We supply the results for term weighting search in this section including the retrieval efficiency and effectiveness for the BASE1 collection and retrieval efficiency results for BASE2 and BASE4 collections.

4.1 OFFICIAL BASE1 RESULTS

A. Timings

The average elapsed search time for 50 queries on the full BASE1 collection operating on Inverted files with position data was 1.74 seconds. The scaleup from 1 to 8 leaf processes was found to be 2.6. The data for queries on an Index with postings only delivered a much better average response time with 0.648 seconds. The scaleup curves for Inversion with position data were generally encouraging apart from a sharp rise in elapsed average time from 7 to 8 leaf processes: the same is found on Inversion without position data (see appendix 1, BASE1 timings).

B. Retrieval effectiveness results

The id of the official term weighting run was pliers1-vlc2. The results for this run were very disappointing with only 80 relevant documents retrieved and a precision of 0.08. On investigation it was found that the problems discovered in Indexing caused data corruption in the postings lists. The removal of this data corruption problem improved the precision to 0.126 with the original tuning constant settings (for k1 and B) and this was increased to 0.132 with k1 set as 1.0. This represents an increase of 65% over the original results. A further evaluation on the top 1000 documents on the original tuning constant settings found 386 relevant retrieved documents so there is still much room for improvement, i.e. there are more relevant documents in BASE1.

4.2 BASE2 RESULTS

The average search elapsed time for 50 queries on the full BASE2 collection operating on Inverted files with position data was 3.14 seconds on the ATM network. The scaleup from 1 to 8 leaf processes was 2.04. For searches in the absence of position data the search times on the 50 queries were much better with average elapsed times of 1.11 seconds: scaleup from 1 to 8 leaf processes was 2.92. The scaleup curves for Inversion with position data were generally encouraging apart from a sharp rise in elapsed average time from 7 to 8 leaf processes: the same is found on Inversion without position data (See appendix 2, BASE2 timings).

4.3 BASE4 RESULTS

The search average time for 50 queries on the full BASE4 collection operating on Inverted files with postings only data was 3.14 seconds. Scaleup from 1 to 8 leaf processes was 2.357. The scaleup curves are identical, but there is a difference on 6 leaf processes with a higher figure for the Ethernet results. There is a sharp rise in elapsed average time again from 7 to 8 leaf processes (BASE4 timings, appendix 3).

4.4 COMPARISON OF TERM WEIGHTING SEARCH RESULTS

The average results for the 50 queries on all data sets are in the main good for all types of Inverted file type and network type. Comparing searches on Inverted files with position data we found that the average elapsed time for 50 queries on BASE2 where slightly less than double than those on BASE1. The comparison with postings data only was less favourable: 50 queries on BASE2 was again just over double that of BASE1, but the queries on BASE4 took just under three times longer than on BASE2 and just over five times longer on BASE1. This is clearly a problem with the scalability of the sequential search algorithm. We put this down to a merging failure in the indexing process, since the method used did not guarantee a unique keyword per fragment: therefore increasing the amount of I/O for any given query term. Comparing the overhead in search speed on queries that do not require position data we find that at BASE1/BASE2 search times for searches on Inverted files were 2.7/2.8 times longer than the same queries on files with postings data only. The scaleup curves for both types of networks appear to be generally favourable on all types of Inverted configurations, apart from a sharp rise in elapsed average time on 7 to 8 leaf processes.

5. PASSAGE RETRIEVAL RESULTS

We supply the results for passage retrieval search in this section including the retrieval efficiency for BASE1, BASE2 and BASE4 collections as well as the retrieval effectiveness results for BASE1.

5.1 OFFICIAL BASE1 RESULTS

A. Timings

The average search time for 50 queries on the full BASE1 collection was 3.49 seconds. The scaleup from 1 to 8 leaf processes was found to be 2.09, much better figures than for term weighting search: however the average search times are just under double that of term weighting search. The scaleup curves are broadly encouraging apart from a sharp rise in elapsed average time from 7 to 8 leaf processes (see appendix 1).

B. Retrieval effectiveness results

The id of the official passage retrieval run on BASE1 was pliers2-vlc2. The data corruption problem discussed in section 4.1 above also affected passage retrieval with a precision of 0.056 on the official run with only 56 relevant retrieved. The revised program produced 79 relevant retrieved documents with a precision of 0.079 but the results are still a long way behind the revised term weighting results declared above. The best revised results were found with k1 set as 0.5 yielding a precision of 0.091 (91 relevant retrieved documents). A further evaluation on the top 1000 documents on the original tuning constant settings found 339 relevant retrieved documents. Significant further investigation is merited into why passage retrieval is not performing as well as hoped and giving improved retrieval effectiveness over term weighting retrieval.

5.2 BASE2 RESULTS

The average search time for 50 queries on the full BASE2 collection was 6.17, just under double the time for the average elapsed search time on passage retrieval over the BASE1 collection. The scaleup from 1 to 8 leaf processes was found to be very good indeed with figures of 1.56: by far the best scaleup recorded in these experiments. There is something of a strange effect in the scaleup curve going from 3 to 4 leaf processes in that elapsed time on 4 leaf processes is better than on 3. The effect is found on both ATM and Ethernet networks. The 4 node search is a superset of the 3 node search. For the moment we are at a loss to explain this and the problem is currently under investigation. There is a sharp rise in elapsed average time from 7 to 8 leaf processes on the scaleup curve again (see appendix 2).

5.3 COMPARISON OF PASSAGE RETRIEVAL RESULTS

With respect to passage retrieval for the 50 queries over BASE1 and BASE2, it was found that the average elapsed search time for BASE2 were 77% larger than BASE1 (6.17 seconds for BASE1 compared with 3.49 seconds for BASE2). This is encouraging as BASE2 is just over double the size of BASE1. However, the average results for passage retrieval are expensive compared with term weighting search, with searches on average being slightly less than double with term weighting searches on Indexes with position data and a factor of just over five times that of term weighting searches on Indexes with postings only. The method therefore would only be worthwhile if gains could be obtained in retrieval effectiveness.

6. OVERVIEW OF RESULTS

We draw the threads of both indexing and search together in this section, discussing various aspects found in the experiments including the issues of Load Imbalance, scalability, communications costs, process mapping and a discussion of PLIERS as implemented.

6.1 LOAD IMBALANCE

DATABASE	TEXT	INDEX		TIME	
		Posting Only	*Position Data*	*Posting Only*	*Position Data*
BASE1	1.05	1.09	1.09	1.07	1.11
BASE2	1.02	1.07	1.19	1.06	3.44
BASE4	1.03	1.05	-	3.52	-

Table 1 - Comparison in LI between Times, Full Text and Indexes

From Table 1 it is clear than in the most part, there is not that much different between the LI for actual text size, build time and Index file sizes. The anomalies are found in BASE2 with position data and BASE4 with postings only and these are caused by the list processing problem described below.

6.2 SCALABILITY ISSUES

An important question to ask is how far the PLIERS software would scale to systems that use more than 8 nodes say up to 1000 nodes. With Indexing we feel confident that the method would scale to systems with much larger numbers of processors. The scalability of the Search process is perhaps not so clear cut because of the communication involved which must eventually affect its performance (see section 6.3). However, the Search process should scale to systems beyond 8 leaf processes (see section 6.4 with respect to the sharp rise in elapsed average time from 7 to 8 leaf processes). These statements are made in the knowledge that the scalability of the sequential algorithms utilised leaves something to be desired (see section 6.5 below).

6.3 COMMUNICATION COSTS

Communication costs are kept to a minimum in the techniques described in this paper. With Indexing only a small number of control messages are used and the Indexers can proceed independently without the need for any communication between each other. With Search, only queries and answers are transmitted rather than whole data sets slowing the computation. Since average elapsed times on both Ethernet and ATM networks are similar we have strong evidence that reducing communication as far as possible in parallel IR systems are important. Some extra communication in search is required in order to satisfy the requirements of the term weighting model: as a keyword may reside on several fragments we need to accumulate the term frequency so that accurate term weighting can be done. We treat the fragmented collection has a whole collection, rather than a number of separate sub-collections. The extra communication is the price to be paid for not keeping a super dictionary, recording the term frequency in a central location. Such a technique would reduce communication further, but would require extra storage space.

6.4 PROCESS MAPPING

One of the most important aspects of Parallel Computing is the mapping of processes to processors: what processor should go were and how many processes should we allow on one processor. We wish to ensure that processors are not overloaded, skewing the computation and reducing the load balance. In most of our search

experiments we were able to place the query batch client and top search node on the same processor, but on a different one from the leaves. This was not the case when all 8 nodes for needed for processing, therefore the top search node and batch query client had to be placed on the same processor as a leaf node. These three nodes on one processor competed for CPU and memory resources. This explains in part the sharp rise in elapsed average time from 7 to 8 leaf processes found in all searches on the full collections. It is possible to get by this in MPICH by using a processor of a different type for the query batch client and top search node. This however does require some reprogramming to deal with differences in fundamental types and representations for data. Not all MPI implementations support this facility, however.

6.5 IMPLEMENTATION ISSUES

As can be seen, particularly with some of the indexing results, there is evidence of non-linearity in system behaviour with respect to elapsed time. As discussed earlier it was found that the merging technique used, by not guaranteeing a unique keyword for the dictionary file for any inverted file fragment caused problems in both Indexing and Search. With indexing it required more write I/O's when saving data (1 I/O extra per keyword duplicate in the dictionary file, plus one for the Inverted list), and with growing data sizes the likelihood of duplicates increased thereby increasing times. These extra duplicates impacted on Search at query time by imposing an extra I/O per duplicate in order to read in the inverted lists for a term. A new merging technique has been implemented and is currently being tested which will eliminate the non-linearities found in the experiments. The use of recursion in list processing was found to be problematic requiring the use of stack space in vast quantities. No matter how much stack space is made available, using such a technique would eventually run into problems with increase in data size. The use of iterative techniques in for list processing is therefore not recommended for IR systems that handle very large data sets. This has been a hard if worthwhile lesson to be learned for the first author. Iterative techniques for list processing have now been implemented.

7. SUMMARY

Overall the results presented in this paper are encouraging, but it is important to recognise that further work is needed to improve the performance of individual nodes of both Indexing and Search (both term weighting search and passage retrieval). Indexing speed is very good on BASE1 and BASE2 with no position data, and good load balance figures (LI) are recorded, all just over the 1 mark. It is quite clear from the data that the method used provides useful scaleup on searching with all scaleup curves being very near flat. The ATM and Ethernet curves on search are almost identical on all collections, and we confirm that using the Shared Nothing architecture transmitting results only rather than whole data sets is useful in parallel IR systems. We also infer that the method defined as "Local Build Document Identifier Indexing" is a useful method and data distribution strategy for fragmented Inverted files in parallel IR systems. Clearly retrieval effectiveness could also be improved further, and that significant further investigation is merited into the difference in results between term weighting search and passage retrieval.

8. ACKNOWLEDGEMENTS

This research is funded in part by British Academy Grant No IS96/4203. We are also grateful to ACSYS for awarding the first author a visiting student fellowship at the Australian National University in order to complete this research, in particular to David Hawking for making the arrangements for the visit.

REFERENCES

[1] S. E. Robertson, Overview of the OKAPI projects, Journal of Documentation, Vol 53, No 1, January 1997.

[2] Gropp, W., and Lusk, E., Users guide for MPICH, a portable Implementation of MPI, Mathematics and Computer Science Division, Argonne National Laboratory, University of Chicago, 6 July 1998.

[3] DeWitt, D., *and* Gray. J. Parallel database systems: the future of high performance database systems. *Communications of the ACM*, *35* (6), 1992, 85-98.

[4] VLC Web Page URL http://pastime.anu.edu.au/TAR/vlc.html

[5] Jeong, B., *and* Omiecinski, E., Inverted file partitioning schemes in multiple disk systems, *IEEE Transactions on Parallel and Distributed Systems*, *6* (2), 1995, 142-153.

[6] C. Fox, A stop list for General Text, SIGIR FORUM, ACM Press, Vol 24, No 4, December 1990.

[7] Robertson, S.E., Walker, S., Jones, S., Hancock-Beaulieu, M.M. and Gatford, M., Okapi at TREC-3, In: Harman, D.K., *Proceedings of Third Text Retrieval Conference, Gaithersburg, USA, November 1994.*

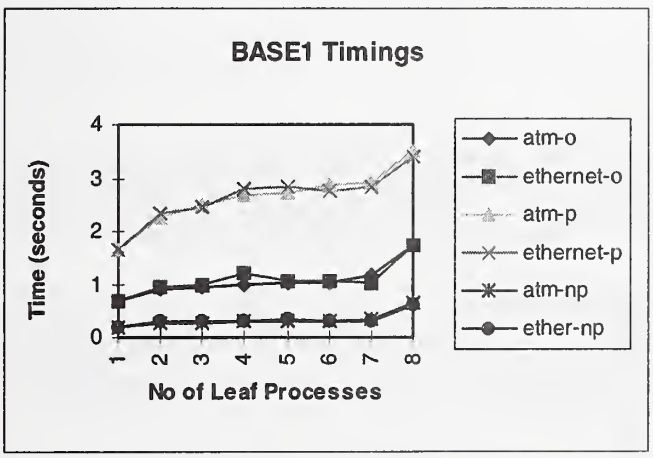
[8] S.E.Robertson, and K. Sparck Jones, Relevance Weighting of Search Terms, Journal of the American Society for Information Science, May-June 1996.

[9] S.E. Robertson, S. Walker, S. Jones, M.M. Beaulieu M. Gatford and A. Payne, Okapi at TREC-4, In: D.K.Harman, ed, *Proceedings of the Fourth Text Retrieval Conference, Gaithersburg, U.S.A, November 1995*, Gaithersburg: NIST 1996.

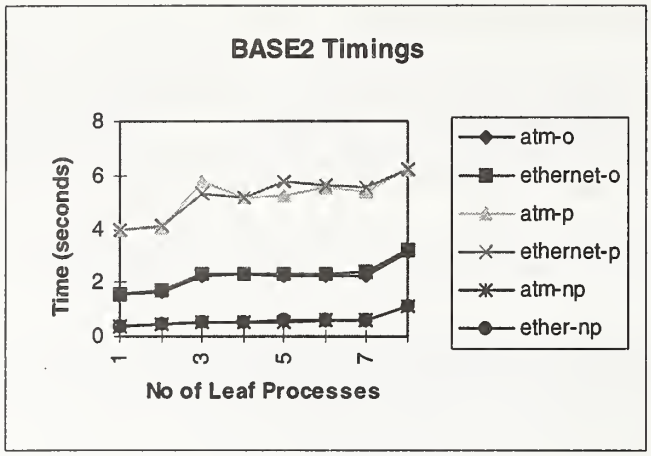
[10] Hawking, D. *The Design And Implementation Of A Parallel Document Retrieval Engine*. Technical Report TR-CS-95-08, Department of Computer Science. Canberra: Australian National University, 1995.

APPENDICES

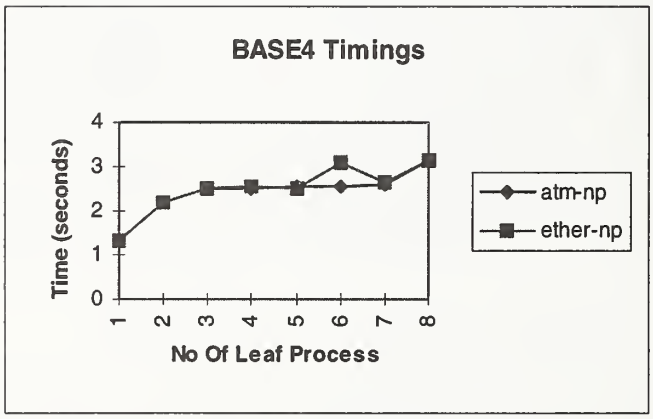
Note on Appendices: Each line on the graph is labelled with either the type of network e.g. ATM or Ethernet together with the type of query processing used: o for term weighing on Inverted files with position data, p for passage retrieval and np for term weighting on Inverted files with postings data only.



Appendix 1- Scaleup for Search on BASE1



Appendix 2 - Scaleup of Search on BASE2



Appendix 3 - Scaleup of Search on BASE4

EMIR at the CLIR track of TREC7

Frédérique Bisson, Jérôme Charron, Christian Fluhr, Dominique Schmit

Commissariat à l'Energie Atomique
DIST
CEA/Saclay
91191 Gif/Yvette cedex
France

E_mail : fluhr@tabarly.saclay.cea.fr

1 Introduction :

EMIR (European Multilingual Information retrieval) was a European ESPRIT project whose aim was to demonstrate the feasibility of a crosslingual interrogation based on the use of bilingual dictionaries. The project lasted from November 90 to April 94. A part of the results are included into a commercial product " SPIRIT " released by the T.GID Company in France.

2 Basic Principles of EMIR :

Principle of relevance ranking :

The system can be considered as a weighted boolean system. That means that the result of an interrogation is a partition of the database into classes of intersection, each of them being identified by the best boolean query that can retrieve the documents of this class.

Intersections are weighted according to the weight of each single word or compound that is in the intersection. The weight of each word is related to its discrimination power. That means that words pertaining to few documents have higher weights than words that are in a large number of documents.

Linguistic analysis :

The documents and the queries are processed by a linguistic analysis that normalizes words (synonyms are represented by the same character string and some homographs are solved and represented by different character strings). Compounds are recognized and normalized. To each normalized word or compound is associated a part of speech.

Reformulation :

To increase the relevance, a query expansion based on monolingual or bilingual reformulation rules is used. Each query word or compound can infer, according to its part of speech, synonyms or words derivated from the same root or translations into an other language.

The intersections are evaluated on the base of the original query words, that means that any of the inferred words from one query word can represent it.

3 Lessons from the previous TREC :

Our results in the TREC6 for the crosslingual track, can be considered not very good for the monolingual interrogation but the difference between mono and crosslingual interrogation was small.

So we have done a study to identify the causes of the level of monolingual results. In fact if we solve this problem we can also increase our results in crosslingual interrogation.

Several problems were identified :

Definition of relevance :

We have not the same definition of relevance than the one used in the crosslingual track. That means that some documents we consider relevant are considered in the "qrels" as non relevant. Probably the reason is that our system is tuned to access information and not to access documents. That means that if only one line in a 200 page document is relevant, we consider the document as relevant.

Ranking of classes :

The ranking of classes is not optimal. One of the main cause is that some compounds which are not important for the query but have a high weight because they are in few documents, can give a better weight to a non relevant intersection and push the good ones to the bottom of the list.

Incompleteness and inconsistency :

An other less important point, but that must be fixed, is the incompleteness and inconsistency of the various dictionaries and reformulation rules. They have a bad effect on both monolingual and bilingual reformulation.

Lack of a multi step reformulation :

In some cases bilingual reformulation must be followed by a monolingual one in the target language. It is especially the case when there is a change of part of speech in the translation of a word. For example an adjective can be translated as a noun.

Ex:

debt for Poland -> dette polonaise (Query 33)
génie génétique -> genetically engineered (Query 38)

4 The situation for TREC7 :

For TREC7 like for TREC 6, runs have been done using only the “desc” part of the topics. Misprints detected by the linguistic processing have been removed. A monolingual reformulation has been performed on the part of the database which is in the same language

than the query. A multilingual reformulation without target language monolingual reformulation has been performed when the query and database language are different.

Concerning the definition of relevance, we don't want to change our definition of relevance because our users are happy with this definition.

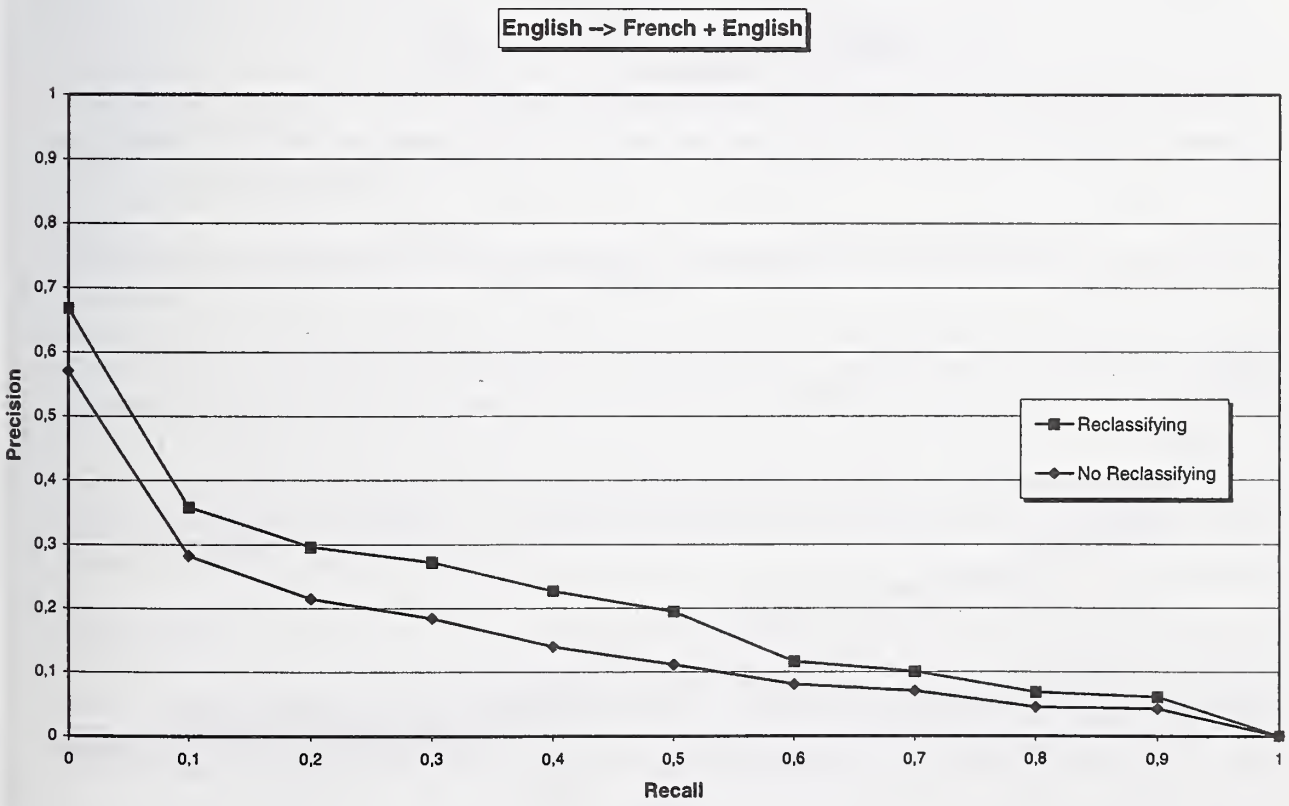
Concerning the ranking of classes, we think that there is two ways of improving the system.

The first one is by applying a better program for compound recognition and we have already one which is not implemented into the commercial product. The one currently used can give wrong compounds. This kind of wrong compounds are typically compounds that can disturb the document ranking.

Ex:
"oil from point " in query 27 (... method of delivering oil from point of origin to the shipping points ...)

Another reason of bad ranking is due to the fact that, in long queries, it is necessary to combine the weights computed from the database with weights measuring the importance of the words for the user. For example in the query about Lötschberg, if this proper noun is not in the intersection between a document and the query, the document is surely not relevant.

To experiment this hypothesis we have done two runs, one using the only weights computed from the database like last year and an other run where intersections that do not contain a compulsory word for the query are automatically pushed after the last intersection containing the compulsory word.



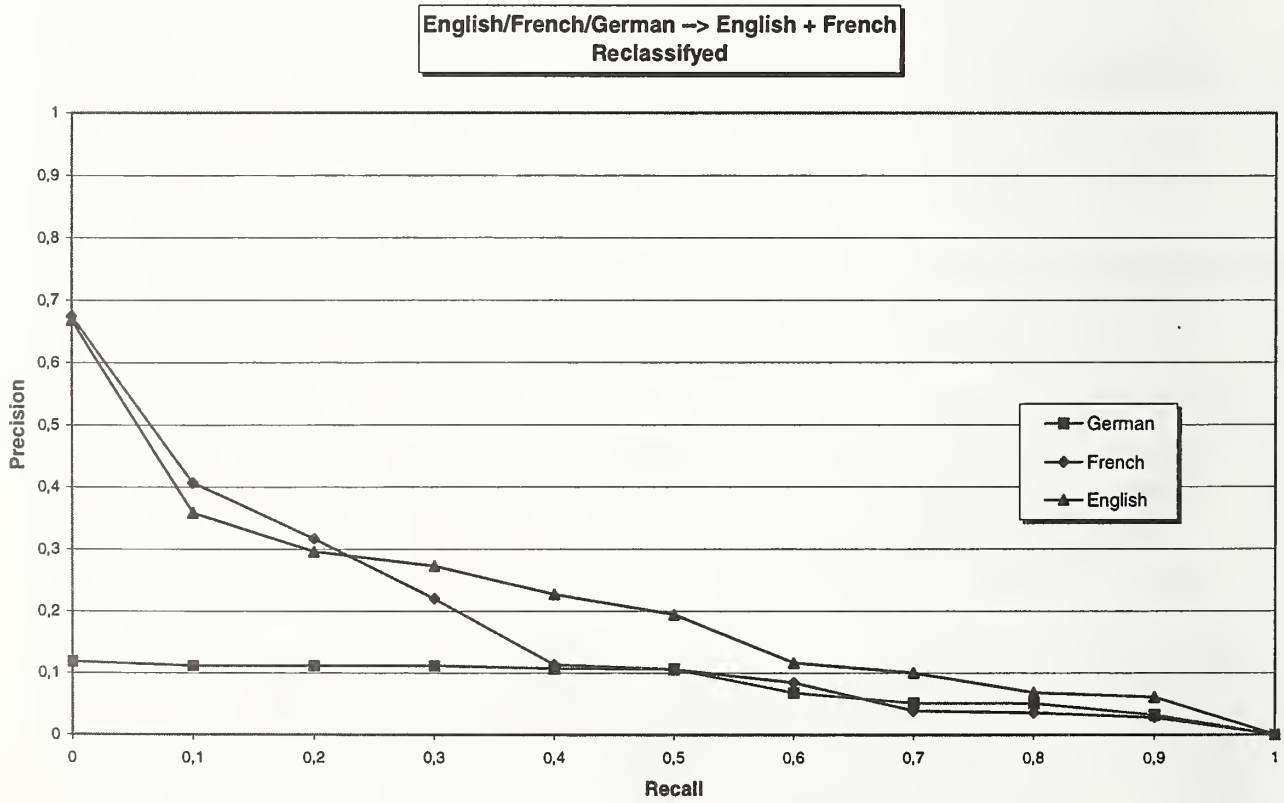
About the incompleteness, we have done a study of the vocabulary of the databases (both English and French) : unknown words have been added to the dictionary and in some cases we have introduced a compound into the automatic expression dictionary. Lacking translations have been added.

About inconsistency, it is a very important problem when large linguistic data is to be managed. At this time we manage monolingual dictionaries (single words and idiomatic expressions) for 5 languages, monolingual reformulation rules for 5 languages, bilingual reformulation rules for 9 couples of languages.

The monolingual dictionaries contain the relation between each word form and a normalized form that is used for retrieval. The normalized form is taken out of the list of equivalent forms. This includes flexion from the same lemma but also absolute synonyms like orthographical variations like "colour" - "color". Of course word forms inside the reformulation rules must be consistent with the choice of the normalized form.

Maintaining a full consistency manually is impossible. We are developing an architecture to ensure a consistency validation. At this time the development of this application is not finished and we are sure that in our runs for TREC7 some lack of intersection are due to this bad consistency.

This is especially visible in the results of the run with German queries because we have received from our German partner (TEXTEC in Saarbrücken) the bilingual dictionaries few days before the deadline and it was not possible to ensure consistency between the monolingual German dictionary and the bilingual reformulation one.



At last, as we have used the standard SPIRIT server, it was not possible to use the multistep reformulation tested in a prototype that is necessary to make a monolingual reformulation after the bilingual one. We expect that the functionalities of the prototype will be introduce before next TREC into the standard server.

5 Software architecture :

At the occasion of TREC 7 we have experimented in a larger way the multilingual, multibase architecture we intend to put into service at the end of this year. The French part of the database has been generated into 3 different databases. In that way it was possible to generate these 3 parts in parallel. The same was done for the English database that was split into 3 parts and generated in 3 different databases in parallel.

The interrogation is done through a web viewer. Databases in the same language as the query are interrogated using monolingual reformulation. Databases in a different language are interrogated using bilingual reformulation. All interrogations are done in parallel.

The interface between the internet viewer and a standard SPIRIT server executes the merging of results and the computation of a global weighting.

This merging is highly facilitated by the fact that the intersection is characterized by the words from the original query.

6 Automatic interrogation :

TREC 7 was also the occasion to test the system for automatic interrogation of information retrieval systems on the web. This system, which is named BeFor (Beyond Forms), has been developed by Jérôme Charron during his PhD. This system uses an XML description of the interrogation screen, an XML description of queries that can be more complicated than the one from TREC and can merge factual information and full text search, an XML description of how to use the result of the search.

In our case we have used this system to make an automatic run of TREC queries to the Web interrogation interface we have used to test TREC.

This system can manage at the same time several sets of queries and several interrogation applications. The extraction of results is converted into a TRECEVAL format.

It has been chosen by the organizer of AMARYLLIS to make an automatic run through internet directly by the organizer without any possibility of intervention by the author of the tested IR systems.

7 References :

About reformulation in full text IRS, F. Debili, C. Fluhr, P Radasoa, Conference RIAO 88, MIT Cambridge, march 1988, Modified version has been published in "Information processing and management" Vol. 25, N° 6 1989, pp 647-657.

EMIR Final report, Christian Fluhr, Patrick Mordini, André Moulin, Erwin Stegentritt, ESPRIT project 5312, DG III, Commission of the European Union, october 1994

Multilingual database and crosslingual interrogation in a real internet application, C. Fluhr, D. Schmit, F. Elkateb, K. Gurtner, workshop "Cross-language Text and Speech retrieval" in "AAAI 1997 Spring Symposium Series", 24-26 march 1997, Stanford University, California.

SPIRIT-W3, A distributed crosslingual indexing and retrieval engine, C. Fluhr, D. Schmit, Ph. Ortet, F. Elkateb, K. Gurtner, INET'97, Kuala Lumpur, June 1997.

EMIR at the crosslingual track of TREC-6, F. Elkateb and C. Fluhr, TREC-6 Conference, 19-21 November 1997, Gaitherburg, Maryland

Cross-language information retrieval, Grefenstette, G. and alii., Boston: Kluwer Academic Publishers, 1998.

TREC-7 Ad-Hoc, High Precision and Filtering Experiments using PIRCS

K.L. Kwok, L. Grunfeld, M. Chan & N. Dinstl

Computer Science Department
Queens College, CUNY

C. Cool

Graduate School of Library & Information Studies
Queens College, CUNY

Abstract

In TREC-7, we participated in the main task of automatic ad-hoc retrieval as well as the high precision and filtering tracks. For ad-hoc, three experiments were done with query types of short (title section of a topic), medium (description section) and long (all sections) lengths. We used a sequence of five methods to handle the short and medium length queries. For long queries we employed a re-ranking method based on evidence from matching query phrases in document windows in both stages of a 2-stage retrieval. Results are well above median. For high precision track, we employed our interactive PIRCS system for the first time. In adaptive filtering, we concentrate on dynamically varying the retrieval status value threshold for deciding selection and during the course of filtering. Query weights were trained but expansion was not done. We also submitted results for batch filtering and standard routing based on methods evolved from previous TREC experiments.

1. Introduction

We continue to use our PIRCS system for experimentation. This system has been described in previous TREC proceedings and summarized in TREC-6. For TREC-7, we participated in the main task of automatic ad-hoc retrieval as well as the high precision and filtering tracks. Two strategies we used throughout for ad-hoc are 2-stage retrieval and collection enrichment. 2-stage retrieval means using a raw query for initial retrieval, employ the top-ranked document list to define the domain of the query and expand the initial query in a pseudo-relevance feedback fashion, and then do a 2nd retrieval to report final results. Collection enrichment means adding external documents to the target collection during 1st stage in order to enhance the quality of the pseudo-feedback documents. Both strategies have been found to work more often than not for queries of different lengths and applicable to different languages such as English and Chinese. Ad-hoc retrievals are discussed in Section 2.

This year we introduce an interactive PIRCS system, and users can interrogate the TREC-7 collection via a GUI that is based on the look and feel of ZPRISE from NIST. We used this facility for our High Precision track to find as many relevant documents as possible within the top fifteen retrieved. Results are reported in Section 3.

We also participated in the Filtering track, in particular, adaptive filtering being done the first time. For this, we emphasize on dynamically setting a RSV (retrieval status value) threshold to select or not select a document for examination. Adaptive filtering is described in Section 4 while batch filtering and routing work are described in Section 5. Section 6 contains our conclusions.

2 Ad-Hoc Retrieval

The target collection for ad hoc retrieval is Disks 4&5, similar to TREC-6 except that Congressional Record documents are not considered. This leaves the collection to consist of Financial Times, Federal Register, Foreign Broadcast Information Service and the LA Times, some 1.9 GB of text in over ½ million documents. We did not bother to re-create a new dictionary and continue to use last year's data that include terms and statistics from the Congressional Record. The documents however were removed from retrieval. As usual, we broke long documents into sub-documents of about 550 words long ending on a paragraph boundary, resulting in some 761K sub-documents. The dictionary of unique terms is 727K in size after stopword removal and conflation; it became 325K in size after Zipf thresholding.

As in previous TRECs, topics for TREC-7 are described in several sections: title, description and narrative. This year, each section contains self-sufficient information so that queries can be constructed from only the Title (short length), the Description (medium) or from All sections (long). Their average number of unique terms after stopword removal, stem conflation and thresholding are 2.62, 7.04 and 17.18 respectively. Short queries lack descriptive term variety and term weighting information

and pose especially difficult situations for a search engine. Queries using All sections are paragraph(s) in size. One does not expect casual users to issue such long queries. On the other hand analysts, information specialists or dedicated information seekers should not refrain from elaborating their needs in such verbose fashion with variety and redundancy, since long queries usually give better results than short. In this experiment, we treat short and medium queries similarly and are discussed in Section 2.1. Long queries are described in Section 2.2.

2.1 Short and Medium Queries

Over the past several years our efforts to investigate short query retrievals have accumulated in [KwCh98] where we applied five methods in sequence to bring effectiveness some 32 to 72% above the initial 1-stage retrieval results for TREC-6&5 environments. These same procedures were used in TREC-7, and it provides a blind experimental test of the approach. The methods we employed are: 1) avtf query term weighting, 2) variable high frequency Zipfian threshold, 3) collection enrichment, 4) enhancing term variety in raw queries, and 5) using retrieved document local term statistics. Avtf employs collection statistics to weight terms in short queries [Kwok96]. Variable high frequency threshold defines and ignores statistical stopwords based on query length. Collection enrichment adds other collections to the one under investigation so as to improve the chance of ranking more relevant documents in the top n for the pseudo-feedback process [WRBJJ98, KwCh98]. Enhancing term variety to raw queries means finding highly associated terms in the initially retrieved documents that are domain-related to the query [XuCr96]. Making the query longer may improve another round of 1st stage retrieval. And retrieved document local statistics re-weight terms in the 2nd stage using the set of domain-related documents rather than the whole collection as used during the initial stage [SiMB97]. For collection enrichment, we form a miscellaneous collection by retrieving the top 200 documents from the sub-collections of Disks 1-3 using the short form of the queries. This miscellaneous collection is used to enrich the retrieval operation during the initial stage retrieval. Some of these documents are ranked high and they are employed to expand the queries during pseudo-relevance feedback.

	Query Type		
	Title % Short	Desc % Medium	All % Long
Relv.Ret (at most)	2983 0 (4674)	3034 2 (4674)	3162 6 (4674)
Avg.Prec	.2427 0	.2543 5	.2723 12
P@10	.4480 0	.4600 3	.4960 11
P@20	.3770 0	.3930 4	.4340 15
P@30	.3353 0	.3613 8	.3947 18
R.Prec	.2705 0	.2831 5	.2960 9

Table 1: Automatic Ad Hoc Results for 50 Queries

Results and Discussion

Our TREC-7 results for short and medium queries are summarized in Table 1 under the columns Title and Desc. These runs are named 'pirc8At' and 'pirc8Ad' respectively. This year, all important terms that appear in the Title section are also repeated in the Description section. The official evaluation measure is average non-interpolated precision (Avg.Prec) which are 0.2427 and 0.2543 respectively. The number of relevants recovered at 1000 documents (Relv.Ret) is respectively 2983 and 3034 out of a maximum of 4674, or about 64% and 65%. The behavior of the two retrievals are very close to each other, with the Desc queries giving an edge of between 2 to 8% uniformly in various average precision values. The number of medium length queries performing better than or about equal to short ones in average non-interpolated precision is 24 out of 50. However, for relevants retrieved at 1000, this number is 40 out of 50. Thus, most medium length queries have larger recall.

Comparisons with the all-sites median average precision, precision at 100 and 1000 documents are given in Table 2. The average precision for short queries is better or equal to median in 33 instances with 2 queries achieving the best, and are worse than median in 17 instances. Similar behavior is also observed for medium length queries. For example, the precision at 100 documents for medium length queries is better or equal to median in 39 instances with 4 being the best, and is worse than median in 11 instances with 1 case being the worst. The median was calculated using results from all automatic ad-hoc experiments without differentiation of query lengths. Thus, our results are well above median. This year there are 7 queries with a single term. Two of these (#364 "rabies" and #392 "robotics")

	Query Type								
	Title Short			Desc Medium			All Long		
	>	=	<	>	=	<	>	=	<
Avg.Prec	32,2	1	17	33,3	0	17	42,3	0	8
RR@100	27,1	8	15,2	30,4	9	11,1	36,2	7	7,2
RR@1K	37,7	2	11	38,13	4	8	44,10	2	4

Table 2: Ad-Hoc Results: Comparison with Median

	← Zipf Threshold 65K →			← Zipf Variable Threshold, max 100K →				
Method	**0**		**1**	**2**	**3**	**4**	**5a**	**5b**
	1st Stage	2nd Stage	2nd	2nd	2nd	2nd	2nd	2nd
				TREC-7	**Short**			
Relv.Retr	2299	**2964**	2960	2989	2971	2984	**2983**	2965
Avg.Prec	.180	**.226**	.228	.227	.236	.242	**.243**	.238
P@10	.386	**.408**	.406	.398	.434	.446	**.448**	.430
				TREC-6	**Short**			
Relv.Retr	2188	**2272**	2384	2517	2656	2738	**2739**	2691
Avg.Prec	.220	**.240**	.258	.258	.284	.289	**.291**	.290
P@10	.334	**.372**	.402	.388	.444	.442	**.450**	.440
				TREC-7	**Medium**			
Relv.Retr	2285	**2982**	3075	3067	2987	3062	**3034**	2978
Avg.Prec	.167	**.240**	.236	.231	.238	.253	**.254**	.240
P@10	.370	**.454**	.470	.452	.458	.462	**.460**	.452
				TREC-5	**Medium**			
Relv.Retr	1763	**2279**	2335	2635	2732	2787	**2792**	2735
Avg.Prec	.140	**.161**	.181	.214	.234	.239	**.241**	.234
P@10	.290	**.284**	.326	.372	.382	.404	**.406**	.378

Method 0: Standard 1st & 2nd Stage Retrieval; 1: add avtf weighting; 2: add variable Zipf threshold; 3: add collection enrichment; 4: add term variety; 5a: add local statistics weighting; 5b: 5a without adding term variety.

Table 3: Comparing TREC-7 Short & Medium Length Query Results with TREC-6 and TREC-5

have average precision 0.418 and 0.375 respectively. The rest have values below 0.2. This is unlike TREC-6 where there were very specific single term queries (like #312 "hydroponics", #316 "polygamy" and #348 "agoraphobia") that achieve average precision well above 0.5 in our system. Thus, in year to year average precision comparison, one should take this into account.

Table 3 tabulates retrieval results as each of the five methods discussed earlier is sequentially applied. This is compared with similar results achieved for TREC-6&5 queries and reported in [KwCh98]. It is observed that both Methods 1 (avtf weighting) and 3 (Collection enrichment) produce positive but less pronounced effects for TREC-7 than for TREC-6. Otherwise the behavior is similar and we do get about 8% improvements over standard 2-stage retrieval that does not use any of the 5 methods (i.e. Method 5a vs. Method 0 average precision of 0.243 vs. 0.226).

Comparison of the medium length query results with TREC-5 data is less favorable. Both the avtf weighting (Method 1) and the variable Zipf threshold (Method 2) precision decreases compared with not using them (Method 0). It is possible that parameters set for short queries of very few terms should not be the same as for medium queries. However, the collection enrichment (Method 3) and adding term variety (Method 4) improve the result. The overall improvement is about 6% (i.e. Method 5a vs. Method 0 average precision of 0.254 vs. 0.240).

2.2 Long Queries

For long queries, we use all sections of each topic and the run is named 'pirc8Aa2'. The average number of unique terms after the usual processing is 17.18. Of the five short-query techniques described previously, we retain two, viz. collection enrichment and local statistics for irrelevant documents. The high Zipf threshold is fixed at 100K. On the other hand we added a phrase re-ranking methodology. The assumption is that query phrases represent more precise concepts than single terms. If these phrases are found within small document windows, this might contribute additional evidence of relevance and may help re-rank more relevant documents to the top of the retrieval list. This procedure can be employed at the initial retrieval stage to provide more effective pseudo-relevance feedback to improve 2nd stage retrieval. It can also be employed for the final retrieval list to report better results.

Employing phrases to improve retrieval has a long history since the early days of the SMART project. With the recent Tipster and TREC programs, large corpora become available, and there are ambitious attempts to use natural language techniques to discover more precise phrases from gigabyte-size collections which can take days of processing time [e.g. STCM95]. However, it appears that the promise of better retrieval by phrases has not yet materialized as witnessed by years of TREC blind retrieval experiments

	← Standard →			← Phrase Re-Rank →		
	1st Stage A	2nd Stage B	Coll Enrich 2nd Stage C	Rerank C C'	Rerank 1st, 2nd Stage D	Rerank D D'
			TREC-7	Long		
Relv.Retr.	2717	3142	3165	3167	3160	3162
Avg.Prec	.214	.257	.266	.269	.271	.272
P@10	.466	.486	.502	.488	.518	.496
			TREC-6	Long		
Relv.Retr.	2537	2947	3043	3064	3074	3088
Avg.Prec	.237	.264	.305	.310	.304	.308
P@10	.402	.452	.492	.498	.488	.490
			TREC-5	Long		
Relv.Retr.	2463	3077	3034	3049	3052	3072
Avg.Prec	.220	.253	.262	.265	.270	.273
P@10	.404	.414	.438	.440	.446	.444

Table 4: Effect of Collection Enrichment & Phrase Re-Ranking on Long Query Retrieval

where the best retrievals are mainly term based.

With this background, we also tried our hands on using phrases but with a less ambitious goal. Our opinion is that weighted term level retrieval is simple and has provided most rewarding results and should form the basis of any retrieval. Phrases may be used to provide 2nd order effects of improvements based on re-ranking of retrieval lists. The difficulty is how to weight their contribution with respect to term level effects so as to achieve cooperation between the two levels of evidence. This is explored empirically by performing a number of experiments.

Given a topic, we first used our standard PIRCS processing to perform a 2-stage retrieval with collection enrichment and local statistics weighting in the final stage. The same topic is then processed by the Mitre POS-tagger to tag each word syntactically, and a simple program is used to bracket noun phrases. We kept only contiguous adjectives and nouns of two or more words long as these presumably would be most content bearing. These phrases are then searched in each document of a retrieval list to discover their occurrence pattern. Only query phrases that occur within a 3-sentence window of a document are considered; any occurrences of longer spans are ignored.

Re-ranking is based on adding a factor that is proportional to the original RSV of the document in a retrieval list, thus:

$$RSV = RSV * (1 + \text{phr-factor})$$

$$\text{phr-factor} = \text{constant} * \text{Matching} * \text{Coverage}$$

where Matching = M(.) is a function that evaluates how good the query phrases are matched within document windows and sum their contributions. When there are matches of three or more terms of a phrase, it is worth twice as much as two-term matches. In addition, the number of times such matching occurs and the size of the query phrase also influence the evaluation. Coverage = C(.) is another function that evaluates how complete a document covers the entire query concept phrases. Below certain thresholds, no re-ranking takes place. For example, if there are two or more three-term matching phrases, the threshold is satisfied. If there are only two-term matching phrases, then the threshold is graded according to the total number of phrases in a query.

Results and Discussion

Long query results are also shown in Tables 1 and 2. The average precision value is 0.2723, and the relevants retrieved at 1000 documents is 3162, about 68% of the maximum 4674. When compared with the all-sites median, our average precision is better in 42 instances with 3 cases being best, and 8 instances are below median. This appears also to be quite good achievement.

Table 4 shows the successive effects of collection enrichment (Column C) and phrase re-ranking on retrieval (D') for these long queries. They add 3.5% (0.266 vs. 0.257) and 2% (0.272 vs. 0.266) respectively to the average precision measure. The behavior is quite similar to what is found in TREC-6 & 5 which are also shown. Column C' and D depicts the intermediate steps of phrase re-ranking: C' shows the effect of re-ranking 2nd stage retrieval only, while D shows re-ranking 1st stage without doing re-ranking on 2nd stage. It seems preferable to do both.

Comparing the results in Table 1 and 2, it can be seen that for this set of queries and documents, long queries are uniformly preferable in average precision to short (between 6% to 18% better), and medium queries performed somewhere in-between (2% to 8% better than short) using

our PIRCS system. Compared to short queries, 34 out of 50 instances have better or equal average non-interpolated precision. This becomes 33 out of 50 when compared to median. However, when the number of relevants retrieved at 1000 is used as the measure for comparison, then ALL long queries are better or equal to short and medium queries. It appears that to obtain good recall, long query length is an especially important factor: the longer the better.

The average precision varies between 24 to 27 percent from short to long queries. This level of effectiveness is far from ideal and needs to be improved by further research and investigation. On the other hand, the precision at 10 documents varies between .448 to .496, meaning that between 4 to 5 documents out of the top 10 retrieved are relevant. For many users who do not need exhaustive retrieval, this represents reasonable performance.

3. High Precision Track

Description of the experimental condition: a single volunteer searched all 50 ad-hoc queries. The searcher in this experiment was a female who had five years of experience using online information retrieval systems. The searcher held a Ph.D. degree and was an assistant professor in the Graduate School of Library Studies at Queens College. At the time of the experiment, the ad-hoc queries were new to the searcher. No prior searching had been performed on any of the topics. The searcher had no prior experience searching on the PIRCS system. The task given to the searcher was to "create a final query that will produce a retrieved set in which the top 15 documents has the highest precision." The single searcher in this experiment employed the following strategy in order to maximize high precision results. First, each query and narrative were read, and a small number of topically relevant words were entered in the query window. Top ranked documents in the retrieved set were quickly examined for relevance, and the full text then displayed in order to identify additional terms for use in query expansion, as well as terms to eliminate. The large size of the full text display window, combined with highlighting of the query terms, facilitated quick examination of the performance of each query, in its particular context within the text. Automatic query expansion was not supported at the time by the retrieval system, so query reformulation was done manually. After every query iteration, the ranked list of documents was judged by the searcher for relevance. Non-relevant documents were marked, and subsequently removed from the ranked list. In other words, the searcher did not "save" individual relevant documents along the way over the course of the search episode. Instead, the searcher created one final query that would retrieve the best possible ranked list, after eliminating the marked non-relevant documents.

Searching time: Each of the 50 queries was searched within the five-minute time limit established by the Track. The number of iterations for each topic search ranged from one to five, with an overall average of 2.1.

Results and Discussion

Precision at 15 documents ranged from .00 to 1.00, with an average overall precision of .48. In a comparison of search results across all seven systems in the high precision track, Table.5, our system performed at or above the median in 48% of the queries and below the median in 52% of queries. Topics # 365 and 372 were the easiest queries for the searcher to search, and attained the highest performance levels, with 100% precision at top 15. No relevant documents were retrieved for Topic 353, due to a misspelling of the query term "Antarctica."

	Above	At median	Below
Precision @ 15	**8**	**16**	**26**

Table 5: High Precision - Comparison with Median

4. Adaptive Filtering Track

We participated in all three sub-tracks within this year's filtering track, viz.: adaptive filtering, routing and batch filtering. They made use of TREC-1 queries 1 to 50 and run against the target AP88, 89 and 90 collections. This section describes our effort in adaptive filtering. Routing and batch filtering are described in Section 5.

We actually did filtering via sub-documents since our old files have already segmented documents. If any or multiple sub-documents of the same document are selected, we regard it as just one document selected. RSV's of different sub-documents are not added, as is done for ad-hoc.

Many considerations are needed for adaptive filtering. These include defining an initial query together with an initial filtering threshold to start the process, adaptively train the query to tailor to the relevant document types seen so far, dynamically change the threshold to select or not select a document for examination, determine how often these changes are to be performed, and at the same time attempt to maximize a target utility value. The task is made more difficult by excluding any use of the topics or the target collections for initial training purposes. The overall properties of the terms encountered (such as collection term frequency, document frequency of terms, etc.) can be captured after the documents have gone through the filtering process, but relevancy information can be used

only if a document has been selected for examination. These data can subsequently be used for training purposes.

To prepare for filtering, a dictionary was defined by processing some 1.2 GB of texts consisting of LA, WSJ-1, DOE and ZIFF-1 collections. These were chosen to reflect the 1988-1990 period news articles as well as introducing some scientific and computer terminology into the dictionary with reasonable term usage statistics. The dictionary size after stopword removal, conflation and Zipf thresholding is about 240K. To help debug some new programs, a mirror problem was set up using queries numbered 51 to 100 for filtering a target collection consisting of WSJ 90-92.

We first define empirically two fixed RSV thresholds Thi and Tlo. Documents with RSV>Thi should have high probability of being relevant to a query, and the opposite is true for documents with RSV<=Tlo. Based on our experience with PIRCS in general we set Thi = 2 or 3 and Tlo = 1.3 or 1.6 depending on whether the objective utility is lax or strict. At the start, the current RSV threshold T was set to Thi in order to improve the chance of getting relevant documents for learning. This contrasts with more realistic filtering situations where one might set a low initial threshold in order to return some documents for the user in order to avoid null answers.

Once the process starts, statistics of term usage is kept for all documents filtered, as well as separate term usage statistics for documents that were selected and found relevant. We did not store any document ID's, their individual or textual properties. In addition, a running total of the number of documents N that pass through the system, the number examined Ne, and the number found relevant Nr are also kept. This allows us to evaluate an overall average precision Nr/Ne for the user and the proportion of documents examined Ne/N at any time.

To reduce the complexity of the problem we decided to implement query weight adjustment only, and not query expansion. The starting topic descriptions are paragraphs in length augmented with concepts and are very rich already. We conjecture that not doing query expansion may not be too much of a loss. Moreover, the RSV's of documents would be in a more stable range. The update schedule is arbitrarily set to once every 10,000 documents filtered so as to be more efficient. Since there is a total of about 323K documents, we updated about 30 times. The weight of each of our query term is defined by log (p/(1-p))*((1-q)/q), where p = Pr(term present|Relevnat) and q = Pr(term present|~Relevant). These p and q values are modified using the captured term statistics during each update.

We try to dynamically adjust the RSV threshold T (to determine select or not select a document) based on N, Nr, and Ne. Specifically:

if (no relevants seen yet)
T = T*x1 when T >Tlo & Ne/N<SRT
else if (Nr/Ne < Fi & Nr > Nd) T = T*x2
else if (Nr/Ne < Fi & Nr <= Nd) T = T*x3
else T = T*x4 when Nr <= Nd
where x1=0.95; x2=1.03; x3=1.005; x4=0.97.

SRT (selection rate threshold) is set to 0.001 to prevent relaxing T too much if there is too much selection already but no relevant document has been obtained. Fi relates to the utility objectives and is set to 0.4 and 0.2 when i=1,2. Nd is a relevance document threshold (set to 30) used to control against too much restriction in T when not too many relevant documents have been seen yet. The parameters are biased towards a tighter condition than Fi.

Results & Discussion

Table 6a,b summarized the results of the adaptive filtering runs which are named 'pirc8FA1' and 'pirc8FA3' respectively for the two utilities F1 and F3. F1 aims at selecting all documents with a probability of relevance > 0.4, and for F3 it is a much more relaxed 0.2. The tables

	Comparison with Median			F1 Utility		prec/rec		#of qrys	
	>	=	<	score	docs	P	R	N+/0	N-
AP88	37,8	2	11	-93	669	.372	.097	34	16
AP89	21,6	4	25,3	-147	1841	.384	.214	20	30
AP90	16,7	4	30,3	-918	1304	.259	.170	18	32
AP89-90				-1065	3145	.332	.198	19	31

Table 6a: Adaptive Filtering F1 Results

	Comparison with Median			F3 Utility		prec/rec		#of qrys	
	>	=	<	score	docs	P	R	N+/0	N-
AP88	27,3	6	17	982	3748	.252	.369	27	23
AP89	24,5	3	23,3	2900	5245	.311	.493	26	24
AP90	22,9	5	23,5	756	3729	.241	.452	24	26
AP89-90				3656	8974	.281	.478	26	24

Table 6b: Adaptive Filtering F3 Results

show comparison with the median of all runs, the F score, number of documents selected, precision and recall values, and the number of queries with >=0 scores and negative scores. Table 6a shows that results for AP88 compares favorably with the median, drops some for AP89 and precipitously for AP90. On closer examination, it turns out that the parameter Tlo was not set right: it was left at 1.3 from previous runs, rather than the higher 1.6 value. It appears many documents with RSV's between 1.6 and 1.3

are allowed to be selected and these have relevance density much less than 0.4. When we repeated the runs with Tlo set to our original 1.6 value, the F1 scores become -93, 172, & -437 and the documents selected become 669, 1579 and 976 respectively for AP88, 89 and 90. The scores are substantially better, but still in the negative score territory for AP90. Table 6b shows the same runs for utility F3. Our runs achieve positive utility and compares favorably with median for all years.

5. Routing Retrieval and Batch Filtering

5.1 Genetic Algorithms

Genetic Algorithms (GA) [Holl75,Gold89,Davi91,Mitc96] are search procedures based on survival of the fittest. A general description of GA may be summarized as follows:

```
Procedure GA
begin
        initialize population P(0)
        evaluate P(0)
        t=1
        repeat
                select P(t) from P(t-1)
                recombine P(t)
                evaluate P(t)
        until (termination condition)
end
```

We utilize the TREC7 data as a test-bed for our research in applying genetic algorithms learning for Information Retrieval in conjunction with our PIRCS system.

To apply genetic algorithms to a specific application domain, one must devise a representation of the potential solution. Then we start the competition between the various potential solutions as specified by the genetic algorithm. The arbitrator of the competition is the fitness function, which is also application specific.

5.1.1 Representation of IR for GA

We describe four types of representations for GA-based approaches to information retrieval below.

1) A retrieval system assigns various weights for its retrieval function. Most frequently, weights are assigned to terms based on various statistics or learning algorithms. A GA can search the solution space by multiplying these weights by 0 or 1. This is equivalent to performing selection of a subset of the original terms.

2) If we allow the multiplier to be a real number in the range [0,1] in the previous scheme, then the weights will be modified and the GA performs a weight optimization.

3) Assuming a retrieval system generates a query with N given training documents, then a subset of these documents may generate a different query. Allowing the GA to search this space will often result in a superior query. This scheme can be further refined, by assigning weights to training documents. This is similar to bagging and boosting [Brei96,FrSc96,Quin96].

4) Given retrieval system r, which assigns a Retrieval Status Value (RSV_r) to retrieved documents, the output of different retrieval systems can be combined by some function f (RSV_r). A simple function of this type is the linear sum_of(a_r * RSV_r) where the a_r are arbitrary weights. A GA can search this space to yield a superior combination. This GA performs a 'second level' retrieval.

5.1.2 GA Fitness Functions for IR

The second domain specific component that has to be defined for the GA is the fitness function. A good fitness function should reflect the quality of retrieval. We describe three possibilities below.

1) Gordon [Gord88] and Chen [Chen95,ChSS98] have used the Jaccard similarity measure in previous GA research in IR. This measure is related to the number of retrieved relevant records and does not account for ranking.

2) The maximum likelihood measure is used for logistic regression. The advantage of using this fitness function is that it yields an RSV that corresponds to the probability of the document being relevant.

3) Average uninterpolated precision is the evaluation measure used by TREC. Since this is the measure one really tries to optimize, it is a logical choice for a fitness function.

5.2 Routing Retrieval Track

5.2.1 Routing query Pirc8R1

Pirc8R1 is a PIRCS retrieval based only on the query text and the term statistics of the AP88 collection without any training. It serves as a baseline measure for the effectiveness of other retrievals.

5.2.2 Routing query Pirc8R2

Pirc8R2 is a 'second level' retrieval by a GA, described in Section 5.1.1 as the 4th possible representation type. Nine retrievals were combined. They are described below:

a) Retrieval with no training documents (not1 and not2)

The PIRCS retrieval system itself is a combination of two networks: one is called query-focused retrieval and the other is called document-focused retrieval depending on the direction of activation spreading. Usually the user is allowed control over the combination coefficient to fine tune retrieval effectiveness. Here, we let the GA choose the coefficient. These two retrievals will be referred to as not1 and not2.

b) Standard PIRCS retrievals (pir1 and pir2)

We similarly create two retrievals using the standard PIRCS retrieval engine. All relevant documents from the AP88 collection were used for the training. Term expansion was set to 250 terms. These two retrievals are called pir1 and pir2.

c) GA used for term selection (gt1)

This is 1st representation for IR described above. The evaluated relevant and not relevant documents were divided into two groups. One group was used to select 300 candidate terms using the standard PIRCS term selection algorithm. The second group was used to train the GA. On the average about 1/3 of the terms were selected.

d) GA used for weight optimization (gw1 and gw2)

This is 2nd representation for IR described above. The evaluated relevant and not relevant documents were divided into two groups. One group was used to select 100 candidate terms using the standard PIRCS term selection algorithm. The second group was used to train the GA. The resulting queries with modified weights were used to yield separate retrievals.

e) Pure GA retrieval (ga)

The previous GA was used to optimize an existing retrieval system using the average uninterpolated precision as their evaluation measure. We also experimented with a pure GA, which uses the maximum likelihood measure. For terms we use about 300 terms, about 100 term pairs and also term triples and quadruples. Since all processing and evaluations are done internally and faster, we were able to use larger populations and more generations than in the previous cases. However we were not able to retrieve from the full AP89/90 collections, so the final retrieval was on 1500 documents selected by the standard PIRCS system.

f) Backpropagation network (bp)

The final retrieval is a backpropagation neural network. We modified NevProp, a publicly available c program maintained by Phil Goodman of the University of Nevada. The terms, test data and retrieval were done in the same way as the pure GA retrieval.

Details of the nine retrieval methods are summarized in Table 10 at the end of this paper.

Selecting training documents for the second level retrieval required some compromises. The ideal situation would be if unseen documents were used, since second level retrieval attempts to predict the behavior of these retrievals for new documents. However many of the component training methods were exposed all training documents. Some of the choices we considered were:

1) Reserve some portion of training data for second level retrieval only. This was deemed unpractical due to the small number of training data for many queries.
2) The only method trained on all training data was the pir1 and pir2. The other methods used half of the training data for term selection only. Create a version of pir1 and pir2 using only half the data to find the combination function and use the fully trained version for the actual addition. In this case too some queries would have inadequate number of training data.
3) Same as 2) but use all training documents. This was the method selected, so that we can use a uniform method for all queries.

5.3 Routing Retrieval Results

Table 7 shows our results in comparison with the median. The combined retrieval Pirc8R2 obviously performed much better than the retrieval without training Pirc8R1. Considering that not all available judged documents were used, only those from AP88, the results are quite good.
Table 8 shows performance of the various methods for the 43 queries, which had both relevant training and testing documents. The method column contains the 9 methods, the 2 official routing runs and max which is the retrieval we get if we choose the best performing method for each query. As can be seen the individual retrievals did not do well. They all performed worse than not2, the better of the retrievals with no training, except for pir1 and pir2, the classic PIRCS retrievals. However the second level retrieval (pir98r2) was successful. It improved 44% over not2, 17% over pir1 and was only 1% worse than max, which is a hypothetical combination that can only be done if results are available.

A possible explanation is the lack of adequate number of training documents. If we just look at the 13 queries, which had the most relevant training documents, the results are

	Pirc8R1			Pirc8R2		
	>	=	<	>	=	<
avg prec	11(3)	2	36	30(7)	7	12

Table 7 Comparison of routing results with median. Number in parenthesis is number of best values.

different. All training methods are better than no training, although still not as good as pir1. Also the second level retrieval outperforms max.
Apparently the 'second level' retrieval is not that much affected by the lack of training documents as the other methods, since its dimensionality (9, the number of retrievals combined) is much lower than the dimensionality of the other methods (>100, the number of terms).

The implication of this experiment is, that it may be quite possible to achieve a retrieval that is close to the maximum of all TREC7 retrievals by a genetic algorithm based second level retrieval.

method	*43 qry*	*% imprv over not2*	*13 qry*	*% imprv over not2*
not1	0.2699	-5.7%	0.2626	-17.7%
not2	0.2862	0.0%	0.3193	0.0%
pir1	0.3519	23.0%	0.4531	41.9%
pir2	0.3341	16.8%	0.4351	36.3%
gw1	0.2705	-5.5%	0.4402	37.9%
gw2	0.2724	-4.8%	0.4118	29.0%
gt1	0.2320	-18.9%	0.3533	10.7%
np	0.2619	-8.5%	0.3676	15.1%
ga	0.2536	-11.4%	0.3614	13.2%
max	0.4172	45.8%	0.5021	57.3%
pir98r1	0.3145	9.9%	0.3721	16.5%
pir98r2	0.4136	44.5%	0.5208	63.1%

Table 8: Individual retrieval results.

5.3 Batch Filtering Track

The Pirc8R2 query was used for batch filtering for the 44 queries, which had relevant training documents. For the others 6 queries, no documents were submitted for the F1 measure. For F3 measure, if the RSV was larger then the largest RSV for Pirc8R1 retrieval then they were selected. Previously we used the maximal F measure point to select the cutoff. This time we utilized a logistic regression transformation. (using a GA). In the past we found that performance in the batch-filtering task was highly correlated with performance in routing, therefore the average performance shown above is a good sign. Table 9 below shows how we compared with the median.

run	>	=	<
Pirc8FB1	23 (11)	17 (10)	10
Pirc8FB3	26, (8)	16 (8)	8

Table 9: Comparison of batch filtering results with median. Number in parenthesis is number of best values.

6. Conclusion

Our system continues to perform well for the ad-hoc short queries. Adaptive filtering was done for the first time. Our simple approach of adjusting thresholds that does not involve storing large amounts of past data seems to work reasonably well. We also show that combination of retrieval based on a genetic algorithms approach can provide superior results in both routing and batch filtering.

Acknowledgments

This work is partially supported by an U.S. Department of Defense contract MDA904-96-C-1481.

References

[Brei96] Breiman, L. (1996) Bagging predictors. Machine Learning, 24: 123-140.

[Chen95] Chen, Hsinchun (1995) Machine Learning for information retrieval: Neural networks, Symbolic Learning and Genetic Algorithms. JASIS 46(3) 194-216.

[ChSS98] Chen, Hsinchun, Shankaranarayanan, G, She, L (1998) Machine Learning approach to inductive query by examples: An experiment Using Relevance Feedback, ID3, Genetic Algorithms, and Simulated Annealing. JASIS, 49(8) 693-705.

[Davi91] Davis, L. in Handbook of Genetic Algorithms; Davis, L. (ed.); Van Nostrand Reinhold: New York, 1991

[FrSc96] Freund Y, Schapire R,E. (1996) A decision theoretic generalization of on-line learning and an application to boosting ("http:/www.research.att.com/orgs/ssr/people/schapire")

[Gord88] Gordon, M. (1988) Probabilistic and genetic algorithms for document retrieval. CACM 31(10), 1208-1218.

[Gold89] Goldberg, D.E. Genetic Algorithms in Search Optimization & Machine Learning. Addison-Wesley, 1989.

[Holl75] Holland, J.H. Adaptation in Natural and Artificial Systems. University of Michigan Press: Ann Arbor, MI, 1975.

[Kwok96] Kwok, K.L. (1996) A new method of weighting query terms for ad-hoc retrieval. In: Proc. 19th Ann. Intl. ACM SIGIR Conf. On R&D in IR. Frei, H.P., Harman, D. Schauble, P. & Wilkinson, R. Aug 18-22, 1996. Pp.187-195.

[KwCh98] Kwok, K.L. & M. Chan (1998) Improving two-stage ad-hoc retrieval for short queries. In: Proc. 21st Ann. Intl. ACM SIGIR Conf. On R&D in IR. Croft, W.B, Moffat, A, van Rijsbergen, C.J, Wilkinson, R & Zobel, J. Aug 24-28, 1998. Pp.250-256.

[Mitc96] Mitchell, M. An Introduction to Genetic Algorithms. MIT Press, Cambridge, MA, 1996.

[Quin96] Quinlan, J.R (1996) Boosting, Bagging and C4.5, In Proceedings, Fourteenth National Conference on Artificial Intelligence, 1996.

[SiMB97] Singhal, A, Mitra, M & Buckley C (1997). Learning routing queries in a query zone. In: Proc. of 20th Ann. Intl. ACM-SIGIR Conf. On R&D in IR. Belkin, N.J, Narasimhalu, D, & Willett, P (eds). pp.25-32.

[StCM95] Strzalkowski, T, Carballo, J.P & Marinescu, M (1995). Natural language information retrieval: TREC-3 report. In: Overview of the Third Text REtrieval Conference (TREC-3). Harman, D.K. (Ed.). NIST Special Publication 500-225, pp.39-53.

[WRBJJ98] Walker, S., Robertson, S.E. & Boughanem, M., Jones, G.J.F & Sparck Jones, K (1998). Okapi at TREC-6 automatic ad hoc, VLC, routing, filtering and QSDR. The Sixth Text REtrieval Conference (TREC-6). Voorhees E.M & Harman, D.K. (Eds.). NIST Special Publication 500-240, pp.125-136.

[XuCr96] Xu, J. & Croft, W.B. (1996). Query expansion using local and global document analysis. In: Proc. 19th Ann. Intl. ACM SIGIR Conf. On R&D in IR. Aug 18-22, 1996. Frei, H.P., Harman, D. Schauble, P. & Wilkinson, R. pp.4-11.

Method	Training data	Term selection	Avg. number of terms	Training method	Retrieval
(not1)	none	All terms from query subject to Zipf cutoff [3,10000]	32	Self training based on statistics	Full ap89 and ap90
(not2)	none	same	same	same	same
(pir1)	All relevant	Term expansion ~250	220	Pircs network	same (only for queries with training docs)
(pir2)	same	Term expansion ~60	60	same	same
(gt1)	Half for term selection Other half for training	Ga selects From 300 terms	90	Ga selects terms for pircs	same
(gw1)	same	Term expansion ~100	80	Ga adjusts weights for pircs	same
(gw2)	same	Term expansion ~100	80	same	same
(ga)	same	~300 terms 100 doubles triples and quadruples	410	Genetic algorithm	1500 documents retrieved by pir1
(bp)	same	same	same	backprop	same

Table 10: Summary of Nine Retrieval Methods for Routing and Batch Filtering

Experiments in Spoken Document Retrieval at DERA-SRU

P. Nowell
Speech Research Unit
DERA Malvern
St Andrews Road
Malvern
Worcs, WR14 3PS

Introduction

A small amount of internal funding allowed DERA-SRU to participate in the TREC-7 SDR evaluations for the first time this year. Since we had almost no experience of entering this or related NIST evaluations (e.g. ARPA HUB-4 LVCSR) there was a rather steep learning curve along with intense development of the experimental infrastructure. The intention was to generate a base for future participation and to build upon this using experience gained from related work on topic spotting. To this end, a straightforward (i.e. non-optimised) speech recogniser was used to generate transcripts and retrieval was performed using the *okapi* [6,9] search engine. Previous work on topic spotting [7] suggested that term expansion using a semantic network (in this case *wordnet* [2,3]) might be useful. This hypothesis appeared to be supported by preliminary work on TREC-6 SDR data which yielded text (i.e. R1) results that were comparable with the best achieved elsewhere.

Speech Recognition

Two sets of speech recognition transcripts were generated. The first set (S1) was generated quickly using available acoustic and language models with aim of developing and validating the necessary infrastructure. Subsequently a second set of transcripts (S2) was generated using the same acoustic and language models but with the recognition process beginning to be tailored to the task.

The first set of speech recognition transcripts (S1) was generated using a continuous large vocabulary speech recogniser constructed using pre-existing (i.e. not Hub4 task specific) acoustic and language models. The language model consisted of a 50000 word vocabulary and 500 word class clustered bigram model which had been trained using the North American News Transcripts (NANT) corpus. A single set of 2548 full-bandwidth (i.e. 8kHz) 8 mixture component triphone models were used along with 45 single mixture component fast-match monophone models. The acoustic models had been trained using the SI284 Wall Street Journal corpus.

The speech recogniser used a 25 channel mel-scale filterbank from which 12 cosine terms (C1 ... C12) and an energy value were computed. The inclusion of delta and delta-delta gave a 39 element feature vector. No online adaptive noise masking or channel normalisation algorithms were used. A summary of the official NIST results for S1 are given in table 1.

	Corr	Sub	Del	Ins	Err	S.Err
Sum/Avg	39.3	47.4	13.3	5.6	66.4	99.8

Table 1; NIST results for S1 speech recognition transcripts

In light of the S1 results, it was decided to generate a second set of recognition transcripts. A public-domain copy of the CMU speech segmenter (CMUSeg_0.4) was obtained from NIST. The scripts and code were modified slightly to handle files containing speech from a single focus group and occasional arithmetic underflows. Otherwise, the scripts were used as given with the default parameter settings (i.e. those used by CMU for the 1996 Hub-4 evaluations).

The speech was then segmented prior to recognition into focus groups F0 (full-bandwidth) and F2 (telephone bandwidth). Each segment was then recognised separately using either full-bandwidth or telephone bandwidth models (again trained using SI284 Wall Street Journal) and the recognition transcripts concatenated. Online adaptive noise masking and channel normalisation were also used. N-best recognition results were generated (using a depth cut-off of 20) and these were then rescored using a trigram language model also trained using the NANT corpus. The trigram rescorer generated a further set of N-best lattices of which only the first choice were used in subsequent retrieval experiments. The official NIST results for S2 are summarised in Table 2.

	Corr	Sub	Del	Ins	Err	S.Err
Sum/Avg	47.3	44.8	7.9	8.8	61.5	99.8

Table 2; NIST results for S2 speech recognition transcripts

The average word error rate is still high at 61% due mainly, we believe, to the use of non-task-specific acoustic and language models along with relatively tight pruning thresholds. Given time these results could be improved significantly by developing optimised models and relaxing pruning constraints to more normal levels. However we do not believe that optimising recognition performance should be a major aim for these particular evaluations.

Both recognition runs were carried out on a bank of 200Mhz Pentium Pros with either 64Mb or 128Mb of local memory. In each case the test data was divided up and assigned to one of seventeen processing units. Unfortunately timing information was not kept for the first recognition runs although it is estimated that recognition would have taken around 10 times real-time on a single CPU. The second recognition was timed and in this case the recogniser ran at approximately 22 times real-time on each CPU.

Information Retrieval

During the course of early SRU work on topic spotting it was proposed that a semantic network be constructed and used to generate keywords from a topic descriptor or vice-versa [5]. Unfortunately, due to funding constraints, it was not possible to follow this line of research further at that time. It was therefore decided

that this proposal would be investigated further under the realm of information retrieval. Term expansion using *wordnet* has also been briefly explored in previous TREC evaluations [4,5,8].

The text retrieval test set contains 23 queries, one of which is shown in figure 1. The function of the retrieval engine is to take such queries and generated a ranked list of up to 1000 matching section. Note however the mismatch between the query, which requests a list of cities, and the evaluation which requires a ranked list of episode sections.

```
<num> Number: 55

<desc> Description:
What cities other than Washington D.C. has the First Lady
visited on official business (i.e., accompanying the President
or addressing audiences/attending events)?
```

Figure 1; Example of a spoken document retrieval (SDR) query

The query text is syntactically tagged and keywords are selected on the basis of their part-of-speech (POS) tags. This largely avoids the need for an explicit stop-list as well as helping to reduce the amount of over-generation during term-expansion.

The syntactic tagger, known as LTPOS [12], contains three major components: a tokeniser, a morphological classifier and a morphological disambiguator. The tokeniser segments the input string into words and sentences which are then classified according to a range of morpho-syntatic features (number, case, gender, etc.). Each feature set is unambiguously mapped to one or more part-of-speech (POS) tags. Unknown words are catered for by a rule set which attempts to guess the POS tag(s). Finally, words that have been assigned more than one POS tag (e.g. books' which can be a noun or a verb) are probabilistically disambiguated using a pre-trained Hidden Markov Model. Accuracy on known words is reported to be 96-98% and on unknown words 88-92%.

Noun and verb groups are also identified by means of a syntactic chunker or partial parser. The parser uses the POS information provided by the tagger and mildly context-sensitive grammars to detect syntactic boundaries. The output consists of simple noun groups (e.g. [[Washington D.C]]) and verb groups (e.g. ((have occurred))).

```
What_WP [[ cities_NNS  ]]other_JJ than_IN [[ Washington_NNP
D.C._NNP ]](( has_VBZ  ))[[ the_DT First_NNP Lady_NNP  ]](( 
visited_VBD ))on_IN  [[ official_JJ business_NN
]](_(i.e._FW,_,accompanying_VBG [[ the_DT President_NNP  ]]
or_CC addressing_VBG[[ audiences_NNS/_NN ]]attending_VBG
[[ events_NNS ]])_)?_.
```

Figure 2; Example of typical output from the syntactic tagger / chunker

The tags are used to extract a set of keywords and keyphrases for the retrieval engine. This is achieved by simply extracting words with tags that are likely to be discriminative (In these experiments JJ, NN, VB[DGNP], RB, PRP, MD and WRB were

used). A small ad-hoc stop list, containing just under 20 words, was also used to remove mostly common verbs such as 'are', be', 'do', 'get', have', etc.

The output from the tagger is also processed to extract compound nouns and adjectival phrases. Where phrases contain three or more words the words are successively removed from the left hand side to yield progressively shorter sub-phrases (e.g. 'military air crash' would also yield 'air crash'). Such a simple process however does occasionally generate erroneous sub-phrases. Figure 3 shows the basic keywords and phrases extracted from the query shown in figure 1.

```
<keywords>
cities Washington D.C. First Lady visited official business
accompanying President addressing audiences attending events

<phrases> Compound Nouns:
Washington D.C.
First Lady

<phrases> Adjectival Phrases:
official business
```

Figure 3; Example of keywords and phrases extracted from a tagged query

Term expansion is performed using *wordnet*, a semantic network originally developed for testing psycholinguistic theories of human lexical memory [2,3]. This program contains a network of nouns, verbs, adjectives and adverbs which is organised into synonym sets representing different underlying lexical concepts. Entering a single word produces a list of related words and phrases at various levels of abstraction. *Wordnet* has been used in the TREC-6 but no details appear to have been published. A related concept of semantic forests [10] has also been used to infer topic categories from keywords.

A major problem in any term-expansion scheme is one of over generalisation. This problem has been reduced to some extent due to knowledge of lexical POS tags. Function words are ignored and ambiguous keywords (such as 'issues' which can be a noun or a verb) are generally expanded in the correct manner. Even so, there are still a large number of options by which nouns, verbs and adjectives may be expanded. A brief examination on the previous year's data showed that most options resulted in overgeneration and/or the production of words that were unlikely to have any beneficial effect on retrieval. Only three options were used for keyword expansion, these being the senses of nouns, senses of verbs and hypenyms of nouns.

Furthermore, only the most frequent sense of any word was used in the hypenym expansion and terms were only taken from the top-most (i.e. least abstract) expansion. Given these constraints and the tagged query in figure 3, the following expansion terms were generated.

<sensn> Senses	<hypen> Hypenyms	<sensv> Senses
city	municipality	have
metropolis	national capital	have got
urban center	federal district	hold
lady	rank	visit
business	woman	see
concern	adult female	attach to
business concern	enterprise	attend
business organization	corporate executive	accompany
audience	business executive	come with
event	gathering	go with
	assemblage	address
		speak to
		turn to
		go to

Table 3; Wordnet expansion of a tagged query

Text retrieval was performed using Okapi, a probabilistic retrieval engine developed by City University and the Polytechnic of Central London. Okapi has been used extensively at previous TREC evaluations by City University as well as other participants [6,9]

All texts were first indexed using default parameter settings i.e. strong stemming and an empty GSL file. A GSL file can be used to specify terms that are dealt in a special way by the indexing processing such as stop terms or words that should be indexed as a single item. Searches were performed using the standard BM25 weighting function reported in TREC-3 [6] with the default parameter settings as used by City in TREC-6 [9].

An off-line retrieval engine was developed around the Basic Search System (BSS) library. This engine is similar to the test-engine provided with the *okapi-pack* distribution but has been modified to enable searches for co-adjacent words (e.g. 'First Lady'). The manner in which multi-word search items are handled is however different from that used by the interactive version of Okapi. In the former case the words must be strictly co-adjacent whereas in the latter the words only have to occur in the same sentence or paragraph. Otherwise the search engine is completely standard with the default weighting functions being used for both single word andmulti-word items.

Three retrieval experiments were run, the first using keywords only (KW), the second keywords and phrases (KWP) and the last using keywords, keyphrases and *wordnet* expansions (KWPE). As it was only possible to submit one set of retrieval results for official scoring the latter (KWPE) were submitted.

Official TREC-7 SDR Results

A summary of the official retrieval results for KWPE (keywords, phrases andwordnet expansion) is presented in figure 4. As described previously, two sets of recognition transcripts were generated.

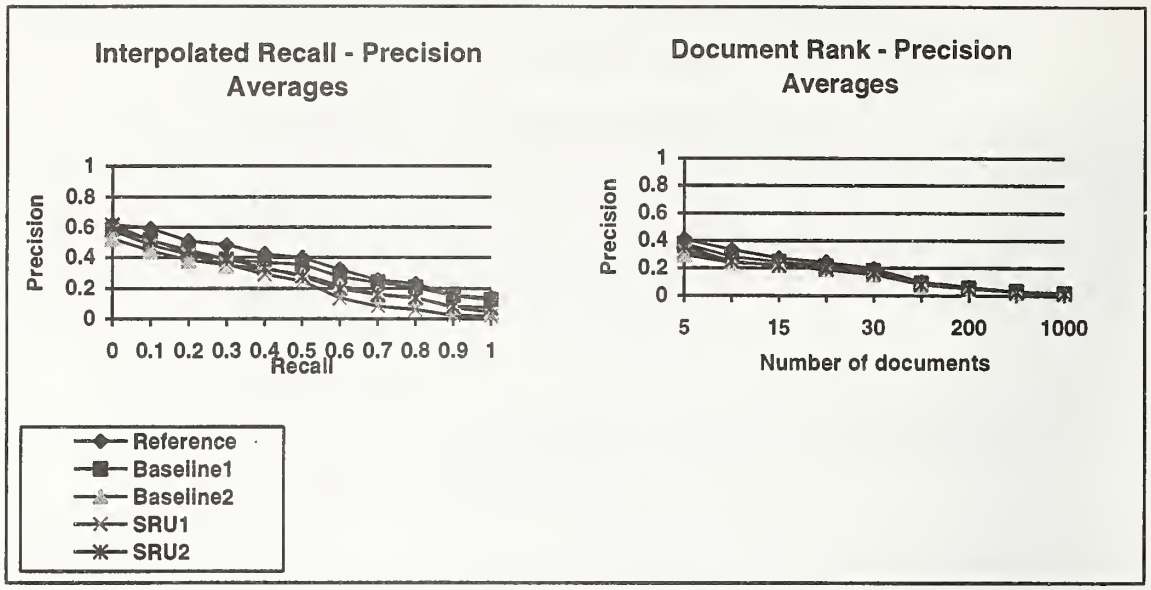


Figure 4; Official TREC-7 SDR summary results

The results show that there does not appear to be a great difference in retrieval performance even though the word error rate ranges from 0% to 66%. It is also interesting that results using SRU2 appear to be slightly better than with the second baseline (B2) although the word error rate is substantially higher.

The following graph compares results on the reference transcripts and therefore shows differences due to the retrieval strategies used by the various sites. Results using recogniser transcripts are likely to be similar but with lower absoluteprecisions.

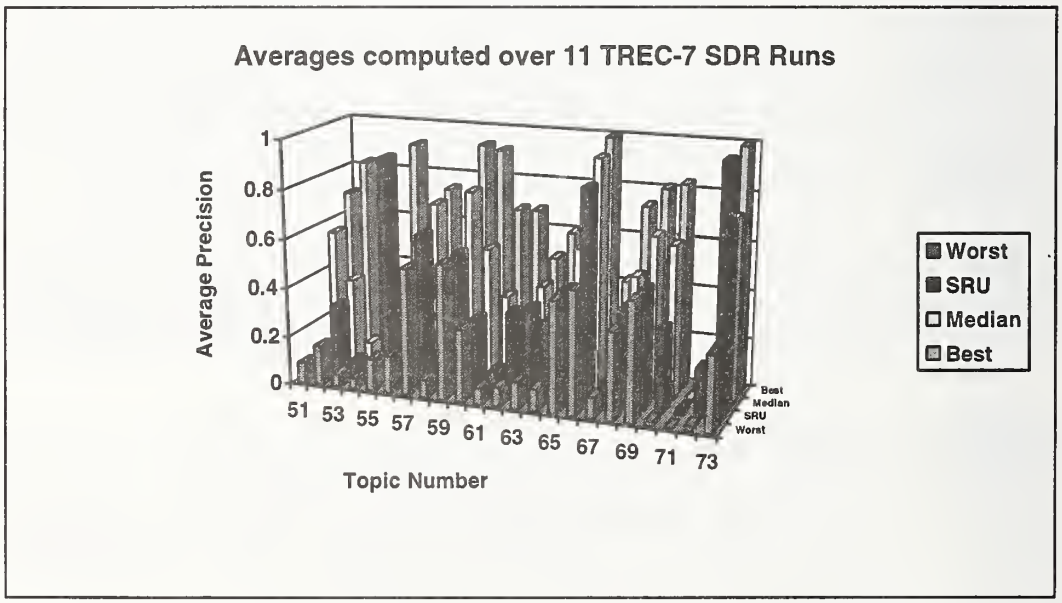


Figure 5; Official TREC-7 SDR results for each topic

Of the 23 topics three have the highest precision, two are above the median, twelve are below the median and six have the lowest precision. Topic 55 is particularly interesting since in this our results have the highest precision and are also significantly higher than the median. One would expect that this is due to *wordnet* expansion yielding one or more particularly useful keywords or phrases. Further investigation is required to determine whether this is indeed the case.

Examination of the queries revealed that only two words were unknown to the speech recogniser, these being 'paparazzi' (topic 60) and ' Trie' - a Chinese name (topic 64). Paparazzi occurs frequently in the 'perfect' (LTT) transcripts whereas Trie does not appear at all. Unsurprisingly our retrieval results are poor on topic 60 and term expansion is unable to help since 'paparazzi' is also unknown to *wordnet.*

Unofficial TREC-7 SDR Results

Figure 6 shows the performance of different retrieval strategies according to the condition and word error rate (WER). Labels along the x-axis refer to the source of the transcripts and associated word error rate (R1=Reference, CU=Cambridge University, DG=Dragon, B1,B2=Baselines 1 and 2, DS=DERASRU S2 and S1). The key labels refer to the source of the retrieval results, these being University of Massachusetts, Sheffield, Cambridge University, University of Maryland and DERASRU. The last key Simple' shows the results that we would have obtained by simply using keywords i.e. omitting keyphrases and term expansion (results using keyphrases are virtually identical).

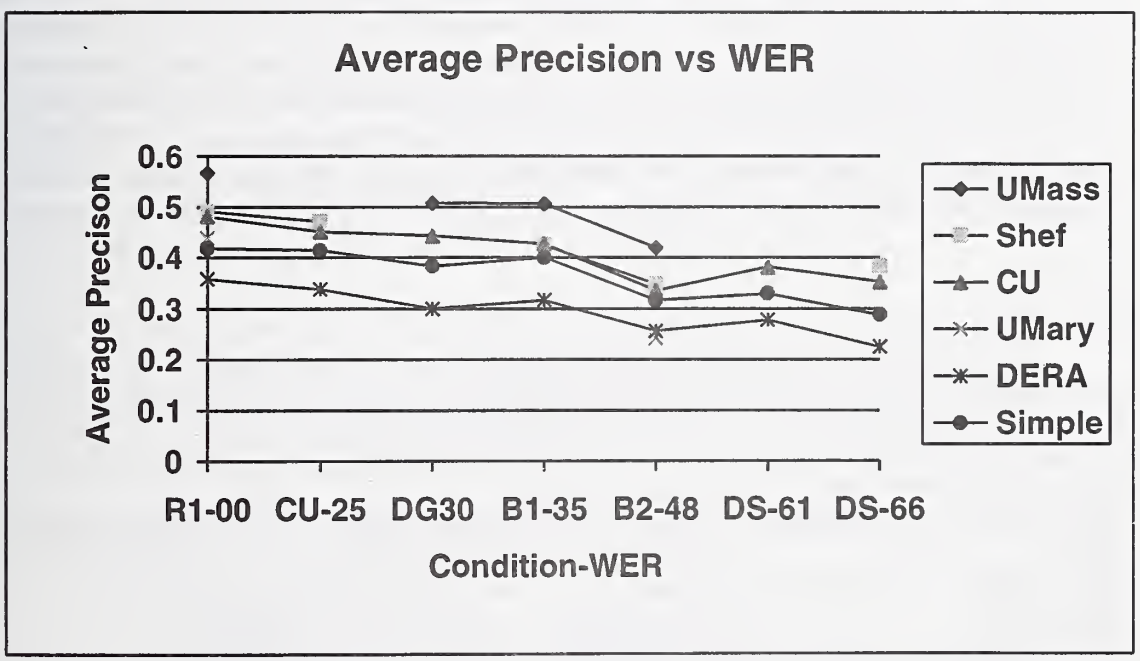


Figure 6; Unofficial comparative TREC-7 SDR results

The graph, which only represents those retrieval results known at the time of writing this report (Oct. 1998), shows that UMass appears to be better and DERA unfortunately worse than average. It is interesting and also disappointing that the

simple keyword-only approach works consistently better than the more complex approach involving term expansion. Furthermore, retrieval performance using DS-61 is consistently higher than with B2-48 suggesting that not all errorsare created equal.

Taking figures 5 and 6 together it seems that term expansion using *wordnet* can occasionally be of great benefit but more often than not leads to a degradation in performance. Further investigation is required to uncover the reasons for the performance shortfall either using simple keywords or keywords, phrases and term expansion.

Summary

Although this is the first time that DERA has participated in such evaluations it is nevertheless disappointing that the current performance is less than that of more experienced participants. Given time, the performance of the speech recogniser could be easily improved through the use of task-specific acoustic and language models. There is also scope for improvement of the information retrieval component as shown by the keyword only results. It is however somewhat surprising that term-expansion should lead to such a large and consistent decrease in performance. However, now that the basic infrastructure is in place the performance of both components should improve quite rapidly.

Of more interest for the future, are the application of techniques that address the specific problems of spoken document retrieval. When DERA first entered the evaluation the intention was to try and make use of phoneme based techniques developed for topic spotting. It is believed that these will be of benefit where the query contains out-of-vocabulary words and / or the recogniser incorrectly recognises significant words. Time constraints meant that it was not possible to carry out these investigations this time around. It is hoped that these and other techniques will be investigated as part of future evaluations although funding sources have yet to be identified.

References

[1] S.E. Robertson and K. Sparck Jones, Relevance weighting of search terms, Journal of the American Society for Information Science 27, May-June 1976, p129-146.
[2] G.A. Miller, Wordnet: A dictionary browser, Proc. of the first conference of the UW centre for the New Oxford dictionary, Univ. of Waterloo, Canada, 1985.
[3] G.A. Miller et al. Introduction to wordnet: an on-line lexical database, Princeton University, Aug. 1993.
[4] E. Voorhees and Y.W. Hou, Vector Expansion in a Large Collection, NIST special publication 500-207, 1993.
[5] E. Voorhees, On Expanding Query Vectors with Lexically Related Words, NIST special publication 500-215, 1994.
[6] S.E. Robertson et al, Okapi at TREC-3, NIST special publication 500-236, 1995.
[7] P. Nowell, Topic Spotting Progress Report, DRA Memorandum DRA/CIS(SE1)/374/04/PR1/1.0, March 1996.
[8] A. Smeaton et al, TREC-4 Experiments at Dublin City University:Thresholding Posting Lists, Query Expansion with Wordnet and POS Tagging of Spanish, NIST special publication 500-236, 1996.

[9] S. Walker et al, Okapi at TREC-6: Automatic ad hoc, VLC, routing, filtering and QSDR, Proceedings of the Sixth Text Retrieval Conference (TREC-6), NIST special publication 500-240, 1998.
[10] P. Schone et al, Text Retrieval via Semantic Forests, Proceedings of the Sixth Text Retrieval Conference (TREC-6), NIST special publication 500-240, 1998.
[11] E.M. Vorhees and D. Harman, Overview of the Sixth Text Retrieval Conference (TREC-6), NIST special publication 500-240, 1998.
[12] University of Edinburgh, Language Technology Group, 2Buccleuch Place, Edinburgh, EH8 9LW, (http://www.ltg.ed.ac.uk)

Addendum

Following receipt of the official SDR submission results it has been possible to devote some time to investigating areas where the retrieval approach could be improved. This has involved analysis of performance on the test data as well as testing of alternative stop-lists and parameter settings. There is therefore an element of training on the test data. However, the intention is that any modifications should be consistent with those that might had been made given sufficient time and experience of previous evaluations. Incremental results using keywords only on hand-transcripts (R1) are shown in table 4.

Previously it was described how keywords were selected based upon POS tags and a small stop-list containing less than 20 words. Replacing this small list by the standard 'van Rijsbergen' stop-list gives a small improvement when applied to the queries only and a larger improvement when also applied prior to database indexing.

Examination of the queries and retrieval output revealed a problem with the handling of abbreviations. One of the queries contains the single term 'U. S.' which is represented in all transcripts as two separate words, i.e. 'U. S. '. Simple attempts at concatenating such words leads to problems since the *okapi* pre-processor substitutes 'us' which is in the stop-list and therefore unindexed. Removing 'us' from the stop-list would of course create other problems. Rewriting all abbreviations in transcripts and queries to the form 'UxxSxx, DxxCxx etc.' overcomes these problems and gives a small increase in performance.

It has been shown previously (e.g. [11]) that using templates of the form 'What data / information is availableon ....' to prune non-topic-descriptive words from queries is advantageous. Table 4 shows that query pruning also gives a relatively large increase in performance on this years data.

Alternative parameters settings for the BM25 weighting function [9] have also been tested but these were not found to give a clear advantage.

Average Precision (R1)	Modification
0.4179	Official / baseline results
0.4216	Stoplist applied to queries only
0.4334	Stop-list applied to queries and database
0.4349	Proper treatment of abbreviations
0.4516	Query pruning using templates

Table 4; Effects of modifications on average precision

The above changes have also been incorporated into the full system employing keywords, keyphrases and *wordnet* expansion which was submitted to NIST. In addition weighting factors [4,5,8] have been added to each term with values of 1.0 for keywords and keyphrases and 0.5 for terms arising from *wordnet* expansion. The effect is de-weight expanded terms which may be less relevant those obtained from the original query.

Figure 7 shows comparative results on the standard R1, B1, B2, S1 and S2 transcripts. No cross-recogniser results have been generated at this time (October 1998).

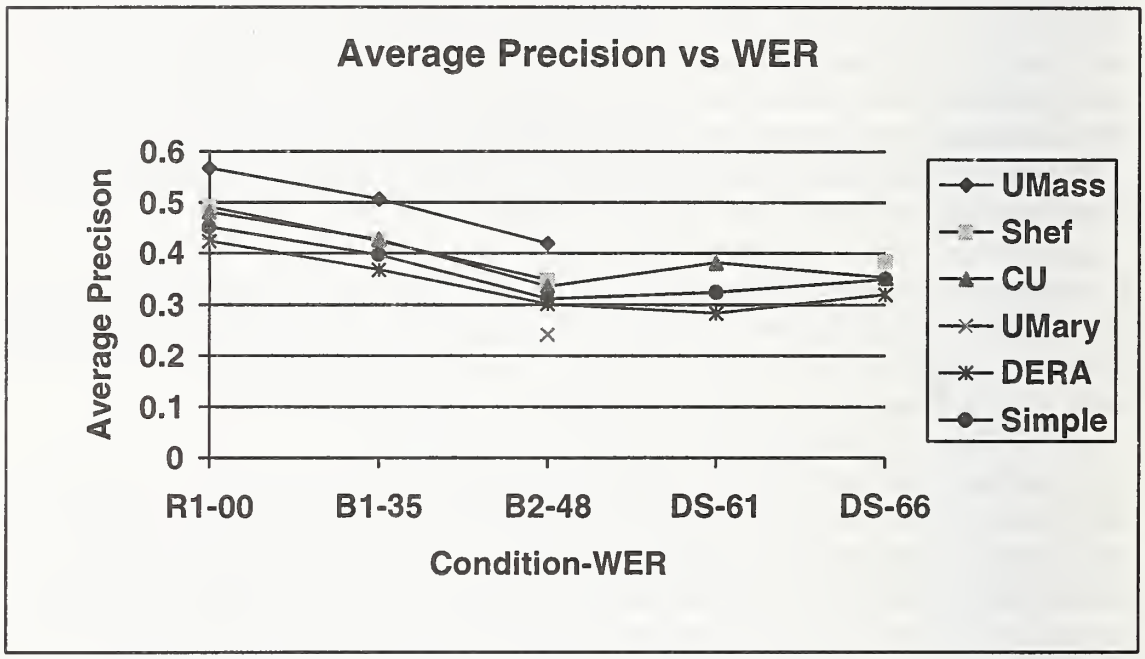


Figure 7; Second iteration unofficial TREC-7 SDR results

Keyword only results (Simple) are improved on R1-00 and DS-66 and are closeto the median performance level. There is however still a small performanceshortfall which shows that there is more work to be done. Although term-expansion still leads to a fall in performance the average precision has been greatly increased. Both sets of results of results could undoubtedly be improved further given time and resources.

Informative term selection for automatic query expansion

Claudio Carpineto
Fondazione Ugo Bordoni, Rome
Italy
carpinet@fub.it

Renato De Mori
University of Avignon
France
renato.demori@
lia.univ-avignon.fr

Giovanni Romano
Fondazione Ugo Bordoni, Rome
Italy
romano@fub.it

Abstract Techniques for query expansion from top retrieved documents have been recently used by many groups at TREC, often on a purely empirical ground. In this paper we present a novel method for ranking and weighting expansion terms. The method is based on the concept of relative entropy, or Kullback-Lieber distance, developed in Information Theory, from which we derive a computationally simple and theoretically justified formula to assign scores to candidate expansion terms. This method has been incorporated into a comprehensive prototype ranking system, tested in the *ad hoc* track of TREC-7. The system's overall performance was comparable to median performance of TREC-7 participants, wich is quite good considering that we are new to TREC and that we used unsophisticated indexing and weighting techniques. More focused experiments showed that the use of an information-theoretic component for query expansion significantly improved mean retrieval effectiveness over unexpanded query, yielding performance gains as high as 14% (for non interpolated average precision), while a per-query analysis suggested that queries that are neither too difficult nor too easy can be more easily improved upon.

1. Introduction

Automatic query expansion from top retrieved documents is a well known retrieval strategy with clear potentials for addressing both thoretical limitations of information retrieval systems, such as the incapability of recovering from word mismatch between queries and documents, and practical deficiencies related to their usage in operational environments, such as the paucity of user-supplied query terms. While these potentials did not, historically, turn into actual better retrieval performance, due to losses in precision being higher than gains in recall, this is not the case in the TREC environment. The combination of better initial retrieval and the collection characteristics of TREC (longer and more numerous relevant documents than in the small test collections) makes this approach very successful (Buckley et al., 1995). According to Donna Harman (Harman, 1998), "by TREC-6 almost all groups were using variations on expanding queries using information from the top retrieved documents (pseudo-relevance feedback)".

The growing interest in this technique calls for a better understanding of its foundations and a more careful evaluation of its experimental design choices. The primary concern here is to develop well founded methodologies for ranking and weighing expansion terms, because most of the approaches that have been proposed leave something to be desired in terms of theoretical justification. Complementary, as virtually any proposed approach to query expansion relies on a number of parameters, it is important to study which factors are critical for good overall performance. Donna Harman (Harman 98), for instance, pointed out that, in the context of TREC, while there is general system convergence on some of the many parameters of query expansion needed for success, these still need to be tested by systems adopting these techniques.

Our participation in TREC-7, in the *Ad-hoc* track, was motivated by a desire to contribute to explore these issues. In particular, we were primarily interested in evaluating the retrieval effectiveness of a novel framework for query expansion based on ideas from Information Theory (Cover and Thomas, 1991). Additionally, we were concerned with evaluating the effectiveness of query expansion from top retrieved documents as the difficulty of the query varies.

2. Using information-theoretic "relative entropy" to select expansion terms

In order to discriminate between good expansion terms and poor expansion term it is convenient to assume that the differences between the distribution of terms in the overall document collection and the distribution of the same terms in a set of relevant documents are related to semantic factors. More precisely, we expect that good terms will occur with a higher frequency in relevant documents than in the whole collection, and poor terms will occur with the same frequency (randomly) in both.

To implement the view that the difference in the distribution of terms will reveal their likely relevance, we

need well founded and practical ways of comparing different distributions and assigning scores to terms based on such a comparison. Our approach is based on the relative entropy, or *Kullback-Leibler* distance (KD), between two distributions A and B. The relative entropy is customarily used to measure the extent of the error that we make by using A as a substitute for B. In our application we do not have a right distribution to approximate, thus it is more suitable to consider the sum of the difference between A and B and the difference between B and A. In the query expansion setting, the definition of this derived, symmetric distance becomes:

Let C be the set of all documents in the collection
Let V be the vocabulary of all the terms.
Let $t \in V$ be a word.
Let R be the set of top retrieved documents relative to a query.
Let v(R) be the vocabulary of all the terms in R.
Let $p_C(t)$ be the probability of $t \in V$ estimated using the whole collection. Let D_C be the corresponding distribution.
Let $p_R(t)$ be the probability of t estimated from the corpus R. Let D_R be the corresponding distribution. The*Kullback-Leibler* distance between the two distributions D_C and D_R is given by:

$$KD(D_C, D_R) = \sum_i \left\{ \left[p_R(t) - p_C(t) \right] \times \log \frac{p_R(t)}{p_C(t)} \right\} \quad (1)$$

The words to be considered for query refinement are those that mostly contribute to KD. In order to take into account the fact that it is possible that $v(R) \subset V$, a default probability is assumed for $p_R(t)$ when t does not appear in v(R), leading to the following definition:

$$p_R(t) = \begin{cases} \gamma \dfrac{f(t)}{NR} & \textit{if} \quad t \in v(R) \\ \delta\, p_C(t) & \textit{otherwise} \end{cases}$$

where f(t) is the frequency of t in R and *NR* is the number of terms in R. This scheme, in principle, better handles the sparse data problem when R is not sufficiently large.
Since:

$$\sum_{t \in v(R)} f(t) = NR$$

the following relation must hold:

$$\gamma + \delta \sum_{t \in v(R)} p_C(t) = \gamma + \delta A = 1$$

and KD can be rewritten as follows:

$$KD(D_C, D_R) = K_1 + K_2$$

$$K_1 = \sum_{t \in v(R)} \left\{ \left[\gamma \frac{f(t)}{NR} - p_C(t) \right] \log \frac{\gamma \dfrac{f(t)}{NR}}{p_C(t)} \right\}$$

$$K_2 = \sum_{t \notin v(R)} \left\{ \left[\delta\, p_C(t) - p_C(t) \right] \log \frac{\delta\, p_C(t)}{p_C(t)} \right\} =$$

$$= A(\delta - 1) \log \delta$$

As $\delta = \dfrac{1-\gamma}{A}$

if: $m(\gamma) = \max_{t \notin v(R)} p_C(t) \dfrac{1-\gamma-A}{A} \log \dfrac{1-\gamma}{A}$

we may impose that selected terms for query refinement should respect the condition:

$$\left\{ \left[\gamma \frac{f(t)}{NR} - p_C(t) \right] \log \frac{\gamma \dfrac{f(t)}{NR}}{p_C(t)} \right\} > m(\gamma). \quad (2)$$

In other words, condition (2) states that the contribution of any selected term to KD should be greater than the contribution of every term not in v(R). As the left-hand side grows with γ while the right-hand side decreases with γ, it is always possible to find a value of $\gamma > 0$ such that the selected terms for query refinement do not contain any element not in v(S). This finding does not solve the parameter estimation problem but it supports using v(S) as an approximation of the set of candidate expansion terms. As γ does not influence the ranking of $t \in v(R)$, the following score can be used for ranking:

$$\sigma(t) = \left[\gamma \frac{f(t)}{NR} - p_C(t) \right] \log \frac{\gamma \dfrac{f(t)}{NR}}{p_C(t)} \quad (3)$$

with the first terms selected for query expansion. The same score can also be used for weighting the selected terms in the expanded query, in which case the result depends on the chosen value of γ.
For actual use in a retrieval system, a number of parameters must be chosen. This aspect is dealt with in the next section.

3. Description of our complete ranking methodology

1. *Text segmentation.* Our system first identified the

individual terms occurring in a text collection, ignoring punctuation and case.

2. *Word stemming*. To extract word-stem forms, we used a very large *trie*-structured morphological lexicon for English (Karp et al, 1992), that contains the standard inflections for nouns (singular, plural, singular genitive, plural genitive), verbs (infinitive, third person singular, past tense, past participle, progressive form), adjectives (base, comparative, superlative).

3. *Stop wording*. We used a stop list, contained in the CACM dataset, to delete from the texts common function (root) words. In addition, we removed the terms that appeared in more than 100,000 and less than 3 documents.

4. *Document weighting*. We assigned weights to the terms in each document by the classical *tf·idf* scheme.

5.*Weighting of unexpanded query:* To weigh terms in unexpanded query we used the function (log *tf*)·*idf*, where *tf* is the term frequency in the query and *idf* is the inverse document frequency.

6. *Document ranking with unexpanded query*: We computed an intermediate (or primary) document ranking by taking the inner product (with cosine normalization) between the document vectors and the unexpanded query vector.

7. *Expansion term ranking*: We used as set of candidate expansion terms the complete text of the first R retrieved documents. The candidates were ranked by using expression (3) with γ=1, which amounts to restricting the candidate set to the terms contained in R, and then the first E of them were chosen. To estimate $p_C(t)$, we used the ratio between the frequency of t in C and the number of terms in C, analogously to $p_R(t)$; in order to estimate $p_R(t)$ for the case when $t \in v(R)$, we used a more sophisticated function than f(t)/NR, taking also into account the likely degree of relevance of the documents retrieved in the initial run:

$$\frac{\sum_{d} f(t) \times score_d}{\sum_{t}\sum_{d} f(t) \times score_d} \qquad (4)$$

The argument made in Section 2 holds also for this new estimation function. It is also worth noting that the system selects only those terms with a higher estimated probability in the first retrieved documents than in the entire collection (i.e., such that the first factor in expression (3) is greater than zero); in fact, in our experiments the top terms always met this condition.

8. *Weighting of expanded query*. Expressed in vector space notation, $Q_{exp} = Q_{unexp} + \text{Smooth-Fn}\ (T_{exp})$, where T_{exp} contains the expansion terms weighted with their normalised σ-score, and Smooth-Fn is a smoothing function. The normalization was performed by dividing each score by the maximum score; the use of a smoothing function was due to the presence of a large fraction of suggested terms with very low scores. The unexpanded query was also normalized by the maximum possible weight.

9. *Document ranking with expanded query*: The final document ranking was computed by taking the inner product (with cosine normalization) between the document vectors and the expanded query vector.

The choice of the three parameters involved in our expansion method (number of pseudo relevant documents R, number of expansion terms E, and smoothing function Smooth-Fn) was based on earlier results obtained in past TREC conferences and on some preliminary experiments that we performed on the TREC-6 data. We selected two parameter combinations, (R=5, E=30, K=power 0.75) and (D=5, E=60, K=power 0.5), and computed the two corresponding document rankings (submitted as run "fub98a" and run "fub98b", respectively).

4. Computational efficiency

The whole system was implemented in Common Lisp and runs on a SUN-Ultra workstation. The time taken to index the whole collection (several hours) and to compute the primary ranking for each query (several seconds) was relatively large because I/O procedures were not optimized. Nonetheless, the time necessary to perform solely query expansion was negligible. As the collection frequencies were stored in the inverted file built from the set of documents, the computation of $p_C(t)$ was straightforward; to find $p_R(t)$, through expression (4), it was faster to perform one pass through the first retrieved documents. In fact, information-theoretic query expansion is practical even for interactive applications, provided that it is used in conjunction with an efficient ranking system.

5. Performance of expanded query versus unexpanded query

As our main goal was to evaluate the effectiveness of the information-theoretic expansion stage, we compared the performance of document ranking with unexpanded query

with that of the two document rankings with expanded query. The results are shown in Figure 1 and Table 1, using the standard TREC performance evaluation measures.

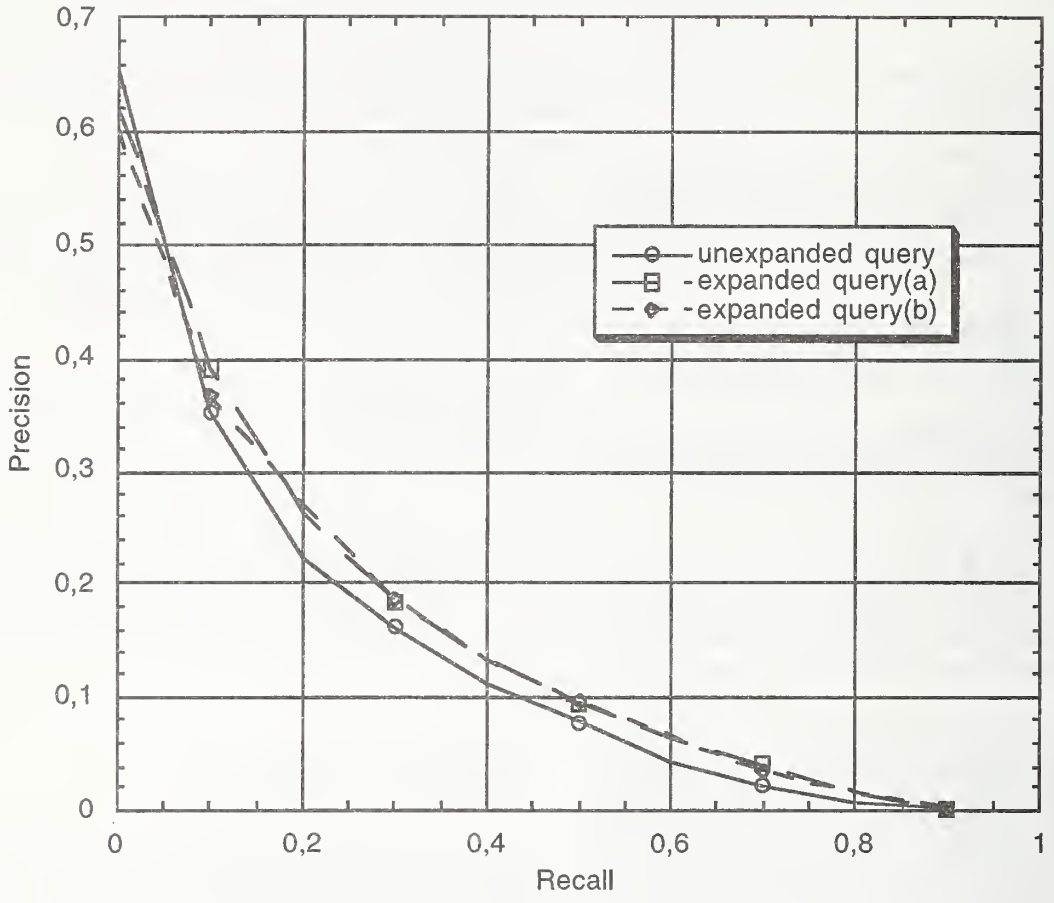


Figure 1. Comparative performance of ranking with and without query expansion (interpolated recall-precision curve).

It turned out that the two rankings with expanded query achieved very similar retrieval effectiveness, and that both of them had better performance than ranking without expansion at almost all evaluation points, with one main exception at recall=0 in the interpolated recall-precision curve. In particular, when passing from unexpanded query to expanded-query-(a), the overall number of relevant retrieved documents increased by 11.92%, average precision by 14.46%, and R-precision by 8.62%. The performance improvement was therefore apparently consistent in all of these tasks; in fact, the benefit of our expansion scheme was statistically significant in all of these measures.

It is also useful to compare the overall performance of our system with that of the other official runs in the Ad-hoc category. For instance, with respect to the the total number of retrieved relevant documents, our run (fub98a) achieved better than median performance for 17 topics, median performance for 4 topics, and worse than median performance for 29 topics. For the other evaluation measures, the behavior was similar. The overall performance of our system was therefore relatively good on average, especially considering that we are new to TREC, but it was definetely inferior to that of the best TREC systems. This implies that the utilization of the query expansion mechanism, while resulting in a marked improvement over ranking with unexpanded query, was not sufficient alone to compensate for the limited effectiveness of the primary ranking scheme.

In fact, it is the combination of several ingredients that makes systems successful at TREC. The current version of our system performs very conservative stemming,

only uses single word index terms, and employs an unsophisticated document weighting function. By contrast, the most successful TREC systems (e.g., Hawking et al., 1998; Walker et al, 1998) adopt specific document weighting functions that have evolved over the years and best reflect the characteristics of the collection, such as the the Cornell variant of the OKAPI BM25 weighting function (Singhal et al., 1995), and customarily perform some kind of linguistic analysis during the indexing stage to better handle ambiguous or misleading words in the topic formulation, for instance through the extraction of multiple-word concepts. High overall performance is thus the compound effect of many critical choices.

Table 1. Comparative performance of ranking with and without query expansion (single-value evaluation measures).

	unexepanded query	expanded query(a)	expanded query(b)
Retrieved and Relevant	1928	2158	2155
Average Prec.	0.1231	0.1409	0.1390
R-Prec	0.1694	0.1840	0.1818
Prec. at 5	0.3880	0.3840	0.3840
Prec. at 10	0.3380	0.3400	0.3400
Prec. at 15	0.3053	0.3187	0.3067
Prec. at 20	0.2830	0.2940	0.2830
Prec. at 30	0.2373	0.2573	0.2473
Prec. at 100	0.1404	0.1450	0.1416
Prec. at 200	0.0977	0.1064	0.1055
Prec. at 500	0.0594	0.0664	0.0656
Prec. at 1000	0.0386	0.0432	0.0431

6. Performance of query expansion versus query difficulty

The results shown in Table 1 were averaged over the set of queries. It is clear that any query expansion method of this kind may behave very differently depending on the quality of the initial retrieval run. In particular, one might expect that query expansion will work well if the top retrieved documents are good and that it will perform badly if they are poor. For instance, we show in Figure 2 the very good expansion terms obtained for query 364, which had mostly relevant top retrieved documents. By contrast, we show in Figure 3 the poor expansion terms generated for query 364, which had some misleading top retrieved documents concerning "Euro Disney". In the former case the good original performance further improved as a consequence of query expansion, while for the latter query the bad performance of unexpanded query further decreased after query expansion.

To test the hypothesis mentioned above, we studied how the retrieval effectiveness varies as the difficulty of a query changes, where the latter was characterized by the average precision of the initial run relative to the given query (the lower the average precision, the greater the difficulty). The results are shown in Figure 4. Each circle represents one of the 50 queries; if the circle is above (below) the bisecting line, then the performance increased (decreased) when we passed from unexpanded to expanded query. The query difficulty decreases as we move away from the origin.

<num> Number: 364

<title> rabies

<desc> Description: Identify documents discussing cases where rabies have been confirmed and what, if anything, is being done about it.

<narr> Narrative: A relevant document identifies confirmed cases of rabies and may contain actions taken to correct the problem.

	Unexpanded Query		Expanded Query	
1	1.000	RABIES	1.788	RABIES
2	0.311	CONFIRMED	0.326	ANIMAL
3	0.268	IDENTIFY	0.313	VACCINE
4	0.251	DOCUMENT	0.311	CONFIRMED
5	0.170	RELEVANT	0.268	IDENTIFY
6	0.165	CORRECT	0.251	DOCUMENT
7	0.113	DISCUSS	0.215	VACCINATION
8	0.091	ACTION	0.185	HUBERT
9	0.082	PROBLEM	0.182	NASPHV
10			0.178	VETERINARIAN
11			0.170	RELEVANT
12			0.165	CORRECT
13			0.136	RESTRICTION
14			0.130	VETERINARY
15			0.113	DISCUSS
16			0.102	APHIS
17			0.100	NONVETERINARIANS
18			0.097	VACCINATE
19			0.097	SAINT
20			0.095	ST
21			0.094	TAILLE
22			0.091	ACTION
23			0.082	PROBLEM
24			0.075	STOLE
25			0.075	BITE
26			0.069	REVACCINATION
27			0.066	POST-EXPOSURE
28			0.066	CENTURY
29			0.064	DISTRIBUTE
30			0.063	PILGRIMAGE

Figure 2. Unexpanded and expanded weighted terms for TREC7 query 364

<num> Number: 378

<title> euro opposition

<desc> Description: Identify documents that discuss opposition to the introduction of the euro, the European currency.

<narr> Narrative: A relevant document should include the countries or individuals who oppose the use of the euro and the reason(s) for their opposition to its use.

	Unexpanded Query		Expanded Query	
1	1.000	EURO	1.625	EURO
2	0.536	OPPOSITION	0.536	OPPOSITION
3	0.380	DOCUMENT	0.380	DOCUMENT
4	0.263	INTRODUCTION	0.298	DISNEY
5	0.257	RELEVANT	0.263	INTRODUCTION
6	0.223	CURRENCY	0.257	RELEVANT
7	0.219	OPPOSE	0.223	CURRENCY
8	0.203	IDENTIFY	0.219	OPPOSE
9	0.174	INDIVIDUAL	0.203	IDENTIFY
10	0.171	DISCUSS	0.189	ENGINE
11	0.159	REASON	0.174	INDIVIDUAL
12	0.153	EUROPEAN	0.171	DISCUSS
13			0.165	TRUCK
14			0.159	REASON
15			0.158	GRAM
16			0.153	EUROPEAN
17			0.140	STANDARD
18			0.127	II
19			0.126	ARMSTRONG
20			0.121	IVECO-FORD
21			0.105	EXHAUST
22			0.100	PARTICULATES
23			0.095	VISITOR
24			0.090	ATTENDANCE
25			0.081	EU
26			0.075	INJECTOR
27			0.074	PARIS
28			0.074	HALVE
29			0.073	LIKELY
30			0.072	FUEL

Figure 3. Unexpanded and expanded weighted terms for TREC7 query 378

These results are somewhat unexpected, because no clear pattern seems to emerge. The performance improvement does not monotonically grow with easiness of query; indeed, if we split the X axis in intervals and compute the average performance of the queries within each interval, then it is easy to see that performance variation is initially negative, as expected, and then it starts climbing until it reaches a maximum (initial precision of 20-30%), after which it declines and may drop again below zero. In fact, our experiment supports the view that queries with low precision do not carry useful information for improvement, while queries with high initial precision can be hardly further improved upon; as an indication to achieve further mean improvement, one might develop selective policies for query expansion that focus on queries that are neither too difficult nor too easy.

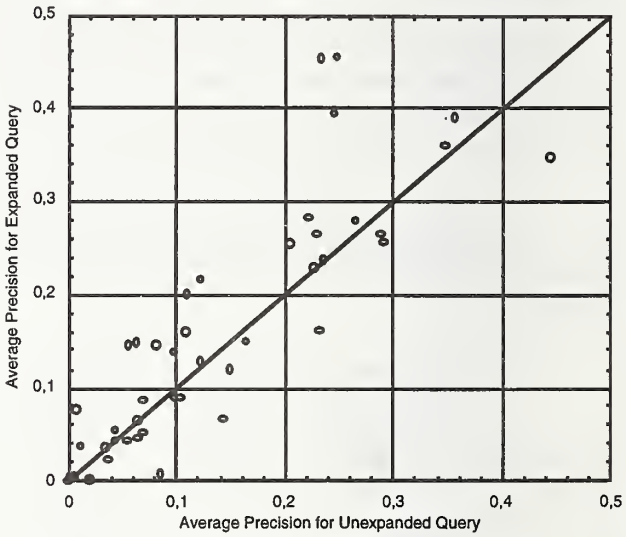


Figure 4. Improvement versus initial query difficulty

7. Current work

We are currently re-implementing the whole indexing stage, which was specifically designed to our own weighting method, to test alternative methods to score documents and queries. In addition, we are experimentally studying the effect that the three main parameters involved in query expansion – namely, how many top documents to use for mining terms, how many terms to select, and how to weight those terms – have on retrieval performance, considering their possible interactions. Finally, as several other researchers have recently reported significant improvement of performance retrieval due to the use of automatic query expansion techniques, especially for the TREC collection, it has become important to evaluate and contrast competing approaches on a more systematic basis. A first step into this somewhat overlooked direction has already been taken elsewhere (Carpineto *et al.*, submitted).

Acknowledgments

This work has been carried out within the framework of an agreement between Telecom Italia and the Fondazione Ugo Bordoni.

References

Buckley, C., Salton, G., Allan, J., and Singhal, A. (1995). Automatic query expansion using SMART: TREC3, *Proceedings of the third Text REtrieval Conference (TREC-3),* NIST Special Publication 500-225.

Carpineto, C., De Mori, R., and Romano, G. (submitted). Automatic query expansion: an information-theoretic method and a comparative evaluation.

Cover, T. M., and Thomas, J. A. (1991). *Elements of Information Theory*. Wiley.

Harman, D. (1998). Overview of the Sixth Text Retrieval Conference (TREC-6). In D. K. Harman, editor, *Proceedings of the Sixth Text Retrieval Conference (TREC-6).* NIST Special Publication 500-240.

Hawking, D., Thistlewaite, P., and Craswell, N. (1998). ANU/ACSys TREC-6 Experiments. In D. K. Harman, editor, *Proceedings of the Sixth Text Retrieval Conference (TREC-6*). NIST Special Publication 500-240

Karp, D., Schabes, Y., Zaidel, M., Egedi, D. (1992). A Freely Available Wide Coverage Morphological Analyzer for English. *Proceedings of the 14th International Conference on Computational Linguistics (COLING '92),* Nantes, France.

Singhal, A., Salton, G., Mitra, M., and Buckley, C. (1995). Document length normalization. Technical report TR95-1529, Cornell University.

Walker, S., Robertson, S. E., Boughanem, M., Jones, G.J.F., and Sparck Jones, K. (1998) Okapi at TREC-6: Automatic ad hoc, VLC, routing, filtering and QSDR. *Proceedings of the sixth Text REtrieval Conference (TREC-6*), NIST Special Publication 500-240.

Document Retrieval Using The MPS Information Server (A Report on the TREC-7 Experiment)

François Schiettecatte
(francois@fsconsult.com)
FS Consulting, Inc.
20709 Delta Drive, Laytonsville, MD, 20882
http://www.fsconsult.com/

1 Introduction

This paper summarizes the results of the work conducted by FS Consulting, Inc. as part of the Seventh Text Retrieval Experiment Conference (TREC-7). We participated in two components of TREC: (1) Category C, running the Ad Hoc experiments, producing two sets of official results (fsclt7m and fsclt7a) and (2) the Very Large Collection (VLC) track, running the entire experiment, producing the required official results as well as a set of unofficial results.

Our long-term research interest is in building information retrieval systems that help users find information to solve real-world problems. Our TREC runs employ two models: for the manual experiments, we have developed a model information systems end-user described more fully in section 5. For the automatic experiments, we use a title-word retrieval model.

2 Overview of FS Consulting TREC-7 Experiments

Our TREC-7 participation centered on two goals: comparing the effectiveness of manually generated queries versus automatically generated queries and evaluating the effectiveness and scalability of our search engine when working with very large collections of documents. In the TREC-7 experiments we set out to answer two questions:

- Are manually generated queries more effective than automatically generated queries?

For our first experiment, we produced two runs (fsclt7a and fsclt7m). For the first run (automatic), the system automatically constructed the queries from the topic title, producing the set of results (fsclt7a). In this run, the title words were connected together, using AND Boolean operators, into a query and submitted to the search engine. This approach reflects an assumption (admittedly false for some fields) that a title contains words that signal the content of the document. For the second run (manual), we employed a set of manually-produced query statements (fsclt7m). Our goal in this experiment is to deliver the highest percentage of relevant documents within the first 20 documents. This reflects our belief that in a 'live' search environment, the searcher would not scan more than @ 20 search hits before modifying or abandoning the search.

- What is the effectiveness and scalability of our search engine when dealing with very large collections?

For our second experiment, we indexed the VLC corpus into the three databases required by the track guidelines (Base-1, Base-10, and VLC100.) Because of its size and the capabilities of the equipment we used, the VLC2 corpus (100GB) was split up into four equally sized 'buckets' of data and indexed into four separate databases. These four databases were searched separately (in parallel) and the results were merged into a single result set.

The results merging algorithm has been developed and refined using the Ad Hoc corpora over the course of the TREC-4, 5 and 6 conferences. This algorithm has proved to work well in practice as was demonstrated in TREC-5 [2] and TREC-6 [3]. What differentiates our merging algorithm from many others is that it is a 'shared-nothing' algorithm where no statistics are shared between collections, neither are any global collection statistic gathered or generated prior to running any searches.

In addition to the required runs, which employed the automatic model described above, we ran our manual model against the same 3 databases. The purpose of this was to explore whether end users might need to adopt alternate search strategies when searching very large databases.

3 System Configuration

The MPS Information Server is a commercial full-text retrieval system that runs on a large number of Unix™ based platforms and on Windows NT™. Given a user query, the MPS system returns a list of relevance-ranked documents from a database. The system is capable of performing simple or complex term searching, phrase searching and proximity searching using parentheses, wildcards and Boolean operators. Soundex, typographical variation[1]. Fielded searches and numerical range searches are also supported. The system is designed to favor precision over recall when performing searches. Because it supports a number of different protocols, including WAIS-88, Z39.50-V2 (WAIS-V2 profile), STARTS and Gopher as well as two internal protocols, LWPS and Direct[2], the MPS Information Server is capable of responding to search requests from a wide variety of client applications.

The TREC-7 experiments employed version 4.5 of the MPS Information Server running on an UltraSparc 1/200 with 256 MB of memory and 160 GB of disk space. Four GB of disk space were set aside for the TREC data and index and 90 GB of disk space were set aside for the VLC2 data and index.

For the purposes of the experiments, we used a driver application that had been built for previous TREC experiments. Running on the UltraSparc, the driver application communicated with the MPS Information Server using the LWPS protocol. This driver application was designed to read TREC topic files, build a query by extracting a specific field, or fields, from the individual topic entries[3], run the queries against the MPS Information Server and save the query results in the TREC result format to a specified file. The result files could then be processed by the Trec Eval program to obtain the precision-recall values for that run.

The parser that was built to index TREC databases for the previous TREC experiments was reused with one modification, namely that we suppressed data present in a greater number of 'noise' fields. The rest of the documents were indexed as plain text, with the SGML tags extracted from the text, and the words stemmed using a plural stemmer. Word positions were extracted from the text to allow phrase and proximity searching if desired by the searcher. No other information was extracted from the text. All keywords in the news articles were suppressed as required by the guidelines.

---

[1] For example, missing letters, 'color' would also pick up 'colour', and juxtaposition of letters, 'animal' would also pick up 'ainmal'.

[2] LWPS is an inter-server communications protocol and Direct is a protocol which allows for rapid integration into front-end development application tools such as Perl or Tk/Tcl.

[3] In fact the queries formulated by the user are embedded into the TREC topic file and are marked up with SGML tags.

While the MPS Information Server's indexing application starts up with a default stop-word list (containing 547 words), it can be made to convert a word to a stop word if that word's total occurrence in the database reaches a specific threshold value. This threshold value would typically be set anywhere in the range of 20,000 to 500,000 occurrences and is site/collection dependent. For the TREC-7 experiments, we opted to turn off this threshold so that no additional words beyond the default list of 547 would be turned into stop words.

Four databases were created, namely the Trec-7 database, the Base-1, Base-10 and VLC100 database. Database sizes and build times were are follows:

Database:	Size:	Build time:
Trec-7	457MB	8 Hours
Base-1	237MB	1 Hour
Base-10	2.2GB	10 Hours
VLC100	20GB	96 Hours

It should be noted that the four databases making up the VLC100 database were searched by individual MPS Information Servers which were accessed by the MPS Information Server Gateway. This gateway presented the various physical databases as a single logical database to the driver application. The driver application was unaware of the fact that multiple physical databases were being searched and that multiple result sets were being merged, or where these databases were located. As explained in section 2 above, this is a 'shared-nothing' system so each MPS Information Server is searching a single collection.

During the year prior to running the TREC-7 experiments, we have been tuning the search algorithms using the TREC-6 data and relevance assessment in an effort to continually improve the search engine performance.

4 TREC-7 Results for FS Consulting Experiments

4.1 Searcher Model and Guidelines

One person constructed all query statements for the experiments. The initial parameters of the searcher 'model' were defined as follows:

- s/he regularly uses Internet search engines and/or library catalogs;
- s/he regularly uses search engines in the work setting;
- s/he may have some search training, but is not a professional searcher;
- s/he dislikes reviewing large search outputs;
- s/he is seeking information to solve a real-life problem;
- s/he may not be a content expert in the topic area of a given question.

The following instructions guided query formulation:

- prepare a simple search statement that will capture the most relevant documents for a given topic;
- use single or multiple terms, employing wild-card capability to capture multiple versions of a word, and/or quotes around several words (e.g., "cardiac arrest") to create a fixed phrase;
- apply Boolean logic as desired, using AND, OR or NOT operators. Create nested statements using parentheses if desired;
- no other databases or thesauri are available for consultation;

- the total time taken to prepare a single query should not exceed 5 minutes

4.2 First Experiment: Ad Hoc Automatic Run

4.2.1 Query formulations

The query formulation strategy for the Ad Hoc Automatic (fsclt7a) run created a set of title terms connected by Boolean AND operators. In the TREC-6 [3] experiments, we employed a strategy of using OR to connect the title terms; this new approach produced better results.

The following are some examples of the query formulations:

Topic 352: british AND chunnel AND impact

Topic 369: anorexia AND nervosa AND bulimia

Topic 386: teaching AND disabled AND children

4.2.2 Server Performance

The results for fsclt7a produced the following precision/recall figures over all of the topics:

```
Queryid (Num):       all  fsclt7a
Total number of documents over all queries
    Retrieved:     9471
    Relevant:      4674
    Rel_ret:        950
Interpolated Recall - Precision Averages:
    at 0.00       0.6078
    at 0.10       0.3026
    at 0.20       0.2080
    at 0.30       0.1367
    at 0.40       0.1001
    at 0.50       0.0639
    at 0.60       0.0339
    at 0.70       0.0236
    at 0.80       0.0228
    at 0.90       0.0208
    at 1.00       0.0132
Average precision (non-interpolated) over all rel docs
                  0.1146
Precision:
  At    5 docs:   0.3720
  At   10 docs:   0.3380
  At   15 docs:   0.3027
  At   20 docs:   0.2820
  At   30 docs:   0.2433
  At  100 docs:   0.1184
  At  200 docs:   0.0715
  At  500 docs:   0.0354
  At 1000 docs:   0.0190
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:        0.1678
```

Overall, for all topics, 20% of the relevant documents were retrieved from the database and only 10% of the documents retrieved were relevant in fsclt7a.

4.3 Second Experiment: Ad Hoc Manual Run

This searcher created manual questions for our TREC 4, 5, and 6 experiments. We reviewed results of earlier TREC manual runs and made slight adjustments to the approach. Earlier experiments showed that using more than 3 or 4 topics in a search statement often resulted in few hits. For TREC-7, the searcher was conscientious about keeping the topic phrases below 4. For TREC-7, the searcher was allowed to review a preliminary list of the 'top 20' hits before settling on a final query statement.

4.3.1 Query formulations

The following examples are typical formulations:

Topic 352: (chunnel OR "channel tunnel")

Topic 369: anorexi* OR bulimi*

Topic 386: (child OR Children OR youth) (disabl* OR disabilit* OR handicap*) (teach* OR educat*)

Most query formulations employed parentheses, wildcards and the AND and OR Boolean operators. As the examples indicate, not all capabilities of the system were employed (e.g., field searching, proximity searching beyond the user of phrases, soundex, typographical variation and "NOT" operators were not used for example). The bounded phrase was the most common feature used.

Past experience influenced the searcher's query formulation behavior in the following ways:

- she preferred to use the wild-card capability selectively to increase recall, rather than entering multiple forms of a word;
- in some situations, she continued to use multiple synonym sets, believing that it would increase recall in a selective fashion and/or counter the impact of the stop-word list;
- in topics whose subject included a unique or unusual word, she used a single term rather than including all concepts in the topics.

4.3.2 Server Performance

The results for the Ad Hoc Manual run, fsclt7m, produced the following precision/recall figures over all of the topics:

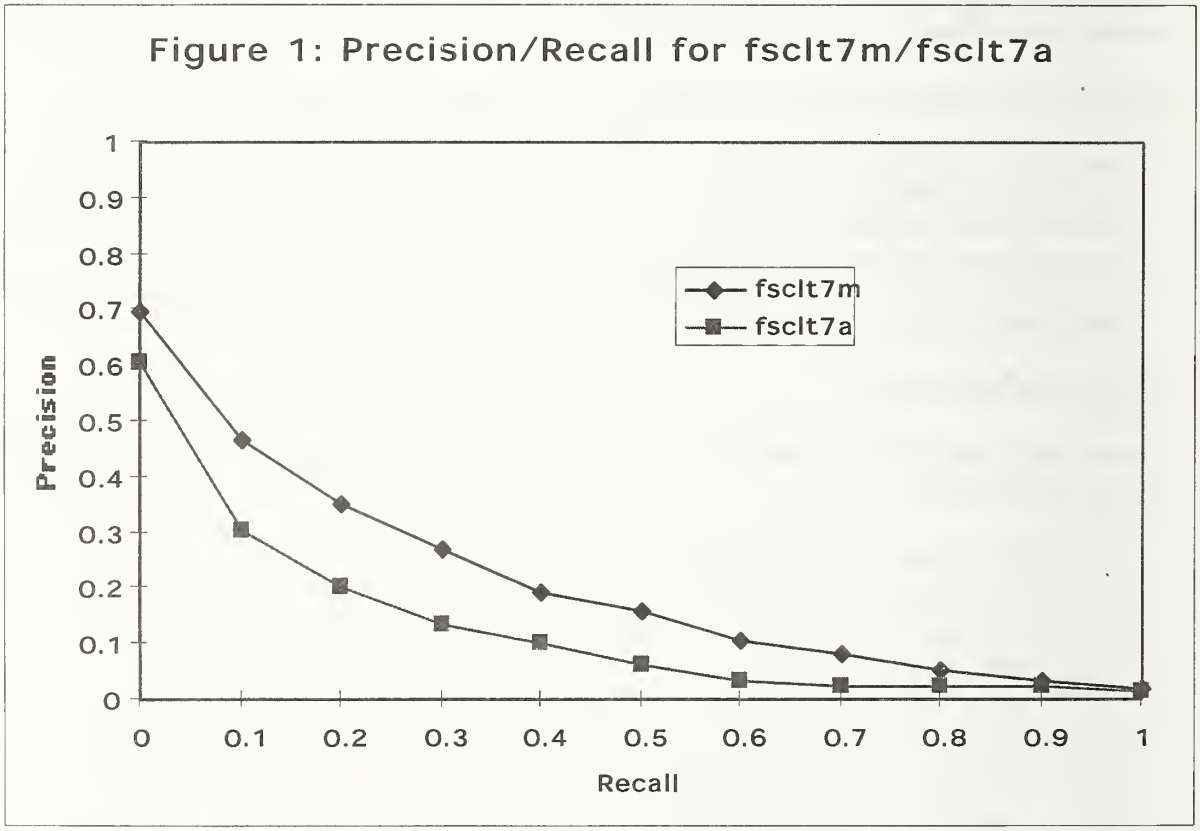
```
Queryid (Num):       all  fsclt7m
Total number of documents over all queries
    Retrieved:    18968
    Relevant:      4674
    Rel_ret:       1849
Interpolated Recall - Precision Averages:
    at 0.00       0.6951
    at 0.10       0.4667
    at 0.20       0.3526
    at 0.30       0.2709
    at 0.40       0.1944
    at 0.50       0.1582
    at 0.60       0.1061
    at 0.70       0.0806
    at 0.80       0.0527
    at 0.90       0.0314
    at 1.00       0.0179
```

```
Average precision (non-interpolated) over all rel docs
                0.1961
Precision:
  At    5 docs:   0.4600
  At   10 docs:   0.4160
  At   15 docs:   0.3947
  At   20 docs:   0.3810
  At   30 docs:   0.3333
  At  100 docs:   0.1962
  At  200 docs:   0.1270
  At  500 docs:   0.0648
  At 1000 docs:   0.0370
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:        0.2562
```

Overall, for all topics, 39% of the relevant documents were retrieved from the database and only 10% of the documents retrieved were relevant in fsclt7m. If we compare these results with our TREC 6 [3] results, we see a marked improvement of recall (up from 28%) and a very small drop in precision (down from 11%). The precision at 20 documents also went up from .3120 to 0.3810.

4.4 Discussion of Ad Hoc Runs

Figure 1 below shows the precision-recall curve for fsclt7m and fsclt7a.



These results indicate that using the title words alone does not work as well as using a manually-constructed query, even if the words in the title are ANDed together. However if we compare these results with our TREC-6 results [3], we find that ANDing the title words together works better than ORing them.

5 Second Experiment: Very Large Database Track (VLC2)

In the second experiment, we ran all the official runs required by the VLC track, namely automatic runs on the Base-1, Base-10 and VLC100 databases. In addition to these we also ran a set of unofficial runs using the manual queries created in the first experiment. As noted earlier, our purpose here was to explore the need for end users to adopt different search strategies when searching large databases.

The official runs were labeled as follows:

	Base-1	Base-10	VLC100
Automatic run	fsclt7a-v1	fsclt7a-v10	fsclt7a-v100
Manual run	fsclt7m-v1	fsclt7m-v10	fsclt7m-v100

Note that in the official runs, only the top 20 document were retrieved, limiting the total number of documents retrieved to 1000 over all 50 runs. Unfortunately a small, but problematic, operator error in setting up the driver application resulted in truncating the maximum number of documents retrieved to 19 as opposed to 20 which explains why only 950 documents were retrieved for each run as opposed to 1000. The analysis below centers only on fsclt7a-v100 and fsclt7m-v100.

A second set of unofficial runs was also submitted; in these runs, the top 100 documents were retrieved (making them a little more comparable to the Ad Hoc runs). These runs were labeled as follows:

	Base-1	Base-10	VLC100
Automatic run	fsclt7a-v1a	fsclt7a-v10a	fsclt7a-v100a
Manual run	fsclt7m-v1a	fsclt7m-v10a	fsclt7m-v100a

These unofficial runs were submitted to the VLC track relevance assessors in the hope that they might be judged, but it was unknown if they had been assessed at the time this paper was sent into NIST. No extended analysis of these runs is reported here, but we do summarize the results in section 5.3.

5.1 Results for VLC2

As with the Ad Hoc runs, we submitted results from an automatic and a manual run, using the same query formulation models.

5.1.1 First Experiment: Ad Hoc Manual Run

The results for fsclt7m-v100, the manually-constructed queries, produced the following precision/recall figures for all the topics:

```
Queryid (Num):       fsclt7m-v100
Total number of documents over all queries
    Retrieved:      950
    Relevant:      4440
    Rel_ret:        255
Interpolated Recall - Precision Averages:
    at 0.00       0.5142
    at 0.10       0.0967
    at 0.20       0.0188
    at 0.30       0.0188
    at 0.40       0.0188
```

```
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel docs(averaged over queries)
                  0.0387
Precision:
  At    5 docs:   0.3080
  At   10 docs:   0.2880
  At   15 docs:   0.2720
  At   20 docs:   0.2550
  At   30 docs:   0.1700
  At  100 docs:   0.0510
  At  200 docs:   0.0255
  At  500 docs:   0.0102
  At 1000 docs:   0.0051
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:        0.0629
```

Overall, for all topics in the fsclt7m-v100 run, only 5% of the relevant documents were retrieved from the database and 23.7% of the documents retrieved were relevant.

5.1.2 First Experiment: Ad Hoc Automatic Run

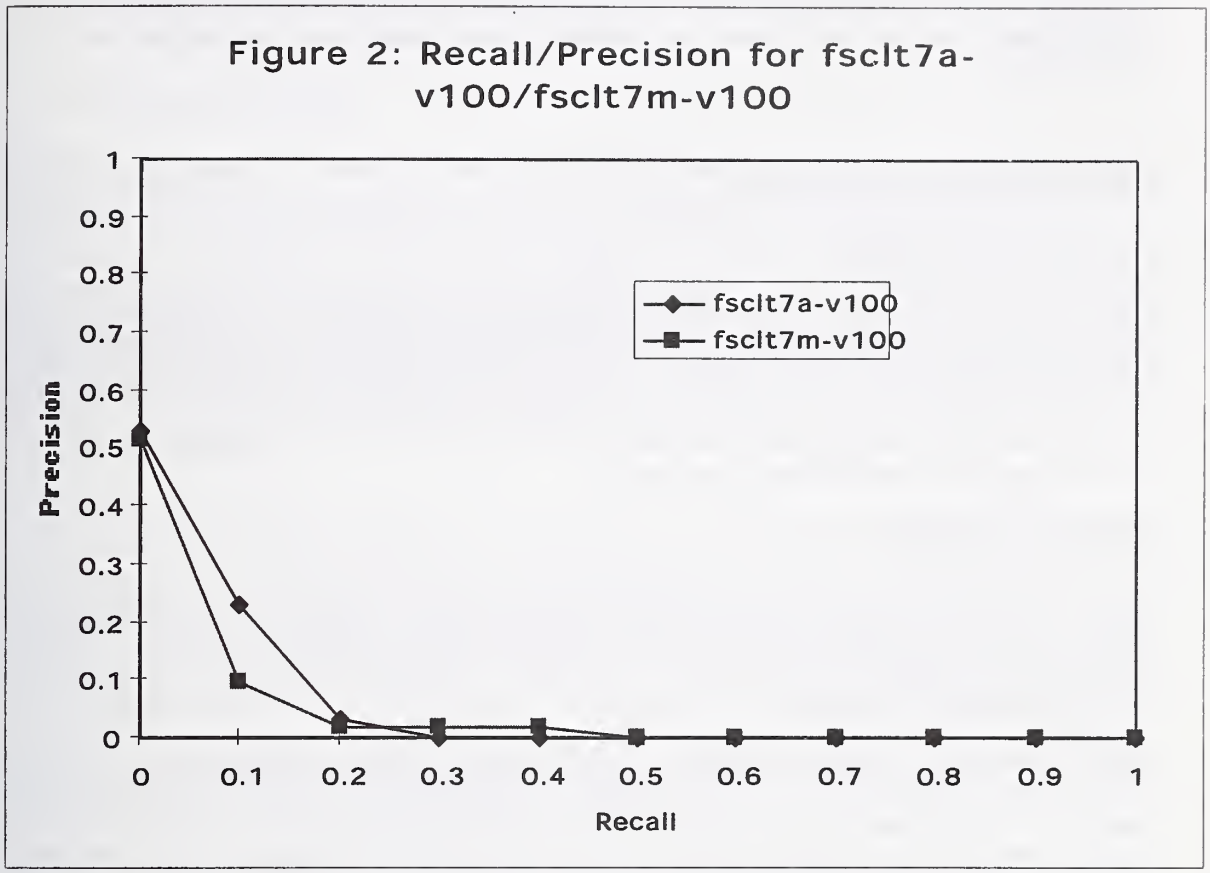
The results for fsclt7a-v100, the run employing automatically-constructed queries, produced the following precision/recall figures for all the topics:

```
Queryid (Num):       fsclt7a-v100
Total number of documents over all queries
    Retrieved:      941
    Relevant:      4440
    Rel_ret:        345
Interpolated Recall - Precision Averages:
    at 0.00       0.5284
    at 0.10       0.2288
    at 0.20       0.0303
    at 0.30       0.0000
    at 0.40       0.0000
    at 0.50       0.0000
    at 0.60       0.0000
    at 0.70       0.0000
    at 0.80       0.0000
    at 0.90       0.0000
    at 1.00       0.0000
Average precision (non-interpolated) for all rel docs(averaged over queries)
                  0.0472
Precision:
  At    5 docs:   0.3760
  At   10 docs:   0.3760
  At   15 docs:   0.3800
  At   20 docs:   0.3450
  At   30 docs:   0.2300
  At  100 docs:   0.0690
  At  200 docs:   0.0345
  At  500 docs:   0.0138
  At 1000 docs:   0.0069
R-Precision (precision after R (= num_rel for a query) docs retrieved):
    Exact:        0.0763
```

Overall, for all topics, only 7% of the relevant documents were retrieved from the database and 37% of the documents retrieved were relevant.

5.2 Discussion of VLC2 Track Results

Figure 2 below show the precision-recall curve for fsclt7a-v100 and fsclt7m-v100.



What is interesting to note is that these results are the opposite those obtained in Ad Hoc experiments. In those runs, the manually-constructed queries obtained better results; in this experiment, the opposite is true. In particular, the number of relevant documents retrieved is lower at every cluster of documents (5, 10, etc.) One possible explanation is because the queries were geared towards recall rather than precision and those kinds of queries might do well with smaller databases (such as 2GB), but break down when dealing with very large amounts of data (such as 100GB). From these results it seems clear that a different search strategy is needed when searching very large databases. We expected to get better results than these and we will be investigating why they are so disappointing over the next few months.

5.3 Additional results

In addition to the searches required by the VLC track, we also ran a number of unofficial searches as summarized below:

Run	Number of topics	Retrieved	Relevant	Relevant retrieved	P@20
fsclt7a-v1	47	692	4315	128	0.1362

fsclt7a-v10	50	885	4440	268	0.2680
fsclt7a-v100	50	941	4440	345	0.3450
fsclt7a-v1a	47	1859	4315	186	0.1340
fsclt7a-v10a	50	4275	4440	566	0.2590
fsclt7a-v100a	50	4746	4440	938	0.2850

These results show that raising the maximum number of documents retrieved with each query to 100 document increased the recall as one might well expect, but the numbers are still disappointing.

6 Life with great exponents

6.1 Data, data everywhere

TREC provides us with 2GB databases with which to experiment. While this could prove to be challenging 3 or 4 years ago, the current availability of cheap large capacity disks have negated this challenge. The 100GB database provided by the VLC track bring back most, if not all, these challenges. Even compressed, the data still require 35GB of disk space to store. Five 9.1 GB disks were needed for the data, five 9.1 GB disks for the indices and three 9.1 GB disks for temporary space required for index creation.

6.1 Time immemorial

Another challenge was the time required performing searches. The VLC100 database was bigger than any database we had searched before and we needed to do some software optimizations to attain the time required to meet the challenge set out by the VLC 2 track and hope to get the 'gold' medal.

The first task was to profile the software. Bottlenecks in software have a tendency to 'move' from function to function (or method to method depending on the language one uses) in any application when that application is subjected to a wide variety of tests. This was the case with our search engine when the amount of data that had to be searched was increased fifty-fold. Functions that used to be fast when dealing with small amounts of data can turn into major bottlenecks when dealing with very large amounts of data.

Prior to tuning the code, the VLC100 automatic run required 43 minutes to complete, after tuning the time required had fallen to 25 minutes.

6.2 I/O, I/O, it's off to work we go

Disk I/O is quite possibly the slowest link in the computing chain. Looking at the I/O statistics that were generated by the Sun Microsystems performance measurement program 'iostat'[4] showed that the system was doing a lot of swapping while running the queries. This is to be expected when searching such a large database on a system with a modest amount of memory (256MB)[5].

[4] 'iostat' is a Sun Microsystems tool which is shipped as part of the Solaris operating system, it allows one to monitor and measure the transaction rate of various devices in the computer. It is especially useful to identify bottlenecks on a computer.

[5] We realize that 256MB may not seem to be a modest amount of memory for most applications, the searching of very large databases benefit from having a much as memory as possible. It is now quite common to find machines with very large amounts of memory in them.

A trick gleaned from Adrian Cockcroft's excellent book on performance tuning [4], is to split the swap space required by the operating system across two separate disk drives rather than putting it all on a single disk drive as is the convention on Unix systems. The optimal solution would be to put the swap space on two separate disk drives each managed by separate disk controllers. The optimal solution was impractical in our setup so we settled for splitting the swap space in two equal parts across two separate disk drives, both working off the same disk controller. Doing that reduced the time it took to complete the run from 25 minutes to 20 minutes, better but still not good enough for the 'gold' medal.

6.3 Thanks for the memories

We then monitored the machine using the SE performance monitoring tool built by Adrian Cockcroft [4] and it showed that the machine was suffering from memory starvation once it got to the fourth or fifth query in the run. In technical terms, the pager was stealing memory pages too rapidly. In layman's terms, memory holding data that had just been read was being released prematurely because there was not enough physical memory to accommodate it in the amounts and speeds with which it was being read from the disks as the run was being executed. Upgrading the memory in the machine from 256MB to 768MB further reduced the time it took to complete the run from 20 minutes to 10 minutes, still not within reach of the 'gold' medal.

6.4 Together we stand

Shortly before the TREC conference, we were able to reorganize disks by striping[6] a number of them together to create a single virtual device using a Sun product called DiskSuite[7]. We created two virtual devices, one for the data (52GB) and one for the indices (78GB). This allowed us to create a single VLC100 database rather than create four separate ones. The MPS Information Server has generally been faster when searching a single large database than searching multiple small databases on a single CPU machine, so we were hopeful that this would allow us to further reduce the time. We were also hopeful that any performance impact that would be incurred by using DiskSuite would be offset by performance gains achieved by searching a single large database. We were proved wrong in our assumption, the time to complete the run increased from 10 minutes to 13 minutes.

6.5 So, where do we stand?

It seems clear it will be difficult to get many more performance improvements from the machines we are using. During the optimization process described above, the code was profiled and tuned several times, memory has been added, a single virtual database as been created. Thus, while the time to complete a run was reduced from 43 minutes to 10 minutes which is in itself no mean feat, we still don't qualify for the 'gold' medal in terms of search time.

---

[6] Striping disks together allows files to be laid out in stripes across a number of disks rather than contiguously on a single disk. The theory behind this idea is one can read the file faster as one is reading data from multiple disks as opposed to a single one, so one benefits from a certain amount of parallelism.

[7] DiskSuite is a software RAID solution from Sun Microsystems. It supports some basic RAID features such as device concatenation, device stripping and device mirroring. Sun is very non-committal about the performance impact of this product, only suggesting that performance might be improved in some situations and worsen in other while not providing any firm guidelines on how best to set up the devices based on their use.

The next step would be to migrate to a multi-processor setup and split the database across a number of disks working off separate disk controllers. Obvious solutions here would be to use a cluster of machines (such as fast PCs with fast disks and a fast interconnection network) or a multi-CPU system (with fast disks and multiple disk controllers). It also seems clear that using a software RAID solution (such as DiskSuite) does not really provide any performance gains at all thereby confirming Sun's assertion that performance gains may vary based on the application. The alternative would be to use a hardware based RAID system, though we don't have the means to do that.

7 Discussion of FS Consulting TREC-7 Results

The MPS Information Server is designed to operate in an interactive setting, where quick response and high precision are generally preferable to high recall. These results are close to our TREC-6 results but the recall is somewhat improved because of the queries.

For the Ad Hoc track, using manually generated queries worked better than using the title words alone. The VLC track results where just the opposite, the title words generated better results than the manually generated queries. As we mentioned above, we are not quite sure why this was the case, but we will be investigating why this was the case over the next few months. Additionally we are not sure why the results were so disappointing, but we will be investigating that as well.

8 Future Work

We will be working on the following areas prior to our participation in TREC 8:

- See if we can improve the precision of the Ad Hoc results.
- Investigate why the VLC results are so poor.
- See if we can further optimize the search engine to reduce the time required to run the searches.

9 References

[1] Schiettecatte, François and Florance, Valerie, "Document Retrieval Using the MPS Information Server". In Harman D. (Ed) The Fourth Text Retrieval Conference (TREC-4). National Institute of Standards and Technology Special Publication 500-236, Gaithersburg, Md. 20899.

[2] Schiettecatte, François, "Document Retrieval Using the MPS Information Server". In Harman D. (Ed) The Fifth Text Retrieval Conference (TREC-5). National Institute of Standards and Technology Special Publication,
Gaithersburg, Md. 20899.

[3] Schiettecatte, François, "Document Retrieval Using the MPS Information Server". In Harman D. (Ed) The Sixth Text Retrieval Conference (TREC-6). National Institute of Standards and Technology Special Publication,
Gaithersburg, Md. 20899.

[4] Sun Performance and Tuning, Adrian Cockcroft, Richard Pettit, Prentice Hall, 1998.

Fujitsu Laboratories TREC7 Report

Isao Namba Nobuyuki Igata Hisayuki Horai Kiyoshi Nitta
Kunio Matsui

Multimedia Laboratory Fujitsu Laboratories Ltd.
{namba,igata,horai,kiyoshi,kunio}@flab.fujitsu.co.jp

1 Abstract

In our first participation in TREC, our focus was on improving the basic ranking systems and applying text clustering techniques for query expansion.

We tested a variety of techiniques including reference measures, passage retrieval, and data fusion for the basic ranking systems. Some techiniques were used in the official run, others were not used because of time limitations.

We applied the text clustering techiniques for query expansion with a text clustering engine. Clustering base query expansion uses the top N best text clusters from the top 1000 documents instead of just using the top N documents.

Clustering base query expansion produces better results than simple query expansion based on passage retrieval.

We submitted three runs, Flab7at , Flab7ad, and Flab7atE. Flab7at is combination of ranking and query expansion by clustering the top 1000 documents on the title field, Flab7ad is combination of ranking and query expansion by clustering on the description field, and Flab7atE is combination of ranking with Boolean (existence) operators and query expansion by passage retrieval.

2 System Description

2.1 Overall

We particpated in TREC with two groups. One group was concerned with search engines, and included index construction, searching process, and normal query expansion. The other group was concerned with the Environment for Document Analysis (EDA), which is related to query expansion with text clustering.

The two groups had different locations, and the two systems were developped in completely different environments. To combine the two systems, we constructed an experimental search procedure using perl script. We also wrote a TREC local procedure in perl script for such tasks as removing stop patterns from input query.

2.2 The Search System Teraß

Teraß[1],[2] is fulltext search system, designed to provide an adequate number of efficient functions for commercial service, and to provide parameter combination testing and easy extension for experiment of information retrieval.

To satisfy both the commercial and experimental requirements, Teraß has many functions and extensibility as described below.

1. Basic search operations

   Boolean, Boolean Ranking (Ranking with Boolean operators for commercial use), Ranking (Accumulator Method) Near operators for phrase, Not operator, existing operator for term in ranking and parameter control, such as df limiter for ranking, term evaluation order control for ranking, etc.

2. Index type

   Inverted file index for fulltext search, number array index for range search, number array index for multiple occurences of number in a single text (eg. IPC code in a patent text) , and B-Tree index for item search.

3. Coding system for inverted file index compression

8-bit block coding (commercial standard)[3] δ, γ coding (academic standard)[4],[5], Extended γ coding [2].

4. Inverted file index construction

   Combination of term number, term frequency, and term offset in document

   With skip list[6],[4], and without skip list selection.

5. Ranking measures and easy extension to other measures

   Teraß supports reference measures such as Okapi, Lt.Lnu, and Cosine Measure, and is easily extended to other measures.

6. Other extensions

   (a) Parallel Processing[7], [8]
   Teraß can be extended to access multi indexes (data parallel) for large-size text collections or for getting better throughput on the AP-3000 parallel processor.

   (b) Language support and coding system
   Because the language-dependent part is separate, Teraß can be easily adapted to any language if the user writes a token analyzer. The current version supports English (stemming, soundex), Japanese (morphological analysis[9], Character N-gram), and Chinese (Character N-gram). Teraß also supports a variety of coding systems such as Unicode, Ascii, ISO8859, Shift-JIS, and EUC.

2.3 Environment for Document Analysis (EDA)

Environment for Document Analysis (EDA) assists in the analysis of many kinds of large document sets. It consists of several component tools. The major tool is Keyword Associator (KA) which provides the basic functions for full-text search engines based on vector space model, although it had originally been developed to support idea creation by presenting words related to a keyword[10]. The other components of EDA are KAcluster (part of KA), which provides a clustering function, Document Selector, which selects documents according to document attributes, and miscellaneous scripts and makefiles that combine EDA tools.

1. Keyword Associator

   Keyword Associator (KA) manipulates dictionaries of pre-calculated relevances between terms and documents. KA has five independent major parameters that determine the system behaviors of the basic functions: 1) Relevance pattern (RP), 2) Result type (RT), 3) Weighting measure, 4) Similarity measure, and 5) Normalization. Each of these parameters is described below.

   The functions provided by KA can be divided into four patterns:

   (RP-1) retrieve document from document

   (RP-2) retrieve term from document

   (RP-3) retrieve document from term

   (RP-4) retrieve term from term

   Each function pattern produces a result with three types of relevance:

   (RT-S) scalar value

   (RT-V) vector value

   (RT-M) matrix value

   Thus, 12 combinations of the RP and RT functions are available for obtaining relevances. The behavior of KA can be controlled by either command line options or by a KA-batch script language. KA-batch is an effective tool for document analysys and for reusing analysis techniques.

   The main dictionary of term-document relevance is produced from a set of documents. One of several term weighting measures, such as Lt.Lnu[11], tf, or idf, is applied to produce the dictionary. The relevance values of the retrieved results are calculated from the dictionary, using one of four similarity measures, Inner product, Dice coefficient, Cosine coefficient, or Jaccard coefficient[12]. The relevance is normalized either before or after the calculation. One of several normalization schemes (not apply, term vector sum = 1, term vector square sum = 1, document vector sum = 1, and so on) is applied.

2. KAcluster

   KAcluster reads the data of either the (RP-1, RT-M) or (RP-4, RT-M) types from KA

and forms clusters of the resulting element. One of the hierarchical agglomerative clustering methods (HACM), such as Single link, Complete link, Group average, Median, Centroid, or Ward's method[13], is applied to perform the task. Usually, clusters generated by a clustering method are disjunctive partitions of the document set by determining an appropriate threshold. However, finding the threshold is known to be of the most difficult problems. Therefore, KAcluster generates all possible clusters produced from a hierarchical structure that the clustering method creates. The resulting clusters are not disjunctive partitions.

3. Document Selector

   Document Selector assigns an order relation to the given document set and selects documents from it. The document set may or may not be clustered. One or more combinations of ordering algorithms using the attributes of each of the documents is applied. For example, the attributes are the rank and score of the pilot search, and the attributes of the cluster the document belongs to, if the document set was clustered. The attributes of clusters are the average and median rank/score, the best and worst rank/score, and size.

3 Processing

The section introduces the overall TREC7 processing.

3.1 Preprocessing

1. Indexing

   The indexing vocabulary consists of character strings made up of letters, numbers, and symbols. For every string of alphabetic characters, stemming algorithm of Porter[14] was applied, and no stop words were used in indexing. Thus, all the tokens in the text are indexed including ",", ".", and "?" . This is because we are not certain as to what is reliable stop word list, how it affects the result. Also, we had not enough time to compare the different indexes.

   In the search, we used two adjacent terms within a specified distance as a phrase instead of using two adjacent words with a specified document frequency.

2. List of stop words for query processing

   We used stop word list of about 400 words of Fox[15], and added some missing words such as "near", "taken", "taking", etc..

   At run time, words with a high frequency (over 50000-80000) were also treated as stop words.

3. List of Stop patterns

   A TREC query has many peculiar expressions such as "find document which," "relevant document will describe," which are artificial, and which cannot be eliminated by a stop word list. Ideally, stop patterns should be collected from the input collection of actual users queries, or from a semantic analysis of the input queries, but we don't have them. So to remove them, we made an N-gram ($N = 1 - 4$) of the tokens from all of the TREC1-6 topics, and manually selected the stop patterns.

4. Synonym dictionary

   We made a simple synonym dictionary of words which appear frequently in TREC topics such as U.S., U.S.A, Japan, British, and U.N .

5. KA dictionary

   All documents were pre-processed by KA , then Lt.Lnu was used to generate a dictionary of relevance between the terms and the documents for query expansion by document clustering. This process took about ten hours on an AP-3000.

3.2 Query Processing

This section contains an outline of the query processing. We used different measures for the pilot search and for the final search after query expansion. The pilot search measure is designed for high precision, where as the secondary search measure is a reference measure in TREC (Okapi). For query expansion, we used two different methods, one method simply used the top N ranked documents, while the other method used the top N ranked clusters from the top 1000 documents.

1. Query generation (Teraß format query generation)

   (a) Stop pattern/word removal

   Remove the stop pattern from the topic and replace it with "*".

   (b) Stop word removal

   Remove the stop word from the topic.

   (c) Query generation

   Generate a query based on words, and phrases.

   A phrase is an adjacent pair of words that are not stop words, and is expressed as a two-word pair within a specified distance without order.

   The distance is set at 4 or 5.

   (d) Must operator attachment (optional)

   Attach the term existence operator by simple sentence pattern analysis.

   This operator means that the term must be in the search result.

   (e) Synonym dictionary expansion(optional)

   Expand synonyms such as U.S etc.

2. Pilot search

   Simple term frequency/text size with range limiting, and average size considering text size normalization is used , which seems to be ok in our experiment.

3. Query expansion

   Query expansion is based on a single term because we have found that automatically generated term pairs at a specified distance hurt performance.

   (a) Passage retrieval base

   This query expansion is rather experimental due to the time limitation.

   i. Extract passages from the top 15-20 documents with the following conditions

      A. Choose one passage of less than 2 KB

      B. The passage contains most of the terms in the query

   ii. Add 20-30 terms in the passage with weighting

      A. Order from most frequent to least frequent in the passages

      B. df of the terms in all document sets is more than 19 and less than 20000

      C. Use the Rocchio formula with weighting $\alpha = 6, \beta = 1$

   (b) Clustering base

   Our method of query expansion by document clustering consists of the three steps described below. All steps are implemented by EDA.

   First, we generated clusters of documents from the results of the pilot search. We compared several document-document relevance calculation methods and clustering methods implemented in KA and KAcluster, respectively. Each query took an average of three minutes.

   Second, we used the generated clusters to select good documents and bad documents. The rank of the pilot search was the basis of the documents evaluation for this selection. We called the rank of the pilot search RPS, hereafter. Document Selector provides several evaluation and manipulation techninuqes for a set of clusters and for clusters (sets of documents).

   Finally, we generated two sets of weighted words from good and bad documents based on document-word relevance. We compared several document-term relevance calculation methods implemented in KA. A set of weighted words from the bad documents was used to suppress inadequate words in a set of weighted words from the good documents. This suppression is very important for improving the precision of the final search. These sets of weighted words and the query for the pilot search (QPS) were combined into a query for the final search. Several parameters were combined. Each query took an average of two minutes.

   Our query expansion by document clustering offers many possible methods and many parameters. We

combined several EDA functions by perl scripts, makefiles, and KAbatch scripts, and succeeded in implementing a flexible environment. Using qrels of TREC6 in order to build a query expansion by document clustering for TREC7, we selected a set of methods and determined a set of parameter values.

4. Final search

   We used OKAPI[16] for tf normalization. because we were not able to obtain good results with reference measure Lt.Lnu [11]. Besides our expermence in Lt.Lnu normaliztation, we were not able to obtain goot results with word frequency base text size normalization. We did not have enough time to determine the cause, but we suspect our handling of stemming or stop words.

4 Details

4.1 Measure

Our basic method was utilization of $tf * idf$, but with consideration of the following points:

- Words with small idf have a tendency to have high term frequency in a document, so the simple idf formula $log_2 N/n$ favors high df (small idf) words.

  A mecanically generated phrase with a large idf sometimes produces a bad result because its document frequency(df) is lower than 10, even as low as 1 or 2. Thus we established a minimal df.

- Measures favoring word co-ocurrence in a query get better results with short queries. To reflect this factor, we tested two solutions. One solution was to narrow the rage of tf in $tf*idf$; the other was to give bonus points when two words co-occur in text.

  In contrast, after automatic query expansion we need not consider the co-occurence of words in a query, because there are many words in the query.

- The text size of a correct text set for queries in TREC is mostly between one and three times of average text size. Measures that consider average text size, such as [16], [11], get better results.

1. idf

   $idf = log_2(\frac{N-\alpha*n}{n})$
   $idf = log_2(\frac{N-\alpha*N/k}{N/k}) if (n < l))$
   N = number of texts in the test collection
   n = document frequency for term
   $\alpha = 8 - 9$
   $l = 50$
   $k = 9000 - 10000$

2. tf of pilot search

   $tf1 = \frac{log_2(1+term\_freq)}{\sqrt[4]{\alpha*average\_doc\_size\_in\_byte+(1-\alpha)*doc\_size\_in\_byte}}$
   $tf = func(tf1, term\_freq, doc\_size\_in\_byte)$

   $func$ controls the range of $tf1$, and the max term num for each text size, and outputs tf value greater than 0.1, and less than 0.5

3. tf of final search (tf of OKAPI)

   $tf = \frac{(k_1+1)*term\_freq}{(k_1((1-b)+\frac{b*doc\_length\_in\_byte}{average\_doc\_length\_in\_byte})}$
   $k_1 = 1.5, b = 0.75$

4.2 Co-ocurrence Boosting

As was pointed out by [17],the measures favoring term co-ocurrence get better results for short queries. With a pilot search of a rather narrow tf range, we adopted a scoring method that favors term co-ocurrence.

Co-ocurrence boosting is implemented by simply multipling boost ratio to the similarity of each term.

$$S_i = \sum_t B * W_{t,i}$$

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.

As expected, parameter B depends on the query. For example, in the TREC6 collection, one query gets the best result with $B = 3.0$, while another gets the best result with $B = 1.08$.

We did not have enough time to determine how to assign the best boost ratio for each query, so we set the parameter to 1.08 for short queries, and to 1.40 for very short queries through all the pilot search. We set the parameter to 1.00 for the final search.

4.3 MUST Operator Attachment

To raise the precision of the pilot search, we tested a new operator called the "MUST operator".

The MUST operator requires that the word specified by this operator must exist in the search results. In some cases, the precision of the pilot search increases when the MUST operator is used. This operator was tested to obtain higher precision in pilot search; it was not used in final search.

The use of the MUST operator consists of two steps. The first step is to select important terms. The second step is to retrieve documents that include the terms specified by the MUST operator.

We used the following simple rules to select important terms for specification by the MUST operator:

- Select a term that begins with capital character.
- Select two terms after the words "of", "in",and "the".
- DO NOT select adjectives (such as "*out", *and" * ble").

Action by the MUST Operator is removed for words that fall under any of the following conditions:

- The df of the word is less than 25.
- The df of the word is more than 80000.
- The number of text hits becomes less than 100.

This rule, and the execution strategy of queries using the MUST operator is still tentative.

4.4 Query Expansion by Document Clustering

Our method of query expansion by document clustering consists of three steps: clustering the results of a pilot search, selecting documents, and generating a query for the final search. EDA supplies many functions for implementing these steps. In this section, we explain the real steps for TREC7.

4.4.1 Clustering pilot search results

We used KA and KAcluster with RP-1, RT-M, LtLnu, Inner Product, and Ward's method, then generated clusters from the top 1,000 documents resulting from the pilot search.

4.4.2 Selecting documents

Two sets of clusters (cluster sets) were generated from the generated clusters: one the good cluster set (GCS) and the other the bad cluster set (BCS). Because a cluster is a set of documents, a cluster set is a set of sets of documents.

The following steps were used to generate GCS and BCS. They were implemented by Document Selector.

1. Select clusters of an adequate size. We discarded every cluster that had fewer than n_1 elements or more than n_2 elements.
2. Generate GCS.
   (a) Measure clusters. For each cluster, we selected the best RPS as the representative document (REP) of the cluster, and calculated the average RPS of its elements (AVE).
   (b) Generate cluster sets. We grouped the clusters into cluster sets, each of which consisted of clusters with identical REPs.
   (c) Select each cluster from a cluster set. For each cluster set, we selected the cluster that had the best AVE in the cluster set. These clusters were combined to form a new cluster set whose elements have different REPs.
   (d) Select good clusters. From the cluster set, we selected n_3 clusters whose REPSs had better RPSs, and discarded all the other.
   (e) Select good documents. For each of the clusters in the good cluster set, we selected n_4 documents that had better RPS, and discarded all the others from the cluster. The result is GCS.
3. Generate BCS.
   We generated BCS in the same way as in Step 2, but 'good,' 'better,' 'best,' n_3, and n_4 in Step 2 were replaced with 'bad,' 'worse,' 'worst,' n_5, and n_6, respectively.

4.4.3 Generating a query

The query for the final search is generated from GCS, BCS, and QPS. A query is a set of words called a "word set," each of which has been assigned a weight.

The query for the final search consisted of the following steps:

1. Generate a good word set (GWS) from GCS.

   (a) Calculate weights. For each cluster in GCS, we used KA using RP-3, RT-M and LtLnu to generate a word set. The results were word sets, each of which corresponded to a cluster in GCS.

   (b) Select words. For each of the generated word sets, we selected n_7 words that had the greatest weight, and discarded all other words from the word set.

   (c) Average weights. For each word selected, we summed its weights in all word sets and divided the sum by the number of occurrences. The result is GWS.

2. Generate a bad word set (BWS) from BCS.

   We generated BWS from BCS as using a similar method to that in Step 1, except that n_7 is replaced by n_8 and the method of averaging weights in 1c is changed to summing the weight of each word in all word sets and dividing the sum by the square of its occurrence.

3. Combine GWS and BWS.

   For each word that is only in GWS, we multiplied its weight in GWS by n_9. For each word that is in both GWS and BWS, we multiplied its weight in GWS by n_9, multiplied its weight in BWS by n_{10}, and divided the former weight by the latter weight. The result is a word set that we call as the interim query (IQ). BWS contributes to suppress inadequate words in GWS.

4. Combine IQ and QPS.

   (a) Classify words. We divided the words in IQ and QPS into three groups: words only in IQ (OnlyIQ), words only in QPS (OnlyQPS), and words in both IQ and QPS (Both).

   (b) Recalculate weights for Both. For each word in Both, we multiplied its weight in IQ by n_{11} and added its weight in QPS to the multiplier.

   (c) Adjust weights. For each word in OnlyIQ, we multiplied its weight in IQ by n_{12}. For each word in OnlyQPS, we multiplied its weight in QPS by n_{13}. For each word in Both, we multiplied its weight, calculated in 4b, by n_{14}. The result was a new word set that consisted of all words in IQ and QPS with the multiplied weights.

5. Select words.

   we selected n_{15} words that had the greatest weight, and discarded all other words from the word set. The result is the query for the final search.

6. Comments

   Fifteen parameters $(n1, n2, \ldots, n15)$ appear in our query expansion by document clustering for TREC7. We decided on a set of values for the parameters by qrels of TREC6.

   At the step of adjusting weights in Step 4c, the multiplier of Both (n_{14}) should be set to a value that is greater than the others (n_{12} and n_{n13}). This adjustment of weights favor good words in QPS over bad words, and improves precision in the higher ranks in the final search.

   It is important to select clusters from cluster sets with appropriate variety in the step described in Section 4.4.2 (Selecting Documents) if the step described in Section 4.4.3 is to be processed successfully. If the clusters were selected only by AVE, the variety of the clusters is restricted. Step 2c is neccesary to solve the problem.

   Adequate values of all parameters depend on the deference of final search engines and documents sets. Our system implemented by EDA and some scripts is a flexible and powerful tool for determing a set of values for many search engines and document sets.

5 Ad hoc Results

Table 1: Eleven Point Average (Official Run)

Name	Flab7ad	Flab7at	Flab7atE
Category	Short	VShort	VShort
Mode	+CLS_QE +PH	+CLS_QE +PH	+MUST +QE +PH
at 0.00	0.7166	0.6771	0.6338
at 0.10	0.4962	0.4698	0.4530
at 0.20	0.3917	0.3635	0.3358
at 0.30	0.3298	0.2857	0.2644
at 0.40	0.2668	0.2206	0.2199
at 0.50	0.2055	0.1850	0.1796
at 0.60	0.1480	0.1370	0.1387
at 0.70	0.1006	0.0963	0.1052
at 0.80	0.0555	0.0579	0.0749
at 0.90	0.0287	0.0461	0.0402
at 1.00	0.0106	0.0154	0.0161
Average	0.2296	0.2126	0.2020

+CLS_QE = clustring-based query expansion; +MUST = Must operator;

+QE = passage retrieval base query expansion; +PH = phrase in query

Table 2: Eleven Point Average for Description (Base Point)

Category	Short	Short	Short	Short
Mode	-PH	+PH	-PH +QE	+PH +QE
at 0.00	0.7620	0.7734	0.6656	0.7103
at 0.10	0.4642	0.4516	0.4468	0.4811
at 0.20	0.3133	0.3340	0.3632	0.3801
at 0.30	0.2473	0.2753	0.2788	0.2974
at 0.40	0.1955	0.2138	0.2188	0.2372
at 0.50	0.1478	0.1504	0.1726	0.1935
at 0.60	0.1017	0.1091	0.1303	0.1475
at 0.70	0.0709	0.0810	0.0887	0.1151
at 0.80	0.0304	0.0505	0.0511	0.0652
at 0.90	0.0065	0.0169	0.0256	0.0330
at 1.00	0.0000	0.0046	0.0026	0.0132
Average	0.1863	0.1952	0.2042	0.2229

Table 3: Eleven Point Average for Title (Base Point)

Category	VShort	VShort	VShort	VShort
Mode	-PH	+PH	-PH +QE	+PH +QE
at 0.00	0.6806	0.6928	0.5806	0.6751
at 0.10	0.4162	0.4075	0.4442	0.4485
at 0.20	0.3056	0.2918	0.3683	0.3253
at 0.30	0.2315	0.2278	0.2902	0.2524
at 0.40	0.1740	0.1635	0.2375	0.2159
at 0.50	0.1322	0.1237	0.1696	0.1718
at 0.60	0.0892	0.0947	0.1259	0.1351
at 0.70	0.0587	0.0690	0.0921	0.1017
at 0.80	0.0380	0.0530	0.0532	0.0674
at 0.90	0.0131	0.0260	0.0296	0.0369
at 1.00	0.0011	0.0075	0.0040	0.0131
Average	0.1707	0.1706	0.2014	0.1993

Query expansion technique also produced a better result in our experiment. Clustering-based query expansion outperformed passage-based query expansion by 0.007 point.

Queries with phrases gets better results than queries without phrases, and queries with phrases also gets better results for query expansion. The phrase as a two-word pair within a specified distance without order sometimes results in documents including a bad phrase but with a high idf, and gets worse results than queries without phrase. We think that we must modify the phrase treatement in such cases.

Acknowledgement

The authors would like to acknowledge useful comments and suggestions of Mr. Isamu Watanabe who is the author of KA and Mr. Masayuki Sonobe of FUJITSU LABORATORIES LTD.

References

[1] Kunio Matsui, Isao Namba, and Nobuyuki Igata. A fulltext search system for large text (in japanese). *D-4-6*, 1997.

[2] Kunio Matsui, Isao Namba, and Nobuyuki Igata. High-speed text search engine (in japanese). *IPSJ 97-DD-7-3 pp15-21*, 1997.

[3] Niwa and Hirotora et al. Large text search system at altavista (in japanese). *IPSJ Advanded DataBase Symposium pp19-25*, 1996.

[4] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. Managing gigabyte - compressing and indexing documents and images. *Van Nostard Reinhold New York*, 1994.

[5] P. Elias. Universal codeword sets and representaions of the integers. *IEEE Transaction of Information Theory IT21:194-203*, 1975.

[6] William Pugh. Skip list a probablistic alternative to balanced trees. *CACM June 1990 Vol33 Num6 668-676*, 1990.

[7] Junichi Hagiwara, Tsunehisa Doi, Yoshinori Yaginuma, Kazuho Maeda, and Tatsuya Shindo. Commercial applications on the ap3000 parallel computer. *IEEE Massively Parallel Programming Models '97*, 1997.

[8] Tsunehisa Doi and Ikuo Miyoshi et al. Parallel text search engine on the ap3000. *Parallel Computing Workshop '97, P1-K*, 1997.

[9] Manabu Sassano and Isao Namba. A fast morphological analysis tool with user tuning (in japanese). *IPSJ 52th 5B-4 pp75-76*, 1996.

[10] Kozo Sugiyama, Kazuo Misue, Isamu Watanabe, Kiyoshi Nitta, and Yuji Takada. Emergent media environment for idea creation support. *Knowledge-Based Systems, 10, pp.51-58*, 1997.

[11] C. Buckley, A. Singhal, M. Mitra, and (G. Salton). New retrieval approaches using smart : Trec4. *TREC4 Proceedings*, 1994.

[12] Gerard Salton. Automatic text processing – the transformation, analysys, and retrieval of information by computer. *Addison Wesley*, 1989.

[13] Edie Rasmussen. Chapter 16, clustering algorithms. *Information Retrieval Data Structure and Algorithms ed. William B. Frakes, Ricardo Baeza-Yates Prentice Hall*, 1992.

[14] Porter M.F. An algorithm for suffix stripping. *Program, 13(3):130-137*, 1980.

[15] Chiristopher Fox. Chapter 7, lexical analysis and stoplists. *Information Retrieval Data Structure and Algorithms ed. William B. Frakes, Ricardo Baeza-Yates Prentice Hall*, 1992.

[16] S. E. Robertson, S. Walker, M. M. Beaulieu, and M. Gatford. Okapi at trec-4. *TREC4 Proceedings*, 1995.

[17] Ross Wilkinson, Justin Zobel, and Ron Sacks-Davis. Similarity measures for short queries. *TREC4 Proceeding*, 1994.

Information Retrieval and Visualization using SENTINEL

Margaret M. Knepper[1], Robert Killam[1], Kevin L. Fox[1], and Ophir Frieder[2]

[1]Harris Corporation
Information Systems Division
PO Box 98000
Melbourne, FL 32902-9800

[2]Department of Computer Science
Illinois Institute of Technology
10 W. 31st Street
Chicago, IL 60616

1. INTRODUCTION

Harris Corporation focuses on information retrieval support for various Government agencies. Time constraints and interest-level limit our user to reviewing the top documents before determining if the results of a query are accurate and satisfactory. In such cases, retrieval times and precision accuracy are at a premium, with recall potentially being compromised. To meet user demands our system, called SENTINEL, was designed to yield efficient, high precision retrieval.

This is the second time Harris has participated in the Text Retrieval Conference (TREC). We learned a lot from our first TREC [Knepper-97]. This year, we enhanced several aspects of our retrieval system and improved our performance over last year's results.

2. SENTINEL OVERVIEW

SENTINEL is a fusion of multiple information retrieval technologies, integrating n-grams, a vector space model, and a neural network training rule. SENTINEL is a C++ implementation of an Object-Oriented design. The basic structure of SENTINEL includes the following components:

- A web browser-based user interface that provides users with a mechanism to build queries for a topic of interest, execute the queries, examine retrieved documents, and build additional or refine existing queries.
- Multiple retrieval technologies utilizing n-grams and a Vector Space Model (VSM) to query the document corpus.
- A fusion component that combines and ranks the results of each of the retrieval engines.
- A *3*-dimensional viewer that provides users with a mechanism to explore various aspects of retrieved documents, looking for additional relevant documents.

A user begins by defining a topic of interest, then proceeds to define one or more queries for that topic. User queries to SENTINEL can take the form of

- Keyword(s) or phrases
- Example document(s)

SENTINEL focuses on an interactive multi-pass approach. We do not assume that the information will be found immediately, and therefore the user needs to iteratively refine the query. SENTINEL allows the user to review the documents and select the documents most relevant to the topic. Relevant documents can be used as queries to further refine the topic. The user can then quickly query over the data with the additional queries.

3. SENTINEL'S ENHANCEMENTS

This year, an improved ranking algorithm and a 3-dimensional visualization capability were incorporated into SENTINEL.

3.1 Ranking Algorithm

Last year only the VSM was used for document scoring. This year the results from each of the retrieval engines were integrated into a final score. The retrieval engines maintain only the high level scores. The user adjusts the lowest acceptable score and retrieval engine weight to effect score results to favor/disfavor a particular engine. SENTINEL standardizes the scores from each retrieval engine to range from 0 to 1. A ranking algorithm fuses the results of the retrieval engines and ranks the documents based on a variety of factors: the number of times the document was selected, highest score, lowest score, average score, location in the query list and number of retrieval engines locating the

document. Irrelevant documents and queries are removed.

Each retrieval engine is assigned a specific percentage. The document scores for the retrieval engines are reduced by the specified percentage. Depending upon the query type, different retrieval engines can be emphasized.

- Potentially misspelled words may put more emphasis on the *n*-Gram retrieval
- Document example queries place more emphasis on the VSM retrieval engine

An algorithm was developed to rank the document scores from different retrieval engines. The algorithm rates the following items:

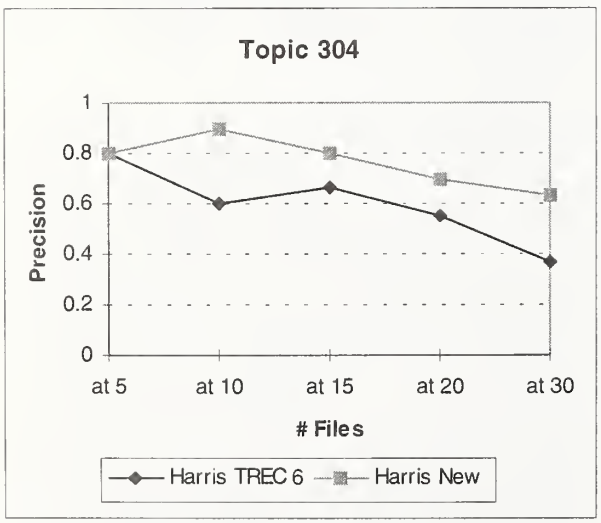
- Number of times document identified
    - Per query
    - Per retrieval engine
- Maximum score
- Minimum score
- Average score
- Penalty points

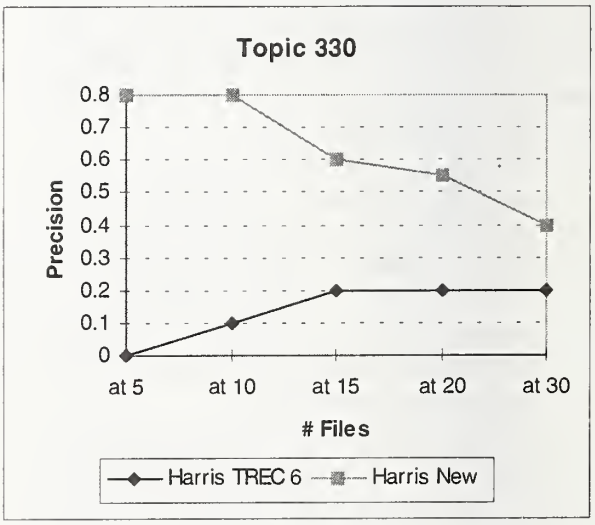
For all the items except the penalty points each item is ranked, the higher the number the lower the score. The individual items are totaled, and the lowest final score indicates the best document. Penalty points are assigned to a document based on the number of retrieval engines locating the document and the document location in the individual query list.

A penalty is added to a document for each retrieval engine not identifying it as relevant. Multiple engines retrieving a document is a strong indication of the document being relevant. This relevance correlation was also shown in [Lee-97]. Retrieval engines based on different search strategies and retrieving the same document are yet an even stronger indication of relevance [Alaoui-98]. The score must be above the minimum acceptable value after it is calculated for scaling and retrieval engine weight. During recent testing of the system, the team placed a lot of emphasis on the number of times the file was identified by multiple queries and multiple retrieval engines locating the same file. Setting high penalties on these values brought relevant documents to the top of the list. More experimentation with different types of data will help identify the values that should be assigned to the parameters.

Last year, the scoring algorithm took all query scores and put them into one final list. No consideration was given to the document's list location in the individual queries. The algorithm was modified so that each document receives a penalty point for its location in each individual query list. This is intended to reward the documents that are located close to the top of the list in the individual queries, by assigning fewer penalty points. Using this new technique we saw an improvement in the retrieval of relevant documents, Figure 1. We also added the ability to review individual results of the queries. This helped us quickly identify the usefulness of the queries and allowed us to focus on a query giving good results to help develop additional queries.

Figure 1 Sample results from the improved ranking algorithm applied to TREC-6 topics.





4. *3*-D VISUALIZATION

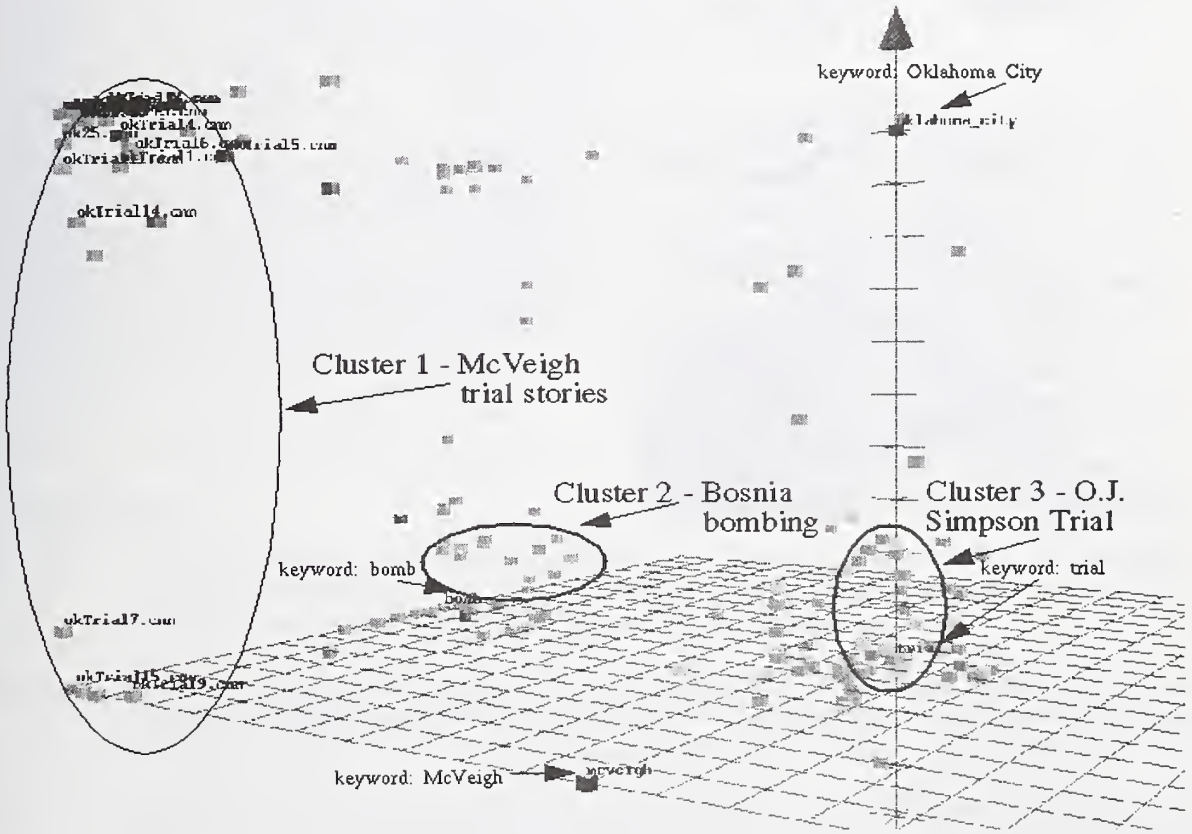
Primary user interaction with SENTINEL is through a web-browser-based user interface. Users can build and tailor queries as the topic of interest is further defined, moving from a generic search to specific topic areas through query inputs. Queries may consist of a single keyword, multiple keywords (or phrases), keyword clusters, an example document, and document clusters.

In SENTINEL, we enhance user understanding of the retrieved document set through the use of a Harris-developed *3*-dimensional visualization toolkit. This visualization tool supports multiple levels of data abstraction, clustered document presentation, data thresholding, and a variety of user interaction paradigms. The *3*-dimensional document visualization display enables the user to view different aspects of the document's topic, and provides an intuitive display of document relationships and similarity.

A set of documents retrieved for a particular topic may exhibit a variety of aspects. An example of different article aspects can be demonstrated by stories from the Oklahoma City bombing of 1996. There are articles about the bomb, damage from the bomb blast, rescue work, the victims, suspects, the Timothy McVeigh trial, the Terry Nichols trial, and the victims' memorial just to name a few. Each of these represents a different aspect of the Oklahoma City bombing.

Displaying the documents in a *3*-dimensional space enables a user to see document clusters, the relationships of documents to each other, and also aids in the location of additional documents that may be relevant to a query. Documents near identified relevant documents (identified through SENTINEL queries) can be easily reviewed for topic relevance. The user is able to manipulate the dimensional view to gain new views of document relationships. Changing the display axes allows the information to be viewed for different topic aspects to aid in further identification of relevant documents. SENTINEL is able to reduce the display down to the most important aspects of the document.

Figure 2 Example of document clustering



As illustrated in Figure 3, each document in the retrieved document corpus is represented mathematically in the *3*-D display space by a cube. The dark text indicates the relevant documents found through text queries submitted to SENTINEL, reviewed, and marked as relevant. The relevant stories appear to separate from the other stories, in the upper section of the view are surrounded by a dark lined box in Figure 3. These documents close to each other in the 3-D view, but are not close to each other on the list, Table 1. In the 3-D display, documents FT924-6336, LA050990-0002, and LA090389-0160 appear close to each other. They appear as on the list at the positions 2, 96, and 295 respectively. Using the 3-D tool gives the user a different view of document groupings that we don't ordinarily obtain from just a list.

5. TREC 7 RESULTS

As part of our evaluation of SENTINEL, we participated in both TREC-6 and TREC-7. We saw a dramatic improvement in the average recall and precision results, illustrated in the graphs of Figure 4. Overall, we also reduced the number of queries this year, as shown in Table 2. Since each query is a pass through the database, fewer queries reduce the retrieval time.

Figure 3 Example of a 3-D view for Query 372

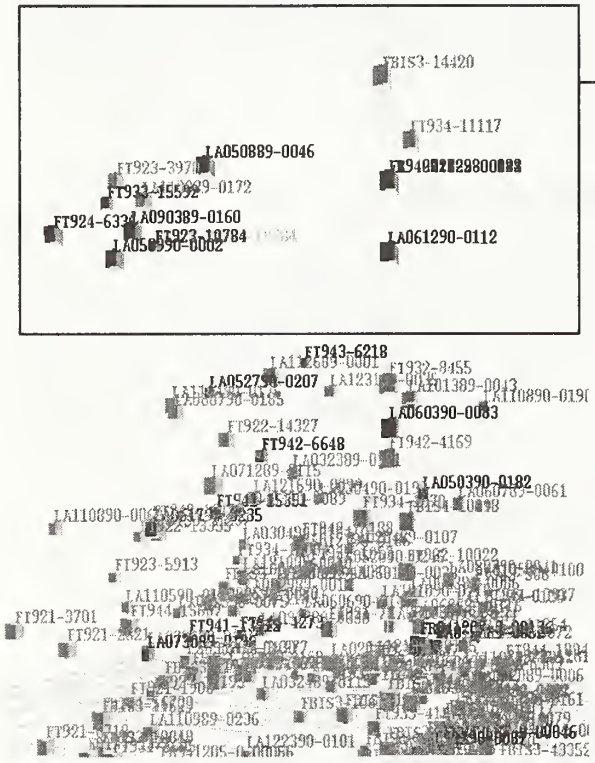


Table 1 - The 3-D view shows a different set of relationships, not observed by just looking at the location of the documents on the list

File Name	List Ranking	Judged as a relevant document
FT924-6336	2	yes
FT923-10784	3	yes
LA052790-0207	5	yes
LA050889-0046	6	yes
FT942-12298	10	yes
LA050990-0002	96	yes
FT934-11117	101	no
FBIS3-14420	150	no
FT943-6218	172	yes
LA110889-0172	198	no
FR940617-2-00244	265	yes
LA061290-0112	271	yes
FT923-3970	284	no
LA123189-0035	293	no
LA090389-0160	295	yes

Figure 4 Average Recall and Precision From TREC-6 and TREC-7

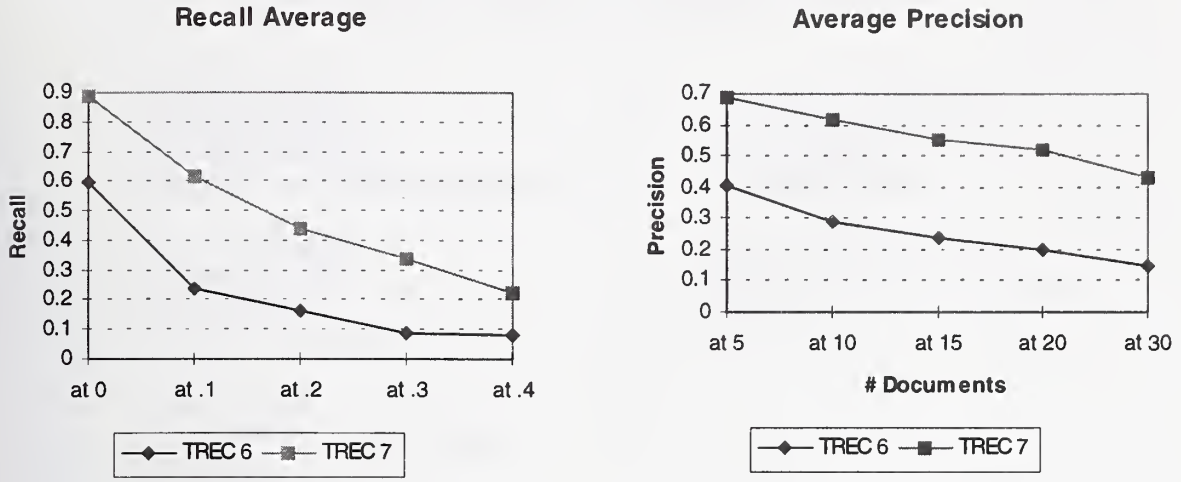


Table 2 Number of Queries

	Number of n-gram queries	Number of VSM queries	Total Number of Queries
TREC-6	404	544	948
TREC-7	449	443	812

6. CONCLUSION

The SENTINEL prototype is an efficient, high-level precision focused information retrieval and visualization system. It allows interactive formation of query refinement. It fuses results from multiple retrieval engines to leverage the strengths of the each. It has been designed for efficient maintenance, making it easy to add new documents. SENTINEL allows for multiple dictionaries and vocabularies – thus allowing a user to develop role-based dictionaries or vocabularies. Finally, SENTINEL provides a web-browser based interface for user interaction as well as a 3-D viewer for exploring the documents retrieved in response to a user's query. From a personal standpoint, we significantly gained a greater understanding of our query results using our visualization component. We see the importance of being able to respond real-time as the user rates the story and the need to filter the results shown to the user, i.e., show only the results from specific queries, or only show the first 30 documents.

7. REFERENCES

[Alaoui-98] S. Alaoui Mounir, N. Goharian, M. Mahoney, A. Salem, O. Frieder "Fusion of Information Retrieval Engines (FIRE)", 1998 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98), Las Vegas, July 1998

[Knepper-97] Knepper, Margaret M., Cusick, Gregory J., Fox, Frieder, Ophir, Killam, Robert A. "Ad Hoc Retrieval with Harris SENTINEL", The Sixth Text Retrieval Conference (TREC-6), NIST Special Publication 500-240, E. M. Voorhess and D. K. Harman, Editors, August 1998, Pp. 503-509

[Lee-97] Joon Ho Lee, "Analysis of Mulitple Evidence Combination", in the proceeding of the 20th annual ACM SIR Conference, (Philadelphia, PA), 1997.

Use of Query Concepts and Information Extraction to Improve Information Retrieval Effectiveness

David O. Holmes
NCR Corporation
Rockville, Maryland
david.holmes@washingtondc.ncr.com

M. Catherine McCabe
Advanced Analytic Tools
Washington, D.C.
cmccabe@gmu.edu

David A. Grossman
US Government
Washington, DC
dgrossm1@osf1.gmu.edu

Abdur Chowdhury
Illinois Institute of Technology
Chicago, IL
abdur@grouplogic.com

Ophir Frieder[1]
Illinois Institute of Technology
Chicago, IL
ophir@csam.iit.edu

Abstract:

In TREC-7, we participated in both the automatic and manual tracks for category A. For the automatic runs, we included a baseline run and an experimental run that filtered relevance feedback using proper nouns. The baseline run used the short query versions and term thresholding to focus on the most meaningful terms. The experimental run used the long queries (title, description and narrative) with relevance feedback that filtered for proper nouns. Information extraction tools were used to identify proper nouns. For manual runs, we used predefined concept lists with terms from the concept lists combined in different ways. The manual run focused on using phrases and proper nouns in the query. We continued to use the NCR/Teradata DBC-1012 Model 4 parallel database machine as the primary platform and added an implementation on Sybase IQ. We again used the relational database model implemented with unchanged SQL. In addition, we enhanced our system by implementing new stop word lists for terms and selecting phrases based on association scores. Our results, while not dramatic, indicate that further work in merging information extraction and information retrieval is warranted.

1. Introduction

Our work for TREC-7 is a continuation of the work started in TREC-3 when we implemented an information retrieval system as an application of a relational database management system (RDBMS). We used unchanged Structured Query Language (SQL) to implement vector-space relevance ranking (Grossman95, Grossman96).

[1] This work was supported in part by matching funds from the National Science Foundation under the National Young Investigator Program (contract IRI-9357785).

TREC-4 work demonstrated the relational implementation on category A data and introduced the concepts-list approach in the manual runs. In TREC-5, we implemented relevance feedback. TREC-5 also used the relational approach for the Spanish, Chinese and Confusion tracks. For TREC-6, we expanded our relevance feedback methodology to include the lnc-ltc term weights (Singhal96) as well as feedback term scaling. During TREC-6 we explored the assumption that certain infrequently occurring terms with high collection weights may actually be artificially inflating the query-to-document relevance ranking scores. We continued that work this year with expanded stop lists and term thresholding. In addition, this year we combined information extraction techniques with information retrieval through the use of a relevance feedback filter based on IE (Information Extraction).

Our manual runs have focused on the concept approach to structuring queries. In TREC-4, we assigned the query terms in up to three concept lists and used general world knowledge to expand the query to include other similar terms not found in the topic. In TREC-5, we continued to use the concept lists and experimented with the use of manually assigned weights to the query terms as well as using manual relevance feedback to identify additional terms. For TREC-6, we augmented our prior work with inexact term matching and an automatically generated thesaurus based on term-to-term co-occurrence. This year, our manual run took a somewhat more structured approach than in years past, with the hope of automating some techniques. In particular, our manual run focused on using phrases and proper nouns to improve precision and recall. Seventy percent of the manual search elements were phrases, which produced an average precision of 0.3333. At the 10-document retrieval level, our precision was 0.64.

2. Prior Work

2.1 Implementation of an Information Retrieval System Using the Relational Model

The implementation of an Information Retrieval (IR) system using the relational model hinges on the use of a relation (table) to model an inverted index which is central to traditional IR systems. The inverted index stores each unique term or phrase from the collection and a list of all the documents containing each entry. The inverted index can also include frequency, offset, or other desired information. In the relational approach, this index is flattened or normalized and stored in a table, as shown in Figure 2.1. Queries can be implemented using standard structure query language (SQL) to find and rank all documents containing the query terms. Full details of the implementation can be found in Grossman97 and Lundquist97. For example:

Collection:

D1: water freezes when cold
D2: the water is cold today

Inverted Index:

Cold → D1, D2
Freezes → D1
Today → D2
Water → D1, D2
When → D1

Relation:

Cold	D1
Cold	D2
Freezes	D1
Today	D2
Water	D1
Water	D2
When	D1

- Figure2.1 -

One benefit to using the relational model for IR is the ability to exploit parallel processing via the DBMS. All commercial DBMS systems offer a parallel version. We implemented an IR system using Teradata's RDBMS on a 4 processor DBC/1012 parallel processing machine. The Teradata DBC/1012 Database Computer is a special purpose machine designed to run a relational database management system using standard SQL.

2.2 Relevance Feedback in the Relational Model

Building on prior work in automated relevance feedback, our TREC-6 submission implemented automated relevance feedback within the relational model, using unchanged SQL. Prior work has shown that the best results are achieved when not all of the terms from the top documents are used (Lundquist97, Buckley95). For instance, the top terms (based on normalized *idf* value) from the relevant documents are added to the original query terms and the query is rerun. Buckley added the most frequently occurring 50 single terms and 10 phrases from the top 20 documents. Lundquist showed that using the best 10-20 terms and phrases from the top 5 –20 documents with a scaling factor of .5 for the terms being added to the query (multiplying the weight by .5) performed best.

2.3 Information Extraction

Our TREC-7 implementation of relevance feedback experimented with a fusion of information extraction and information retrieval. Information Extraction is the process of identifying instances of entities within text.

Examples of entities include: organization, location, person, date reference, time reference, etc. The TIPSTER sponsored Message Understanding Conference is dedicated to improving the technologies for entity extraction. Our work focused on a subset of entities – proper nouns. Numerous techniques exist for tagging parts of speech using natural language processing. We used the commercial product, INSO, as the basis for identifying proper nouns. (Inso97).

2.4 Term-Term Association Techniques used for Stop List generation

The stop list used in the IIT (Illinois Institute of Technology) system was expanded for TREC-7 based on the positive impact of a larger stop list used during experimentation for TREC-6. A manually generated stop word list for term processing which consists of approximately 3,500 words included based on high document frequency and variants (prefixes and suffixes, as we do not use stemming) of those terms with high document frequency. A second stop list, a subset of the term stop list, is used for phrase preprocessing. The idea is to index high quality phrases rather than merely locate adjacent word pairs. We eliminate from the phrase stop list those terms which have a high affinity with other terms in the collection that are found adjacent to the potential stop word. High frequency terms remain on the term stop list but because they qualify as existing in a high-association pair, they are removed from the phrase stop list and phrases containing that word are indexed. For instance, "big apple" might be retained while 'big' alone might be removed.

Three measurements determined the contents of the stop list. To measure the affinity of the first word in a phrase to the entire phrase, we computed:

$$a1 = \frac{b}{c}$$

where

a is the association measure
b is the term frequency of a phrase
c is the term frequency of the first word in a phrase
d is the term frequency of the second word in a phrase

To measure the affinity of the second part of the phrase to the phrase, we computed:

$$a2 = \frac{b}{d}$$

The overall association measure was computed as follows:

$$a3 = \frac{b}{((c+d)/2)}$$

Terms were removed from the stop list if *a3* greater than 0.0005 (we calibrated several different values and this was best) and either *a1* or *a2* was also greater than 0.0005 for one or more phrases in the collection. Figure 2.4-1 provides examples; italicized entries do not qualify as phrases. Given the word *higgledy*, there is a 0.94 probability it is followed by *piggledy* in the TREC6 collection, which indicates, by our metric, it is a content-bearing phrase. While *brothers owned* is not a phrase, *Ringling Brothers* is. While not empirically measured, a casual inspection of phrases with high association scores revealed a substantial percentage of proper nouns.

First Word	Second Word	Phrase TF	First TF	Second TF	a3
Higgledy	Piggledy	16	17	17	0.94000
Humpty	Dumpty	38	51	39	0.84000
Sherlock	Holmes	105	236	1,641	0.11000
World	Bank	7,813	176,487	172,982	0.05000
Orange	Juice	536	32,578	2,477	0.03000
Nuclear	Waste	1,745	83,479	34,811	0.03000
Big	Apple	106	73,665	4,984	0.00300
Ringling	Brothers	12	54	13,059	0.00200
David	Holmes	13	34,375	1,641	0.00070
Nuclear	Family	43	83,479	73,511	0.00060
Women	***Buy***	***13***	***59,844***	***43,870***	***0.00025***
Brothers	***Owned***	***3***	***13,059***	***43,939***	***0.00007***

Figure 2.4-1 Self-Association Examples

We compared several possible term and phrase stop lists using TREC-6 topics to determine our choice for TREC-7. The results are summarized in Figure 2.4-2.

Stop List	Average Precision	Relevant Retrieved
SMART	0.1694	2102
Fox	0.1744	2086
TREC-6 (GMU)	0.1816	2085
TREC-7 (new lists)	0.1905	2122

Figure 2.4-2 Comparison of Stop Lists on Average Precision

3. Implementation Details

3.1 Automatic Runs

3.1.1 Automatic Run 1 — Proper Nouns Filter Relevance Feedback

The IIT automated long run experimented with using proper nouns for relevance feedback only. The initial queries consisted of the title, description and narrative portions of the TREC-7 queries. Relevance feedback was used to augment the queries with the top proper nouns from the top documents.

In order to limit the number of terms added in relevance feedback, we developed a filter which limits the terms added in relevance feedback to the top 10 proper nouns, ranked by N*_idf_ where N is the number of times the term occurs in the top 20 documents. We used the INSO product to identify the noun phrases (including single-term nouns) from a subset of the TREC7 collection. A crude assumption was made that any capitalized noun phrase was a proper noun. This list was then used as a filter to select terms for relevance feedback. First, the unaltered long queries were run. The top 10 documents were retrieved, and the top ten index entries for those documents were identified (based on N*idf). That list of candidates was then compared against the extracted proper nouns. If found on the list, the term was added to the original query. In this way, ten or fewer terms were added to each query. Because the IIT system only indexes single terms and term pairs, only those noun phrases were actually found in the index. INSO identified 367,142 noun phrases from the training subset of theTREC-7 corpus. The benchmark against TREC-6 is shown in figure 3.1-1.

Test	Average Precision	Relevant Retrieved
No feedback	0.1905	2122
INSO phrase feedback	0.1925	2164

Figure 3.1-1 IIT Automatic Run 1 TREC6 Benchmark

3.1.2 Automatic Run 2 — Baseline

The second automatic run formed our baseline, using the title portion of the TREC-7 queries and no relevance feedback. This run was implemented using Sybase IQ and took advantage of the bit-wise indexing available in that database. We implemented query thresholding, which eliminated low *idf* terms from the query unless it was the last remaining term in the query (Grossman97). In addition, we thresholded the index used for this run and eliminated the most common terms and phrases. This resulted in a very fast system – because the most frequent terms were removed. In addition, the index size was reduced by half. The results were not good, however, indicating that we may have gone too far in the cut-off point used for thresholding.

3.2 Automated Concept Experiment

This experiment was conducted after submission of our official runs for TREC-7. Automatic tests against TREC-6 and manual runs against TREC-7 indicated improved precision and recall with the implementation of concepts which limit the size of the answer set for each topic. To do this automatically, we required each

retrieved document to contain at least one term or phrase from the Title concept. Terms and phrases from the Description and Narrative sections contributed to the relevancy score, but did not identify new documents.

The IIT system does not use stemming and the fallout from using terms and phrases from only the Title section for document qualification is dramatic. To minimize the problem, the Title concept was automatically expanded, based on several criteria. The scoring only concept was also automatically expanded to improve ranking. Figure 3.2-1 summarizes the results of the experiment with TREC-6 topics.

TREC-6 Test	Average Precision	Relevant Retrieved
Long	0.1905	2122
Title Concept, Long Scoring	0.2559	2273

Figure 3.2-1 Automated Concepts Experiments on TREC-6 data

3.3 Manual Run

3.3.1 Manual Run Implementation Details

We conducted failure analysis of our TREC-6 results and found that when the statistical approach to relevance feedback added phrases and proper nouns, precision and recall generally improved (Lundquist98). In particular, topic 311 performed well and contained names from high profile industrial espionage cases. To produce the manual queries, we followed a two-step process. First, the analyst read each topic and spent approximately one to ten minutes building an initial query. After completing the initial queries, the analyst briefly scanned the results and examined a few documents in detail for each topic. The analyst read high-ranking documents and others with intriguing headlines, adding promising terms and phrases to the queries. In some cases, the analyst issued a query to find frequently occurring phrases within a selection of five to ten potentially interesting documents. The entire process took approximately 15 to 30 minutes per topic.

Our manual submission concentrated on using phrases and proper nouns to improve precision and recall. Queries consisted of 908 phrases and 389 single terms. Of the total, 541 were proper nouns including 235 names of people.

Similar to last year, we implemented one or more mandatory concepts for each topic. Queries selected documents only if the documents contained one or more entries from each mandatory query concept. For a few topics, we attempted to eliminate non-relevant documents by adding 'negative concepts' — terms which must NOT be in the document for the document to be selected. However, our implementation of this feature

contained an error that resulted in these terms actually being *required*. For topic 351, documents should have been dropped if "fishing boats" occurred within the body of text, instead those documents were actually added to the result set. Finally, to enhance ranking without qualifying additional documents, we implemented a *scoring-only* concept. If a document contained a term from this concept, its ranking would be adjusted.

3.3.2 Manual Run Failure Analysis

Our official manual run was above the median on 30 topics and below on 17. Figure 3.3-1 shows how the manual results differ from the median.

The SQL used in the manual run had an error that applied the negative concept terms to all topics, instead of limiting them to their assigned topics. This error degraded the performance of 28 topics. Interestingly, the error actually improved 21 topics and on topic 363 the mistake pushed the total recall to 100% — the fix lost one of the relevant documents by ranking it lower than the top 1000. Overall, average precision degraded by 5% because of the system error regarding negative concepts.

Topic 389 demonstrates the potential value of using proper nouns and phrases. While the median was 0.0305, the IIT system averaged 0.4985. Proper nouns such as South Africa, Carlos Cardoen, Christopher Drogoul, Edward Bush, Jonathan Pollard contributed to the success of this query.

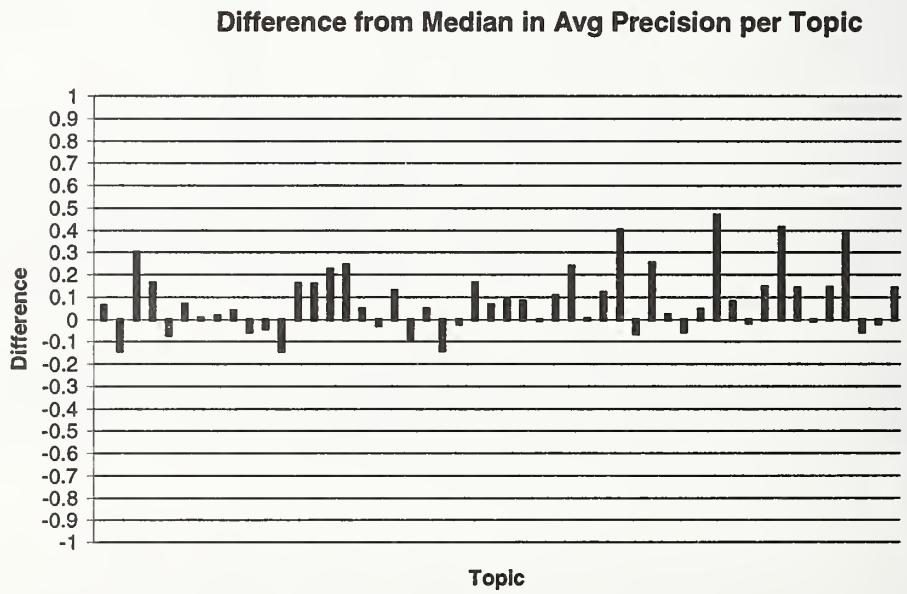


Figure 3.3-1: IIT Manual Run Difference from the median

4. Results

Figure 4-1 summarizes the results of our TREC-7 tests. An error in the SQL used for the manual run lowered our official results. The corrected SQL, using the exact same search terms, was run again and the results are included.

	iit98au1 (Long)	iit98au2 (Short)	Iit98ma1 (Manual)	iit98ma1 (Fixed)	Auto Concept
Avg. Precision	0.1929	0.1459	0.3333	0.3511	0.2091
Precision at 10 Documents	0.4500	0.3200	0.6400	0.6540	0.4740
Relevant Retrieved	2505	1676	2793	2914	2495
Above Median (Avg. Precision)	24	10	30	34	27
Below Median (Avg. Precision)	26	40	17	16	23

Figure 4-1: IIT Trec7 Results Summary

5. Conclusions and Future Work

For TREC-7, we focused on integrating Information Extraction and Information Retrieval through the use of a relevance feedback filter. In addition, we refined our use of concepts in the manual runs. Our results show promise in the use of phrases and proper nouns. Using term-term association scores to control the preprocessing of phrases improved our system over last year. This year, we took our initial steps in implementing information extraction tools to automatically expand queries to include proper nouns. The marginal improvement experienced this year is probably indicative of the preliminary implementation using only proper nouns from a sample of the corpus.

Our future challenges include automating the techniques used in our manual process. Our initial tests indicate that some degree of automatic concept detection is possible in the TREC environment. Post-TREC-7 experimentation with automatically selected terms and phrases for a mandatory concept and a scoring concept resulted in improving average precision by 8.9%. Further use of information extraction and term-term associations are candidates for further concept refinement. In addition, further integration of information extraction in relevance feedback is needed to move beyond proper nouns and experiment with the use of entities as feedback filters.

6 Acknowledgments

We would like to thank James Dyer and Bret Bailey of Sybase for their support in the IQ implementation.

References:

(Buckley95) Buckley, C. A. Singhal, M. Mitra, and G. Salton, "New Retrieval Approaches Using SMART: TREC-4," sponsored by the National Institute of Standards and Technology and the Advanced Research Projects Agency, November 1995.

(Fox90) Fox, Christopher. A Stop List for General Text. SIGIR Forum, (v. 24, no. 1-2) 1990, p. 19-35.

(Grossman95) Grossman, D., D. Holmes, O. Frieder, M. Nguyen, and C. Kingsbury, "Improving Accuracy and Run-Time Performance for TREC-4," Proceedings of the Fourth Text REtrieval Conference (TREC), sponsored by the National Institute of Standards and Technology and the Advanced Research Projects Agency, November 1995.

(Grossman96) Grossman, D., C. Lundquist, J. Reichert, D. Holmes, and O. Frieder, "Using Relevance Feedback within the Relational Model for TREC-5," Proceedings of the Fifth Text REtrieval Conference (TREC), sponsored by the National Institute of Standards and Technology and the Advanced Research Projects Agency, November 1996.

(Grossmam97) Grossman, D., D. Holmes, O. Frieder, and D. Roberts, " Integrating Structured Data and Text: A Relational Approach," *Journal of the American Society of Information Science*, January 1997.

(Inso97) Inso Intelligent Parts of Speech Tagger. Inso Corporation. 1997.

(Lundquist97) Lundquist, C., D. Grossman, O. Frieder, and D. Holmes, "A Parallel Implementation of Relevance Feedback using the Relational Model," *Proceedings of the World Multiconference on Systemics, Cybernetics, and Informatics*, July 1997.

(Lundquist98) Lundquist, C., D. Holmes D. Grossman O. Frieder. Expanding relevance feedback in the relational model. NIST Special Publication 500-240, pages 489-502, August 1998.

(Singhal96) Singhal, A., C. Buckley, and M. Mitra, "Pivoted Document Length Normalization," *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Ed. Hans-Peter Frei, Donna Harman, Peter Schauble and Ross Wilkinson, SIGIR Forum, August 18-22, 1996.

Feature Reduction for Information Retrieval

Stefan M Rüger
Department of Computing
Imperial College of Science, Technology and Medicine
180 Queen's Gate, London SW7 2BZ

1 Introduction

Our experiments for the ad hoc task were centred around the question how to create a document surrogate that still contains enough information to be used for a high-quality, efficient retrieval.

In the first step we drop all the function words and all the auxiliary words that although having a proper meaning merely help to communicate about the topic without being relevant to the topic. We apply part-of-speech analysis in order to retain the nouns and adjectives of a document. Standard term and document frequency analysis is used to compute a weight factor for each of the remaining words.

In a second step, we plan to set the relevant words into a relation that conveys a part of the meaning. Like in vector space models, both topic and document would be represented in this keyword-relation form and a suitable metric would quantify the relevance of a document to a topic.

At this stage of our research, no relations are stored in document surrogates. The automatic processed topic descriptions, however, include some very crude relation analysis that, eg, transfers "relevant documents describe cases of drink-driving outside France" to "drink driving outside France" and hence, knowing about the connotation of " ... outside ...," a negative weight factor for the occurrence of drink-driving and France. It is planned for future work to analyse relations more and more with statistical models and with trained probabilistic models and less with linguistic analysis.

For now, the purpose of our experiments is assessing the performance of the above very simple model of pure feature reduction without relations, without training/learning weights without sophisticated recall procedures, without inverted document files and without a proper document retrieval system. It might be interesting to see which effect feature reduction algorithms have in other, sophisticatedly tuned systems.

2 Preprocessing of the Documents

2.1 Data Flow

The basic assumption is that a document collection consists of one or more files. Each file contains one or more SGML/XML-tagged document bodies that are each preceded by the line

```
<DOCNO> document-name </DOCNO>
```

The preprocessing process is shown in Figure 1. In the moment, several steps are involved. Most notably, a part-of-speech tagger **pos-tagger** (Brill 1994) is used to find the role of each

word. We believe that nouns and adjectives are most vital to the contents of a document. Hence, we only consider words (including proper nouns) that are used in this way. We eliminate stop words, fold all characters to lower case and use Porter's stemming algorithm to obtain word stems (thus, eg, identifying singular words with their plural form). A simple analysis associates a document with a list of relevant word stems (these are the *terms* in our context), and their term and document frequency.

In a heuristic attempt to increase the weight of titles, headlines, etc, we count each word fourfold that is enclosed in a matching XML command pair at the beginning and end of the same line such as `<TITLE> Cuba Crisis Revisited </TITLE>`.

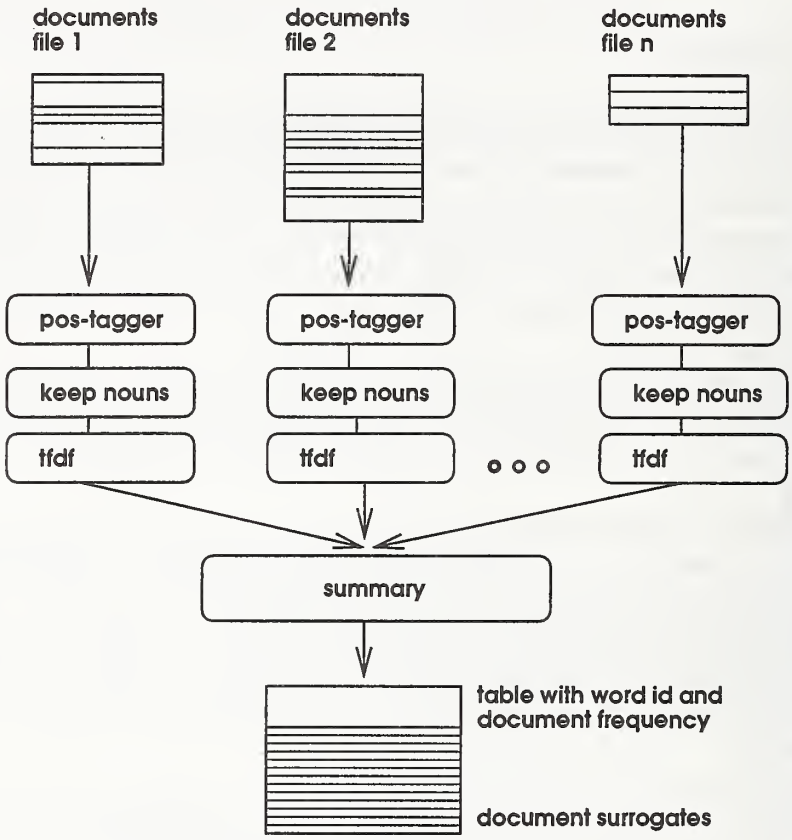


Figure 1: Preprocessing: data flow

The vocabulary (set of different words or terms) of a growing document collection does not seem to saturate even at a high number of documents. Figure 2 shows the number of different words versus the number of words in a growing document collection (up to 210,000 articles, 4 years of Financial Times).

As the vocabulary increases, so does the set of adjectives and nouns that are left after preprocessing. We verified that the validity of Zipf's law (Zipf 1949) also extends to the subset of nouns and adjectives. Hence, most of these words are only used in one document. We compute a histogram of the document frequencies (this histogram maps the document frequency to the number of nouns that have this document frequency). The basic idea is that *potentially meaningful* words occur with medium document frequency. Hence, we disregard words that occur in only one document or that occur in more than half of the documents.

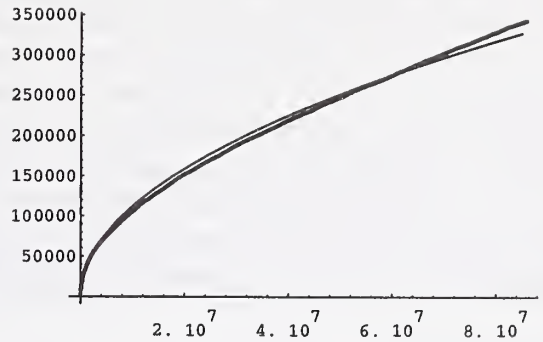


Figure 2: Number of different words vs number of words in a growing document collection (fat line, 4 years Financial Times) and the square root in comparison (thin line).

The output of the whole preprocessing is *one* summary file that contains a table of the relevant words together with an id number and the document frequency, respectively. This file also contains a document surrogate of each document, ie, a collection of the relevant words together with the term frequency of this word in the collection.

The retrieval process is based on this summary file. Owing to the lack of a retrieval system at the time of the experiments no inverted document list was available. So, in order to retrieve documents, the whole summary file had to be scanned, resulting in an inferior query time of 10 seconds per query for the TREC-7 task.

2.2 Complexity and Resources for the Document Preprocessing

Let m be the number of different words in the document collection and n be the number of words in the document collection. The time for preprocessing is predominantly linear in n and the space for internal arrays is predominantly linear in m. Theoretically, the time is of order $O(mn)$, but a clever use of hash-tables or other methods can disguise the m dependency. The preprocessing throughput is around 12 Megabyte/hour of documents on a typical Sparc workstation. It should be borne in mind that the prototype was not tuned for speed and that the algorithms used are highly parallelisable. (Remark: in the meantime, a tuned all-in-one version of the same software achieves a throughput of well over 300 Megabytes/hour.)

3 Processing of the Topics

The topics are processed in a similar way as the documents are preprocessed. The vocabulary of the topics is, however, not limited by document (or topic) frequencies, as is the case with the document collection. The internal structure of the topics are exploited: The title and description part are processed as discussed before, and a corresponding list of relevant words together with their term frequency (how often this word is used as an adjective or noun in this topic) is associated to this topic. This list is called `r-list`.

For the run `ic98san3`, this list of words is matched to the list of relevant words of each document and a ranking number is computed for each document using the set of common words as explained below. The 1000 best-ranked documents are the result of this run, which does not make use of the narrative field of the topics.

The run `ic98san4` re-ranks the 1500 best-ranked documents of the `ic98san3` run. Here two

new lists, the `p-list` and `m-list`, are created per topic: the narrative field is examined and all clauses are analysed w.r.t. the relevance to the topic. This is done using a set of phrases such as "* is relevant, but * not relevant". Adjectives and nouns that appear in clauses relevant to the topic are added to the `p-list`, and those that appear in clauses that are explicitly not relevant to the topic are added to the `m-list`. In the end, the elements of the `r-list` from the `ic98san3` run are added to the `p-list`. Both lists are matched to each of the previously 1500 best-ranked documents and two corresponding ranking numbers are computed per document. Their difference is used to re-rank the 1500 "best" (according to the `ic98san3` run) documents per topic. The corresponding 1000 best-ranked documents are the result of this `ic98san4` run.

4 Relevance Assessment

This section describes how relevance numbers are computed given the `r-`, `p-` and m-lists of a topic. Let n be the number of documents, i be one of the documents and j be a term (in our case lowercase word stems of words that are used as adjectives or nouns in a document or a topic). Let V_i be the vocabulary of document i, let $T^{r,p,m}$ and $t_j^{r,p,m}$ be the vocabulary and the j-term frequency of the corresponding topic lists. Let d_j be the document frequency of term j and t_{ij} be the term frequency of term j in document i. Then,

$$w_i^r := \left(\frac{|V_i \cap T^r|}{|T^r|}\right)^4 \cdot \frac{1}{(1+|V_i|)^2} \cdot \sum_{j \in V_i \cap T^r} t_{ij} t_j^r \log(n/d_j)$$

is the ranking number that is used for run `ic98san3`.

For `ic98san4` the 1500 best-ranked documents of the `ic98san3` run are re-ranked with the ranking number $w_i^p - w_i^m$, where

$$w_i^{p,m} := \left(\frac{|V_i \cap T^{p,m}|}{|T^{p,m}|}\right)^3 \cdot \frac{1}{(1+|V_i|)^2} \cdot \sum_{j \in V_i \cap T^{p,m}} t_{ij} t_j^{p,m} \log(n/d_j).$$

5 TREC Evaluation and Conclusions

TREC assigned an average precision over all relevant documents of 0.1259 to the run `ic98san3` and 0.1333 for `ic98san4`, respectively. These numbers are not very impressive in comparison to other system's performance, and reflect the rather basic structure of the whole retrieval process.

Our experiments were concerned with the *feature reduction* component of a whole system. It is interesting to note that evaluating the description part of the topics increased the average precision slightly, as would be expected from a sensible feature extraction algorithm.

The next natural steps of a better retrieval performance would be to incorporate a standard inverted-document list retrieval to obtain a sensible query speed, optimise the ranking procedure, implement relevance feedback and reassess the system.

References

Brill, E. (1994). Some advances in rule-based part of speech tagging. In *AAAI*.

Zipf, G. K. (1949). *Human Bahavior and the Principle of Least Effort.* Addison-Wesley.

Acknowledgements: This work was supported by the Fujitsu European Centre for Information Technology (FECIT).

Mercure at trec7

M. Boughanem , T. Dkaki, J. Mothe & C. Soulé-Dupuy

IRIT/SIG
Campus Univ. Toulouse III
118, Route de Narbonne
F-31062 Toulouse Cedex 4
Email : {bougha, dkaki, mothe, soule}@irit.fr

1 Introduction

The tests we performed for TREC-7 were focused on automatic ad hoc and filtering tasks. With regard to the automatic ad hoc task we assessed two query modification strategies. Both were based on blind relevance feedback processes. The first one carried on with the TREC6 tests: new parameter values of the relevance backpropagation formulas have been tuned. On the other hand, we proposed a new query modification strategy that uses a text mining approach. Three runs were sent. We sent two runs for the relevance backpropagation strategy: one used long topics (titles, descriptions and narratives) and the other one used titles and descriptions. We sent one run for the text mining strategy using long topics. With regard to the filtering task, we sent runs in batch filtering and routing using both relevance backpropagation and gradient neural backpropagation.

2 Mercure model

Mercure is an information retrieval system based on a connectionist approach and modeled by a three-layered network (as shown in the figure 1). The network is composed of a query layer (set of query terms), a term layer representing the indexing terms and a document layer. 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 tf.idf measure inspired from the OKAPI and SMART term weightings.

Let be :

w_{ij} : the weight of the link between the term neuron N_{t_i} and the document neuron N_{D_j},
tf_{ij} : the term frequency of t_i in the document D_j,
N : the number of documents in the collection,
T : the total number of indexing terms,
n_i : the number of documents containing the term t_i,
$doclen_j$: document length in words (without stop words),
avg_doclen : average document length.

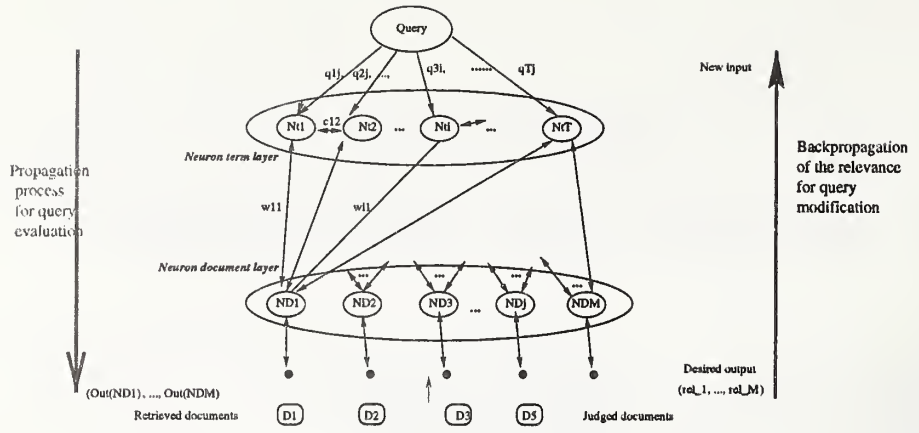


Figure 1: *The Mercure Model.*

- The query-term (at stage s) links are weighted as follows:

$$q_{ik}^{s} = \frac{(1+log(tf_{ik}))*(log(N/n_i)}{\sqrt{\sum_{j=1}^{T}(1+log(tf_{jk}))*(log(N/n_j)^2}} \text{ if } tf_{jk} \neq 0$$

$q_{ik}^{s} = 0$ otherwise

- The term-document link weights are expressed by:

$$w_{ij} = \frac{(1 + log(tf_{ij})) * (h_1 + h_2 * log(\frac{N}{n_i}))}{h_3 + h_4 * \frac{doclen_j}{avg\_doclen}}$$

The query evaluation is based on spreading activation. Each neural node computes an input and spreads an output signal:

1. The query k is the input of the network:

$$Input_k = 1$$

Then, each neuron from the term layer computes an input value from this initial query:

$$In(N_{t_i}) = Input_k * q_{ki}^{s}$$

and then an output value :

$$Out(N_{t_i}) = g(In(N_{t_i})$$

where g is the identity function.

2. These signals are propagated forwards through the network form the term layer to the document layer. Each neuron computes an input and an output value :

$$In(N_{D_j}) = \sum_{i=1}^{T} Out(N_{t_i}) * w_{ij}$$

then,

$$Out(N_{D_j}) = g(In(N_{D_j}))$$

The system output is :

$$Output_k(Out(N_{D_1}), Out(N_{D_2}), ..., Out(N_{D_N}))$$

These output values are then ranked to build the corresponding retrieved document list.

3 Ad hoc experiment and results

3.1 Ad hoc methodology

Our work in the TREC7 automatic ad hoc has been undertaken in two directions. The first one concerns the improvement of the automatic query modification based on document relevance. We tested new weighting formulas and we better tuned different parameters. The second proposition concerns a new investigation in query expansion based on a text mining approach. Both approaches have been performed using blind relevance feedback.

Query modification based on relevance backpropagation

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].
Each document computes its input and output according to its relevance.

$$In(N_{D_j}) = rel_j$$

$$Out(N_{D_j}) = g(In(N_{D_j}))$$

Where,

$rel_j = \frac{Coef\_Rel}{Nb\_rel}$ for relevant document
$rel_j = \frac{Coef\_NRel}{Nb\_Nrel}$ for non-relevant document

and

$Coef\_Rel$, $Coef\_NRel$: relevance coefficient of the documents (positive for relevant and negative for non-relevant documents),
Nb_rel, Nb_Nrel : number of relevant and non-relevant documents respectively.

Notice that the relevance is automatically assigned to the documents: only the top 12 retrieved documents are considered as relevant.

Finally, this activation is backpropagated to the term layer according to the formulas:

$$In(N_{t_i}) = \sum_{j=1}^{N} (Out(N_{D_j}) * w_{ij})$$

$$Out(N_{t_i}) = g(In(N_{t_i}))$$

New query-term link weights are then computed according to this formula :

$$q_{ik}^{s+1} = M_a * q_{ik}^{s} + M_b * Out(N_{t_i})$$

The new query is evaluated the same way the initial query is.

The parameters we have chosen for the TREC7 experiments are : $h_1 = .8$, $h_2 = .2$, $h_3 = .8$, $h_4 = .2$.

The other parameters used in the relevance backpropagation are : $Coef\_Rel = 1$, $Coef\_NRel = -.75$, $Nb\_rel = 12$, $M_a = 2$, $M_b = .75$. Both MerAdRbtnd and MerAdRbtd runs used this technique.

Query modification based on text mining approach

The text mining approach consists in analyzing a sub-set of the retrieved documents in order to expand the initial query. A blind process as been chosen for this experiments as well and the top 12 documents constitute the set of documents to mine for each query. This number has been chosen according to previous experiments in TREC. The mining was performed using the Tétralogie system [5]. This system includes advanced information extraction functionalities. It implements document reduction (under the form of contingency tables) and factorial analysis mining functions. Given a set of items expressed in a n-dimensional space, the factorial analysis methods reduce the data dimensionality into a space which is the most important to characterize the items [1].
According to the experiments performed for TREC7, the document set mining was used to chose the terms to be added to the initial query as follows :

1. The most frequent terms have been extracted from the analyzed set of documents,
2. The co-occurrence value of these extracted terms and the query terms have been computed,
3. This crossing table was used as an input to a Correspondence Factorial Analysis (CFA),
4. Then, the terms that have the highest weight according to the AFC have been kept.

This technique was intended to determine the terms which are the most characteristic of the query terms according to the analyzed document set.

3.2 Ad hoc results and discussion

Three automatic runs have been submitted : MerAdRbtnd (long topic : title, description and narrative) and MerAdRbtd (title and description) based on backpropagation and MerTetAdtnd (long topic) based on text mining. These runs were based on a completely automatic processing of TREC queries and automatic query expansion based on "blind" feedback . Table 1 compares our runs against the published median runs.

TREC results			
Run	Best	$\geq$median	$<$ *median*
MerAdRbtnd	0	38	12
MerTetAdtnd	0	34	16
MerAdRbtd	0	30	20

Table 1: Comparative automatic ad hoc results at average precision

Most of the runs are above the median. These results show that we obtained better results on the long topics than using the titles and descriptions only.

The average results obtained using the blind relevance feedback were less good than the one using no query reformulation. In fact, this observation is not uniform as some query results were improved by the reformulation. A deep analysis of these results could lead to a better understanding.

Type	Run	average precision	Exact precision
Long topics	basic search	.2290	.2760
"	MerAdRbtnd	.2278 (-.5%)	.2746 (-.5%)
"	MerTetAdtnd	.2237 (-2.3%)	.2617 (-.5%)
Titles-Descriptions	basic td	.1918	.2380
"	MerAdRbtd topic	.1918 (0%)	.2380 (0%)

Table 2: Ad hoc component results - 50 queries

4 Batch Filtering and Routing Experiment

The batch filtering and routing experiments were performed using Mercure system as described above.

Experiment :

The technique we used to build the batch and the routing queries is based on the relevance backpropagation process presented above. The AP88 documents were used as training data. The filtering algorithm starts with an initial query, built from all the parts of the topic, and its AP88 relevance judgments (positive and negative). Relevant and non relevant documents computes a relevance value that is backpropagated to the query. The query-term links are then modified and a query evaluation process is done through the new links. This process is repeated and a new learned query is built at each iteration.

A pool of queries was then selected. For the routing task, the queries showing having the best average precision in the training data were selected as routing queries. For the batch filtering, the TREC standard output file of each query was analyzed to build an output file containing:

$$< topic >< func >< value >< thresh >< rank >< prec >< recall >$$

as it has been done in [7]. The document activation weights that maximize each function F1 and F3 were then found and selected as thresholds. Then, the queries corresponding to these thresholds were selected and tested against the test data.

Results and discussion :

Routing task : The top 1000 retrieved documents were submitted for each routing query. One run was submitted (MerRou). The table 3 shows the comparative routing results at average precision.

TREC Routing			
Run	Best	$\geq$median	$<$ *median*
MerRou	5	13	37

Table 3: Comparative routing results at average precision

Batch Filtering : Two runs were submitted. One based on utility-[$F1$], it is labeled MerBF1. The other one based on the utility-[$F3$] and labeled (MerBF3). The table 4 lists the comparative batch results.

TREC batch filtering			
Run	Best	$\geq$median	$<$ *median*
MerBF1	8	22	28
MerBF3	8	18	32

Table 4: Comparative batch filtering results for F1 and F3 functions

5 Conclusion

Our main goal this year was to perform completely automatic runs. We assessed two query modification techniques. The first one aimed at tuning the parameter values of the backpropagation formulas used in TREC6. Whereas the second one trained a text mining approach using Tétralogie system. It is difficult to draw any firm conclusion with respect to the results obtained this year. In average, the basic searches performed using Mercure have led to better results than the two query reformulation methods we have experimented. Fortunately, some reformulated queries performed better than initial queries. More attention will be paid next year to identify these queries and to better tune the two techniques. In addition to that, we plan to deeply investigate query reformulation using Genetic Algorithms. In fact, our first investigations in GA on small databases are encouraging [3].

References

[1] J.P. Benzecri, *L'analyse de données, Tome 1 et 2.* Edition Dunod, 1973.

[2] M. Boughanem & C. Soule-Dupuy, *Query modification based on relevance backpropagation*, Proceedings of the 5th International Conference on computer-assisted information searching on Internet (RIAO'97), Montreal (Canada), June 1997.

[3] M. Boughanem & L. Tamine, *Reformulation automatique de requête basée sur l'algorithmique génétique*, Proceedings of the 15th National Conference INFORSID'97, Toulouse (France), June 1997.

[4] C. Buckley & al, *Query zoning : TREC'5*, Proceedings of the 5th International Conference on Text REtrieval TREC5, Harman D.K. (Ed.), NIST SP 500-236, Nov. 1996.

[5] T. Dkaki, B. Dousset & J. Mothe, *Mining information in order to extract hidden and strategical information*, Proceedings of the 5th International Conference on computer-assisted information searching on Internet (RIAO'97), Montreal, June 1997.

[6] J.Hertz, A. Krogh & R. G. Palmer, *Introduction to the theory of neural computation* Addison Wesley, March 1992.

[7] S. Robertson and al, *Okapi at TREC-6*, Proceedings of the 6th International Conference on Text REtrieval TREC6, Harman D.K. (Ed.), NIST SP 500-236, Nov. 1997.

Indexing Using Both N-Grams and Words

James Mayfield and Paul McNamee
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Road
Laurel, MD 20723-6099 USA
James.Mayfield@jhuapl.edu
Paul.McNamee@jhuapl.edu

Goals

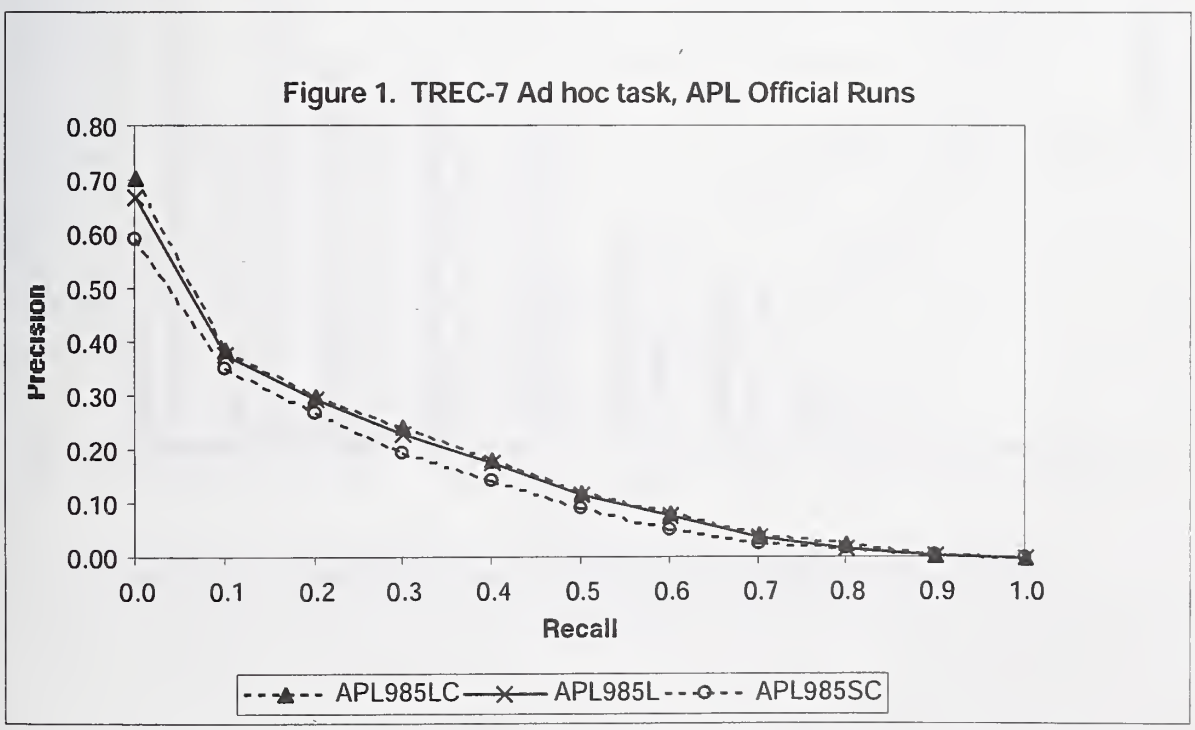
The Johns Hopkins University Applied Physics Laboratory (JHU/APL) is a first-time entrant in the TREC Category A evaluation. The focus of our information retrieval research is on the relative value of and interaction among multiple term types. In particular, we are interested in examining both words and n-grams as indexing terms. The relative values of words and n-grams have been disputed; to our knowledge though, no one has previously studied their relative merits while holding all other aspects of the system constant.

Approach

The Hopkins Automated Information Retriever for Combing Unstructured Text (HAIRCUT) system was built to explore the use of multiple term types. The system is implemented in Java for ease of development, portability, and of course blazing speed. It implements a vector model, using cosine as its similarity measure. Terms are usually weighted by Okapi BM 25 [Walker *et al.*, 1998], which is a variant of TF/IDF weighting that boosts the scores of longer documents. Normal TF/IDF and plain TF weightings are also supported. Cosines can be computed either relative to the origin, or relative to the corpus centroid. Terms that appear in a high percentage of documents are effectively stop-listed.

HAIRCUT performs rudimentary preprocessing on queries to remove stop structure [Allan *et al.*, 1998], *e.g.*, affixes such as "... would be relevant" or "relevant documents should...." Other than this preprocessing, queries are parsed in the same fashion as are documents in the collection.

Figure 1. TREC-7 Ad hoc task, APL Official Runs



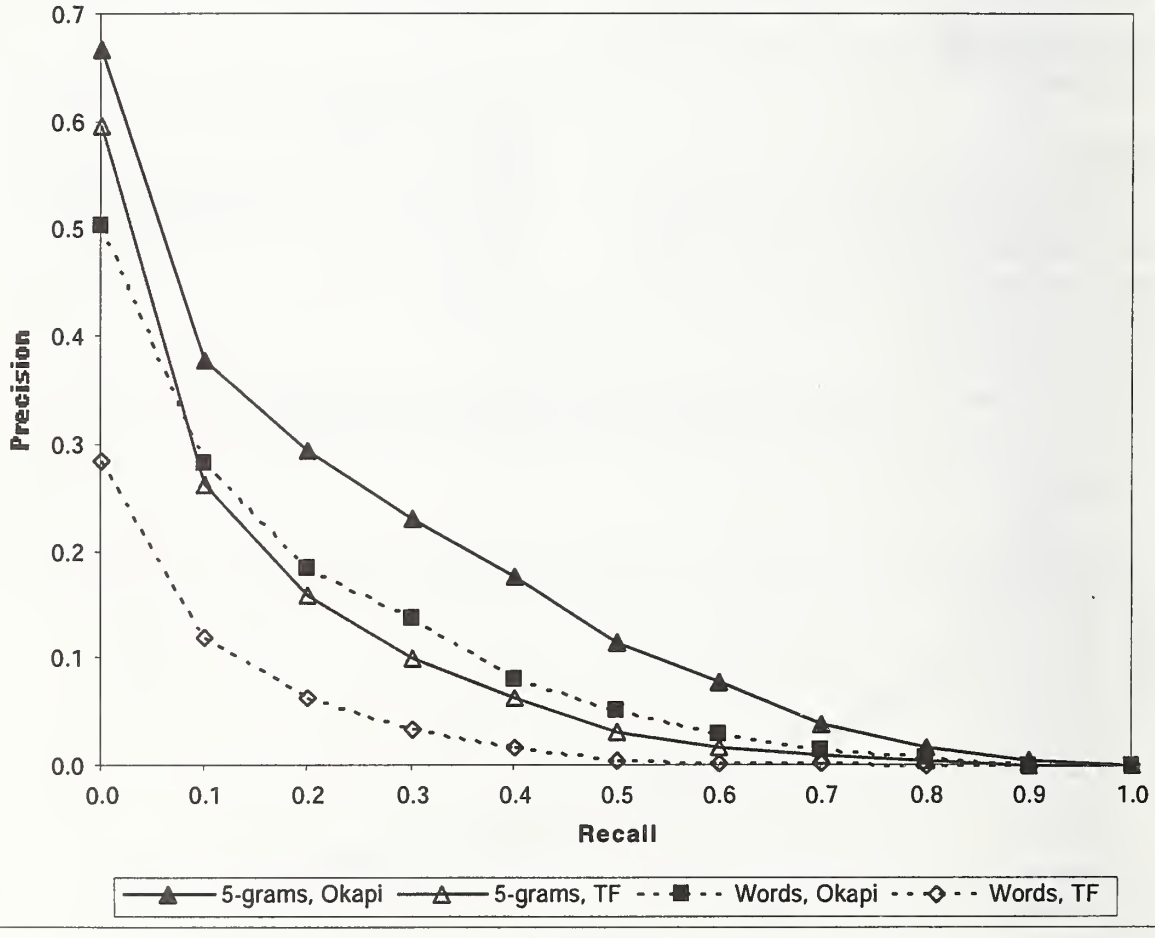
We conducted our work on a pair of Sun workstations, an UltraSPARC 2 with two 300 MHz processors and a 4-node Ultra Enterprise 450 server. Both workstations had 512 MB of physical memory and access to 26 GB of shared hard disk space.

The HAIRCUT system comprises approximately 28,000 lines of Java code.

For TREC-7 we tested two types of terms: words and 5-grams. After eliminating punctuation, downcasing letters, and mapping numbers to a single digit, a word was any remaining blank-delimited sequence of characters.

For n-grams we used 5-grams formed from the same character stream used for selecting words, but with common words replaced by a single character. Although Java uses the 2-byte Unicode format to represent strings, HAIRCUT represents terms using byte sequences. Since the input stream is downcased, all uppercase letters and certain of the Latin-1 characters can be used as replacements for common words such as "the," "with," *etc.* This has an effect of lengthening n-grams that span common words. For example, the phrase "statue_of_liberty" might produce the 5-gram "e_¢_l" in HAIRCUT, where the common word 'of' has been replaced by the single character '¢'.

Figure 2. TREC-7 Ad hoc task, Long Topics, Words *vs.* 5-grams



Ad hoc Results

JHU/APL submitted three ad hoc runs. Our first run, labeled APL985L, was a baseline run that used 5-grams as indexing terms, and used no relevance feedback. Runs APL985LC and APL985SC combine two separate query runs; a 5-gram run and a word-based run that used automated relevance feedback. APL985L and APL985LC used the title, description, and narrative portions of the topic statements, while APL985SC only made use of the title section.

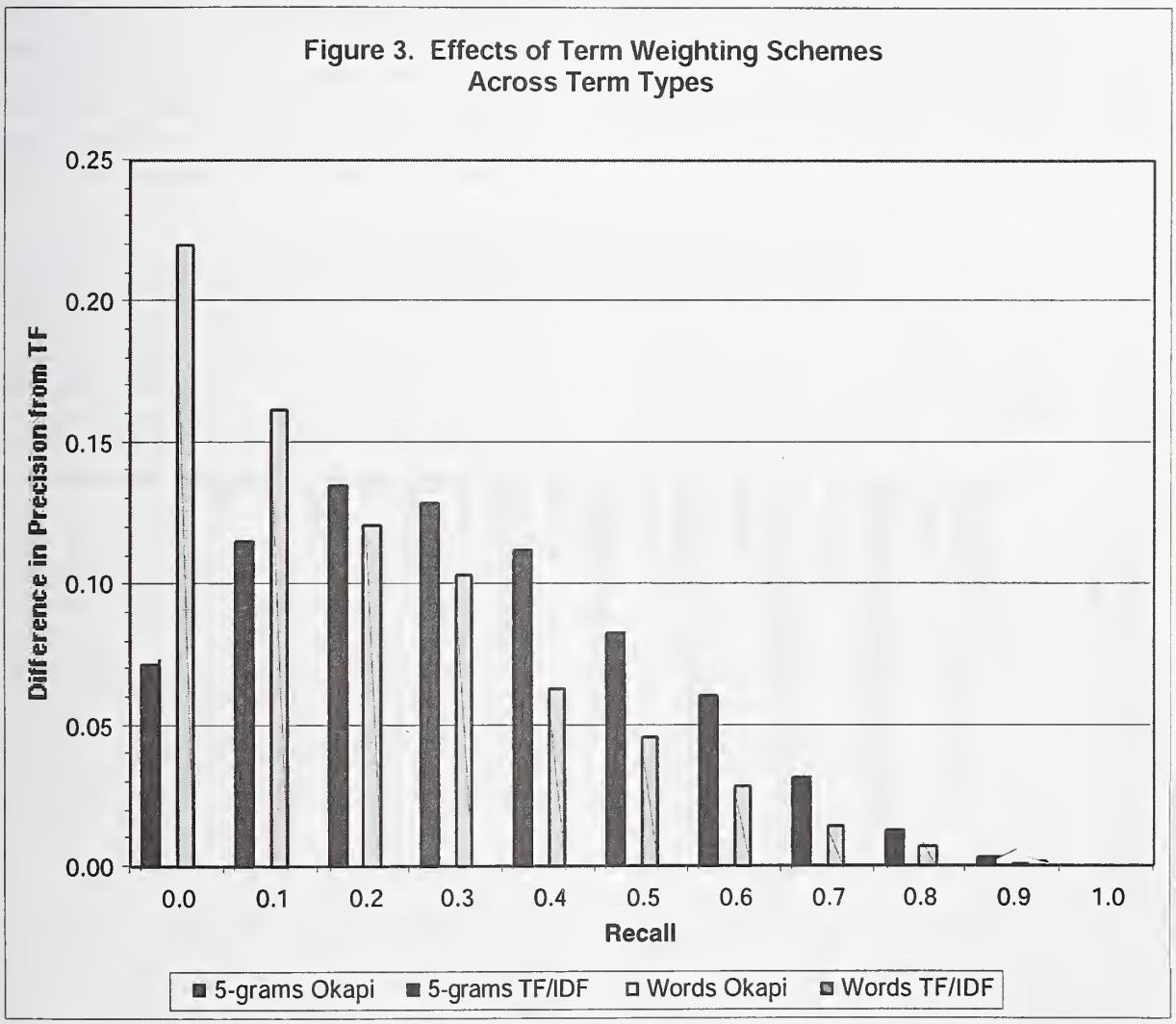
This year we concentrated our efforts on techniques that improve precision at low recall levels. Our official results are shown in Figure 1. The APL985L and APL985LC runs are similar, showing that our relevance feedback techniques were ineffectual on queries composed of the title, descriptive, and narrative portions of the topic statements.

N-Grams vs. Words

One of our main goals in developing HAIRCUT has been to compare the relative merits of n-grams and words as indexing terms when all other aspects of the system are held constant. To that end, we scored a set of word-based runs against the TREC-7 relevance assessments.

Figure 2 depicts the results of our experiments comparing the use of unstemmed words against 5-grams. Surprisingly good performance was obtained from the 5-grams. In fact, 5-grams using term frequency weighting do about as well as words using Okapi BM 25 term weighting. None of these runs uses relevance feedback.

Figure 3. Effects of Term Weighting Schemes Across Term Types



Other Experiments

In addition to comparing n-grams and words as term types, we also wanted to investigate the effects of alternative term weighting schemes and the effects of using cosine with corpus centroid subtraction as a measure of similarity. We performed a series of runs on the TREC-7 topics to explore the variability in performance attributable to these alternative approaches.

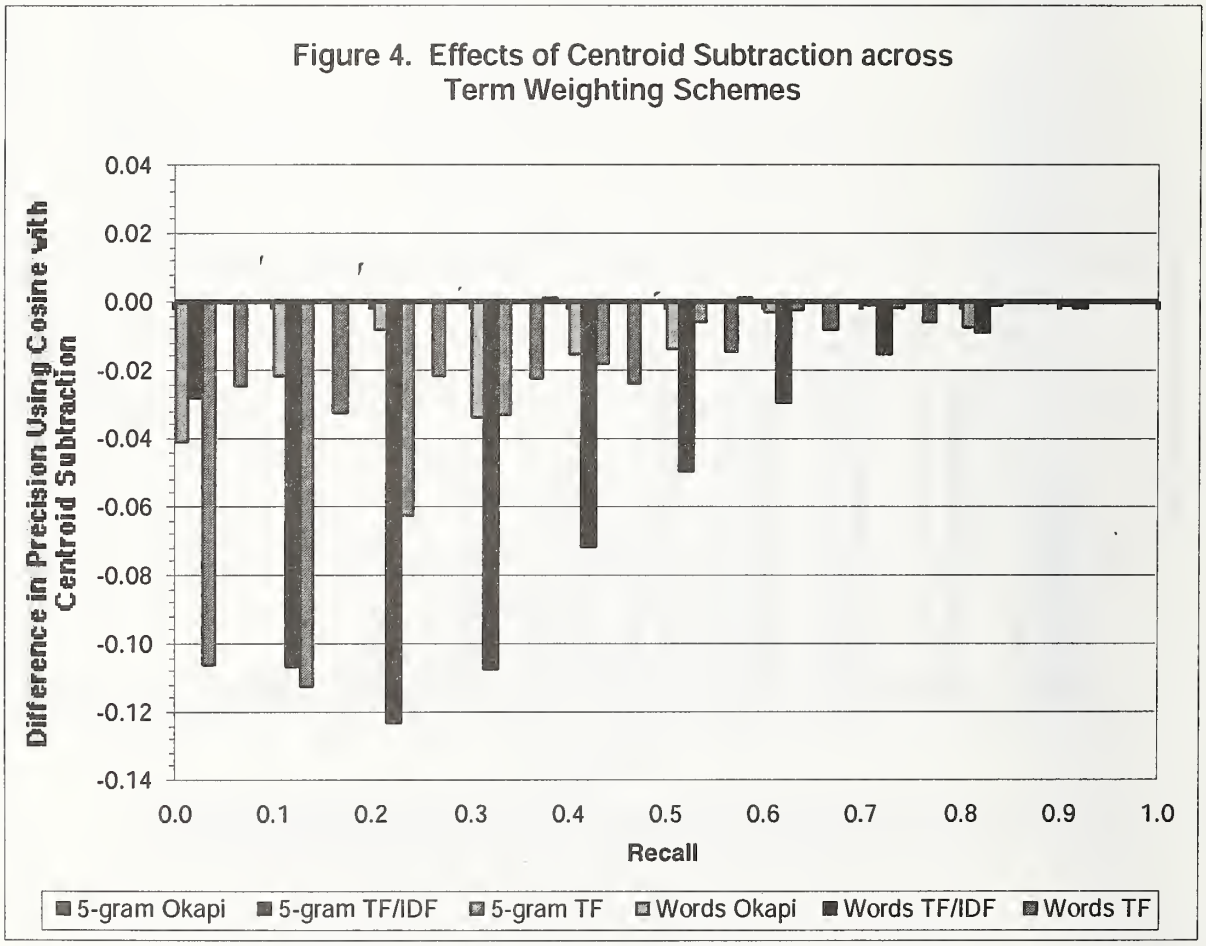
As expected, both Okapi-style term weighting and TF/IDF term weighting surpass the use of term frequencies alone. Figure 3 illustrates the gain from the use of Okapi and TF/IDF. Figure 4 shows the effects of adding centroid subtraction to term weighting.

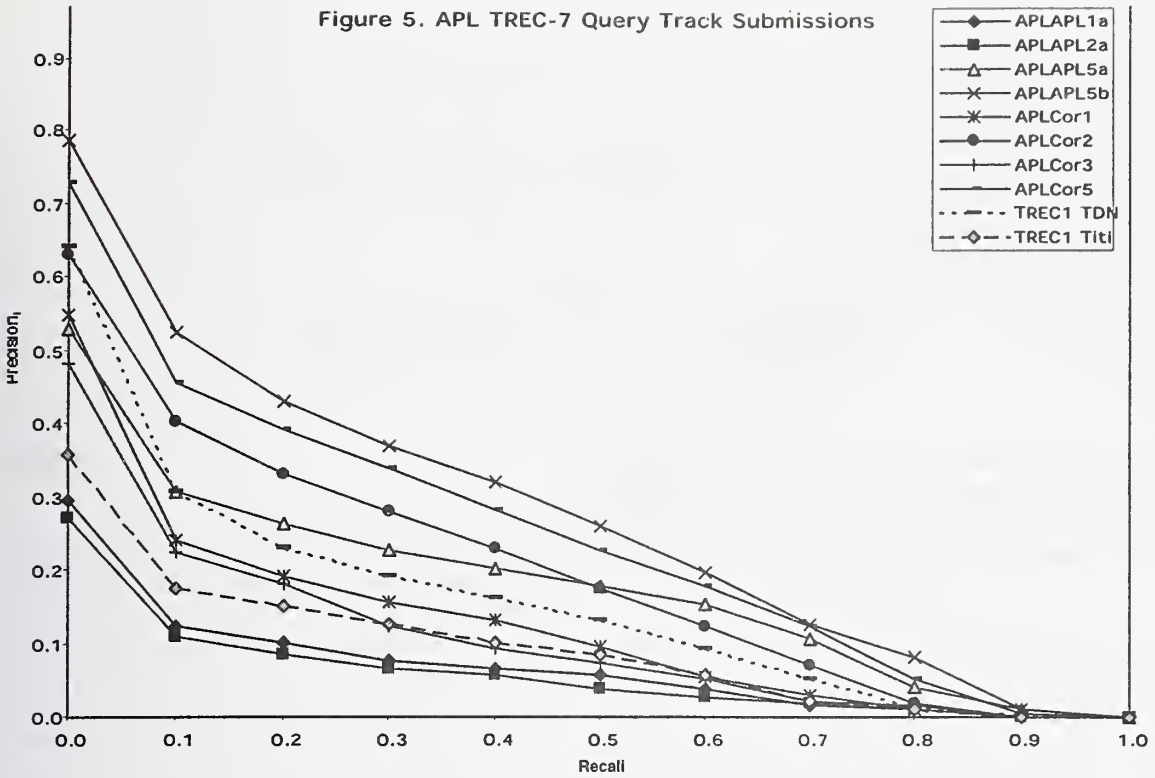
Four observations are worth making about these data. First, Okapi and TF/IDF term weighting are very nearly equal in terms of their improvement from TF weighting. Secondly, the use of Okapi and TF/IDF appears to provide more of a gain for word-based runs than for n-gram runs at lower recall levels. Correspondingly, the n-gram based runs appear to receive more advantage from the alternative term weightings at higher recall levels. Thirdly, these results are based on unofficial runs, for which the top one hundred documents are not guaranteed to have been assessed. The average number of documents in the top one hundred that were assessed in the Okapi and TF/IDF runs ranges from 92% to 95%, but the TF runs averaged only 78% for n-grams and 66% for words. It is conceivable that some of the difference reported here stems from differences in level of assessment. Finally, we have yet to see circumstances under which the use of centroid subtraction is justified.

Query Track

JHU/APL also participated in the query track. We generated four query sets. Two (APL1a and APL2a) were generated by hand, by reading the narrative version of each source query and generating a title query and a description query for each. The third (APL5a) was created using a

Figure 4. Effects of Centroid Subtraction across Term Weighting Schemes





variant of the mutual information statistic to extract important terms from the top 75 documents retrieved for the source query. The last (APL5b) used the same statistic to extract important terms from the query track training set. All terms in the last two query sets were unstemmed words; we did not anticipate that other systems could make use of n-grams.

Our goal this year was simply to assess the variability in precision found across queries. To that end, we used a single system configuration to process eight of the nine query track query sets (one query set from Cornell included a Boolean component, which our system cannot handle). This configuration used unstemmed words as terms, and cosine based at the origin to gauge document similarity. No relevance feedback was used.

Our results, shown in Figure 5, exhibited tremendous variability in result quality across the eight query sets. The best results were obtained from the two query sets developed using training data. The query sets that we generated by hand after reading the source narratives fared worst. Figure 5 also shows our results on the original TREC-1 queries, both title-only and title-description-narrative. We are currently trying to assess the relative contributions of vocabulary choice, lack of assessments, and system configuration to this range of results.

References

[Allan *et al.*, 1998] James Allan, Jamie Callan, W. Bruce Croft, Lisa Ballesteros, Don Byrd, Russell Swan, and Jinxi Xu, 'INQUERY does battle with TREC-6.' In E. M. Voorhees and D. K. Harman, eds., *The Sixth Text REtrieval Conference (TREC-6).* NIST Special Publication 500-240, pp. 169-206, 1998.

[Walker *et al.*, 1998] S. Walker, S. E. Robertson, M. Boughanem, G. J. F. Jones and Karen Sparck Jones, 'Okapi at TREC-6, Automatic ad hoc, VLC, routing, filtering and QSDR.' In E. M. Voorhees and D. K. Harman, eds., *The Sixth Text REtrieval Conference (TREC-6).* NIST Special Publication 500-240, pp. 125-136, 1998.

DSIR: the First TREC-7 Attempt

Arnon Rungsawang
fenganr@ku.ac.th
Department of Computer Engineering, Faculty of Engineering
Kasetsart University, Paholyothin Rd., Bangkok, Thailand.

Abstract

This paper describes our first large-scale retrieval attempt in TREC-7 using DSIR. DSIR is a vector space based retrieval system in which semantic similarity between words, documents and queries, is interpreted in terms of geometric proximity of vectors in a multi-dimensional space. A co-occurrence matrix computed directly from the collection is used to build the underlying semantic space. We have implemented DSIR on a cluster of low-cost PC Pentium-class machines, and chosen the PVM message-passing library to manage our distributed DSIR version. Although our first adhoc retrieval results are quite poor in terms of recall-precision measure, we believe that more work and experiments have to be explored in order to obtain more promising retrieval performance.

1 Introduction

For our first large-scale text retrieval attempt in TREC-7 adhoc experiments, we use our own retrieval artifact, called "DSIR", a full-text retrieval system developed on a cluster of PC Pentiums at the department of Computer Science, Kasetsart University[1]. DSIR stands for "Distributional Semantics based Information Retrieval", a retrieval model based on vector space. In this model, the contents of retrievable objects, such as words, phrases, sentences, documents, are represented in a unified way by multi-dimensional vectors. These vectors are derived from a co-occurrence matrix computed on textual collection being indexed. Semantic proximity among objects is then simply interpreted in terms of geometric proximity between corresponding vectors in the multi-dimensional space, called the "meaning space".

Former source codes of DSIR algorithm are written in C, and Perl programming languages, running on a standard Unix machine. To achieve TREC-7 experiments, we reexport the distributed DSIR to a cluster of low-cost PC Pentiums, running Linux operating system. Each PC is hooked together through a low-cost Ethernet local area network. New distributed version is developed using PVM[2] [2], a widely used message-passing software package.

We organize this paper in the following ways. Section 2 gives a brief overview of fundamental concepts constituting DSIR model. Section 3 provides more detail about distributed DSIR implementation. Section 4 presents TREC-7 retrieval experiments and gives the results. Finally, section 5 concludes this paper.

2 DSIR Model

2.1 Basic Concept

Research in distributional semantics concerns with the utilization of distributional information extracted from textual collections to represent the meaning of linguistic

[1] Indeed, this retrieval model has been devised during the author's Ph.D. study at ENST-Paris, in France [8].

[2] Parallel Virtual Machine.

entities, e.g. words, phrases, sentences, documents. We assume that there exists a correlation between meaning of a word and its observable distributional characteristics within particular contexts in a given language [6]. These distributional characteristics can either be "occurrences" of that word itself, or its "co-occurrences" with the other words appearing within the documents.

In this retrieval approach, we are especially interested in using word contexts to characterize the meaning of a word [9, 10, 7, 8]. In general, every word has meaning. Each contributes its own meaning, according to its occurrence, to the whole content of the document in which it appears. Here, we choose "word" as an elementary entity that holds the meaning. We consider tokens of length at least two characters, beginning with an alphabet, excluding those words in a pre-defined non-significant word list, as words (i.e. keywords or index terms) that constitute a set of vocabulary chosen for indexing a document collection. Following the "distributional structure" definition of Harris [3, 1]:

> "*The distribution of an element will be understood as the sum of all its environments.*",

we denote the "context" of a word as a knowledge concerning its usage, i.e. how that word is used with the other words in order to compose the content of a document. We characterize word contexts on the basis of "co-occurrence statistic". This choice is made because it is a source of distributional information that is easily extracted from a document collection.

We then define the co-occurrence statistic of a word as the number of times that word co-occurs with one of its neighbors within a pre-defined boundary. We denote this boundary, the "distributional environment". Possible distributional environments can be sentences, paragraphs, sections, whole documents, or windows of k words.

The definition of this distributional environment is essential in our retrieval model. It is used to delimit the scope of the contexts which are of interest. Co-occurrences measured within distributional environment defined by a sentence will let the "local" context information of words written in the documents to be observed. On the other hand, co-occurrences measured within the environment of a paragraph or the whole document will let the "global" context information of words to be examined. A window of k words can be used to extract the information between local and global contexts.

A specialized case of representing a word based on its contexts is that true synonyms will have identical contexts. Near-synonyms or related words will have just similar contexts. On the other hand, in case of a polysemy, its contexts are different because its meanings are in general invoked with different sets of words in different contexts. Representing the contents of documents on the basis of word contexts rather than just word occurrences thereby makes this retrieval model different from other standard keyword-based approaches. Documents in the collections should be retrieved without difficulty even if a query is composed of synonyms or related terms.

In our computational model, we use a co-occurrence matrix illustrated in Figure 1 to represent distributional information extracted from a document collection. Each row in this matrix represents the distribution of a word x_i, while each column represents the distribution of another word y_j which appears close to x_i. The intersection between row i and column j, i.e. the m_{ij}, records the co-occurrence frequency between x_i and y_j extracted from a document collection.

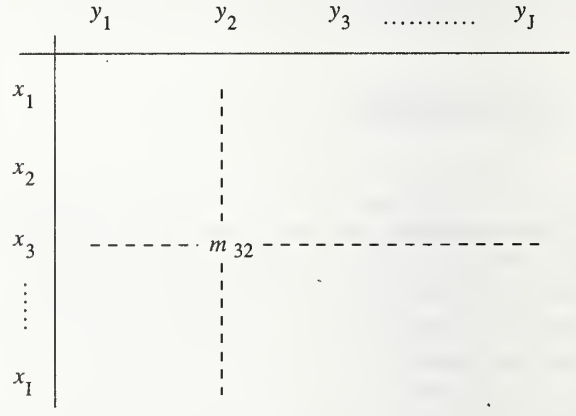


Figure 1: Co-occurrence matrix.

To represent meaning of a word according to its contexts by a vector, we depict each word distribution x_i corresponding to row i in the co-occurrence matrix by a vector $\vec{v}(x_i)$ using the sequence of $\{m_{ij} \mid j \in J\}$ as its coordinate. Each dimension of this vector is associated to word y_j representing the column of the matrix. We hereafter call this vector, "co-occurrence vector".

Therefore, if a co-occurrence matrix built from a document collection consists of I rows representing I word distributions, and J columns representing J word distributions, the meaning of these I words can, by this way, be projected onto a vector space of J dimensions by I corresponding co-occurrence vectors. We name hereafter this vector space, "meaning space". Figure 2 is supposed to illustrate the first three vector representations corresponding to words x_1, x_2 and x_3 in the first three-dimensional meaning space associated with words y_1, y_2 and y_3, derived from a document collection.

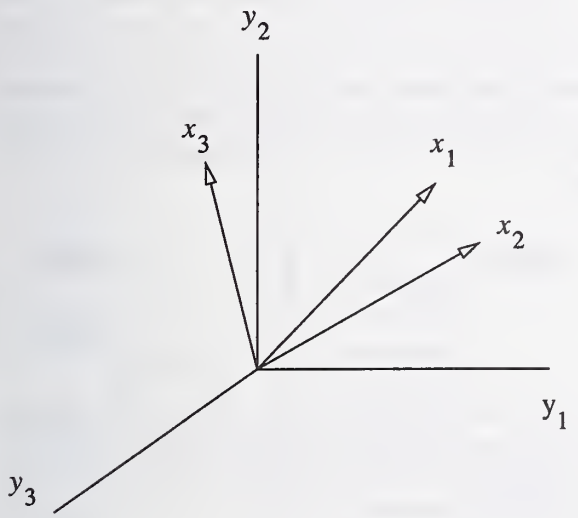


Figure 2: Vector representation of words in a meaning space.

2.2 Document Representation

A full-text document consists of words. Since we have already represented the meanings of words as vectors in the meaning space, our problem now is limited to define the vector representation of a document on the basis of the co-occurrence vectors of words of which that document is composed. We propose to define the vector representation of a document using the weighted vector sum of the co-occurrence vectors corresponding to words occurring in that document. Formally, if we choose I words, and J features, to index a collection of N documents, a document vector n is written by:

$$\vec{v}(d_n) = \Big(\sum_{i=1}^{I} w(f_{ni}) m_{i1}, \sum_{i=1}^{I} w(f_{ni}) m_{i2}, \sum_{i=1}^{I} w(f_{ni}) m_{i3}, \ldots, \sum_{i=1}^{I} w(f_{ni}) m_{iJ} \Big) \quad (1)$$

where w(f_{ni}) is the weighting function addressing the importance of the word i in document n. Since a query can be considered as a specific document, its vector representation is derived in the same way as those of documents. Figure 3 below illustrates our document and query vector representations.

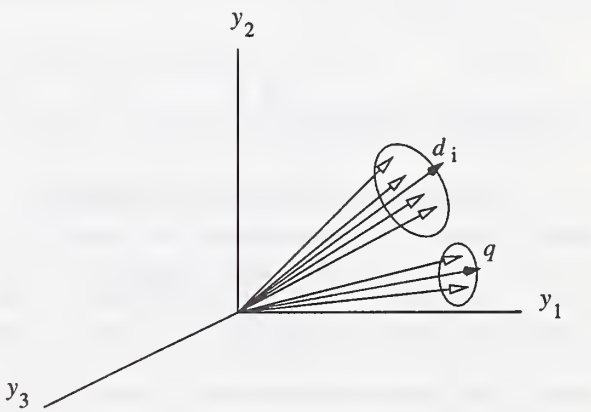


Figure 3: Document representation.

The document vector representation defined in Equation (1) can be seen as an approximation of semantic content of a document, because the (weighted) vector sum averages

the direction of a set of vectors corresponding to words constituting that document. The intuition underlying this proposition is that a given document is composed of several words corresponding to different topics. If at least some of the words in a document are frequently used to described what the current topic is about then their corresponding co-occurrence vectors will pull the final vector sum towards the direction of that topic.

We also include the document vector components derived from the conventional vector space retrieval model [11] in our retrieval model. If we define $\vec{v}_{\text{DS}}$ as the component vector written in Equation (1), and $\vec{v}_{\text{VS}}$ as the component vector conventionally derived from the standard vector space method, our final document vector representation can be written as follows:

$$\vec{v}_{\text{DSIR}} = \mathcal{H} \,.\, \vec{v}_{\text{DS}} \;+\; (1 - \mathcal{H}) \,.\, \vec{v}_{\text{VS}} \qquad (2)$$

The $\mathcal{H}$ parameter, which we call "hybrid parameter", takes the real value between 0.0 and 1.0. When $\mathcal{H}$ is defined = 1.0, each document vector just takes the DS component vector. On the other hand, when $\mathcal{H}$ is defined = 0.0, each document vector is derived from conventional vector space retrieval model.

2.3 Document Retrieval

Since words, documents, and queries are represented as vectors in the same vector space, the basic retrieval operation in this retrieval model is then very simple; the query vector is compared to every document vector, and the documents whose vectors locate close to that query vector in the meaning space are presented to the user as relevant answer. These documents are returned in decreasing order of their closeness.

In addition, other similarity comparison can be obtained as well. For example, it is easy to combine traditional keyword matching method during any retrieval operation since each keyword also has a corresponding co-occurrence vector in the same meaning space. The user can first use his keyword(s) as query to filter for ranked documents which locate close to that keywords, and then select one (or several) of them as his new query to find the closest remaining documents of interest. In the same way, traditional relevance feedback [4] can easily be integrated into this retrieval model as well. During a retrieval process, the user can choose certain terms or documents (with weights) so that their vectors can simply be added up to the query vector to search more documents in the collection.

If we assume that there are chosen J distinct features for representing documents in a collection, a given document d_n can then be written as a J-dimensional vector of the form:

$$\vec{v}(d_n) = (d_{n1},\, d_{n2},\, d_{n3},\, \ldots\,,\, d_{nJ}), \qquad (3)$$

where d_{nj} represents the j^{th} element of document vector n. In the same way, a vector representation of a given query q can be written as a J-dimensional vector of the form:

$$\vec{v}(q) = (q_1,\, q_2,\, q_3,\, \ldots\,,\, q_J), \qquad (4)$$

A typical vector similarity measure that we use in this retrieval model is the cosine similarity function. This function represents the cosine of the angle between vectors of query q and document d_n in a J-dimensional vector space, which is written by:

$$\texttt{sim}(\vec{v}(q), \vec{v}(d_n)) = \frac{\sum_{j=1}^{J} q_j \times d_{nj}}{\sqrt{\sum_{j=1}^{J} (q_j)^2 \times \sum_{j=1}^{J} (d_{nj})^2}} \qquad (5)$$

3 DSIR Implementation

DSIR system consists of 4 parts; document preprocessing, co-occurrence matrix computation, document vector derivation, and document retrieval. We use a small Perl routine to arrange TREC adhoc collections to be ready for document preprocessing. Document preprocessing in DSIR integrates both Porter and Lovin stemmers, including standard stopword elimination.

Distributed-DSIR implementation uses master/slave or pool of tasks programming style on PVM platform [2]. During co-occurrence matrix computation, a big word-by-word co-occurrence matrix is partitioned into small portions, each can be fit in physical memory of n PC Pentium machines (see the machine configuration in Figure 4). Due to the fact that we have only one big local harddisk, each machine reads and performs co-occurrence computation on TREC collections via NFS[3]. A scheduler (or master), the machine at which the TREC collections is located, has responsible to manage the messages between machines in the pool. Finally, that scheduler accumulates all co-occurrent matrix portions from other machines and writes them out to its local disk.

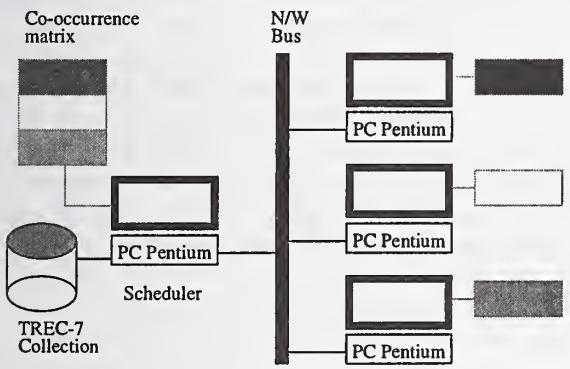


Figure 4: Distributed-DSIR machine configuration during co-occurrence matrix computation and document vector derivation.

[3]Read and write large amount of data via Network File System is one of the bottle-neck problems in our current distributed-DSIR implementation.

During document vector derivation, we still use the same machine configuration illustrated in Figure 4. The big co-occurrence matrix is partitioned, each small portion is distributed through other machine. The main scheduler then reads each TREC-7 document from the collections to see which co-occurrence vectors are needed to be retrieved from other machines to compose that document vector. A careful design of caching strategy during document vector derivation is necessary in order to reduce message passing between scheduler and other machine in the pool in order not to saturate the network bus.

Distributed document retrieval algorithm in DSIR is quite straightforward (see the machine configuration in Figure 5). Each machine in the pool, called "retrieval engine", reads portion of document vector into its main memory, and waits for retrieval command from the central scheduler. The central scheduler reads each query vector, and distributes it to every retrieval engine. Each retrieval engine retrieves and ranks its document vectors, the ones which are close to the query vector are ranked first, and send its ranking back to scheduler. Note that during this phase, retrieval engines perform their retrieval tasks in parallel. Then scheduler accumulates all rank lists from retrieval engines, and performs the final ranking scores.

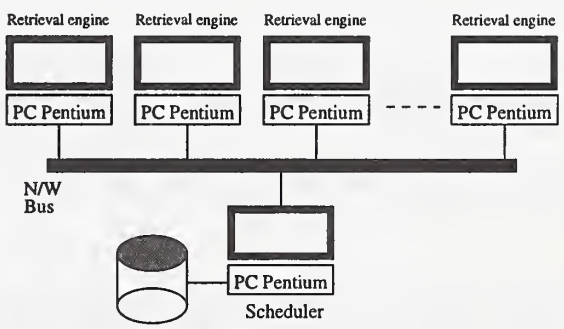


Figure 5: Distributed-DSIR machine configuration during document retrieval.

4 Experimental Results and Discussion

We participate quite lately our TREC-7 adhoc experiments, i.e. in May 1998. That means we have only 3 months before the official deadline to prepare and scale-up DSIR on our PC cluster to perform this large-scale text retrieval task. DSIR adopts standard SMART stoplist, and uses Lovin stemmer to pre-process all adhoc documents. For each run, two term sets are chosen by document and occurrence frequency criterion to build a co-occurrence matrix. We also applied the chi-square correcting weight, which we call "spatial transformation" [8], to every entry in the co-occurrence matrix. That means, each co-occurrence entry m_{ij} is transformed to $m'_{ij} = (\frac{1}{r_i \sqrt{c_j}})\, m_{ij}$, where $r_i = \sum_j m_{ij}$ is defined as row total, and $c_j = \sum_i m_{ij}$ is defined as column total.

Document vectors are computed using formulae explained earlier in Equations (1) and (2). DSIR uses 'aaa.bbb' SMART weighting style [5] during indexing the documents. Query vectors are calculated in the same way, using words found in "title" and "description" field of the topics 351-400. We submitted two adhoc run; dsir07_a01 and dsir07_a02. Table 1 gives the values of different DSIR parameters, and results.

Run	Index	Features	$\mathcal{H}$	Weighting	Av. P
DSIR07_a01	11488	566	0.05	ntc.atc	0.0117
DSIR07_a02	15567	566	0.05	ntc.atc	0.0135

Table 1: Indexing parameters used in TREC-7, and results.

Considering our first official testing of DSIR over the TREC-7 collection set, we found that the recall/precision results were very bad. This level of performance is far below DSIR's typical performance tested over the old standard, very small size collections such as Cranfield and Time. We look forward to getting a more complete set of experiments and to spending more time understanding the situation in which DSIR had difficulty in identifying relevant documents. We believe that the word contexts that DSIR derived from the co-occurrence matrix built directly from the whole TREC-7 collections is confused by several domains that are specific to those TREC-7 collections themselves. DSIR should work better when derived word contexts are learned from a specific domain of interest. Thus, we plan to use DSIR to index and search TREC-7 collections separately so that each semantic space (i.e. meaning space) has been derived from word co-occurrences more correctly with respect to a certain domain.

5 Conclusion

In this paper, we introduce our DSIR retrieval system that has been tuned to perform large-scale text retrieval in TREC-7 adhoc track. DSIR is a distributional semantics based retrieval model in which semantic proximity is derived from a co-occurrence matrix calculated from the textual collection being indexed. Words, documents, and users' queries, are represented in a unified way by vectors in a multi-dimensional space. Retrieval is performed on the basis of the geometric proximity between vectors representing documents and the user's query; documents whose corresponding vectors are closed to that of query are returned as relevant answer.

To achieve TREC-7 adhoc experiments, we have reexported our distributed DSIR on a cluster of low-cost PC Pentium machines using PVM framework. Since the results of our first large-scale retrieval attempt in this TREC-7 are not quite successful, in terms of recall-precision measure, more work and experiments must be continued.

References

[1] J.P. Benzécri. Analyse Statistique des Données Linguistiques. In J.P. Benzécri & Collaborateur, editor, *Pratique de l'Analyse des Données*, volume 3 of *Linguistique & Lexicologie*. Dunod, Paris, 1981.

[2] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. PVM: Parallel Vir-

tual Machine–A Users' Guide and Tutorial for Networked Parallel Computing. The MIT Press, Cambridge, Massachusette, 1994.

[3] Z.S. Harris. Structure Distributionnelle. In M. Arrivé and J.C. Chevallier, editors, *Initiation à la linguistique: La grammaire*, Serie A3, pages 249–258. Klincksieck Linguistiques, Paris, 1975. French translation from original article of Z.S. Harris, "Distributional structure", in **Word**, number 2-3, 1954, by J. Throne and M. Arrivé.

[4] E. Ide. New Experiments in Relevance Feedback. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter 16, pages 337–354. Prentice-Hall, Englewood Cliffs, 1971.

[5] J.H. Lee. Combining Multiple Evidence from Different Properties of Weighting Schemes. In E.A. Fox, editor, *Proceedings of the* 18^{th} *Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, Seattle, Washington, USA, July 1995.

[6] M. Rajman and A. Bonnet. New Tools for Text Analysis: Corpora-based Linguistics. In *The First Conference of the Association for Global Strategic Information*, Bad Kreuznach, Germany, November 1992.

[7] M. Rajman and A. Rungsawang. How to find the nearest by evaluating only few? Clusterization techniques used to improve the efficiency of an Information Retrieval based on Distributional Semantics. In *The Fifth Conference of International Federation of Classification Societies (IFCS-96)*, volume 1, Kyoto, Japan, March 1996.

[8] A. Rungsawang. Distributional Semantic based Information Retrieval. PhD thesis, ENST-Paris, Department of Computer Science, Paris, France, 1997.

[9] A. Rungsawang and M. Rajman. A New Approach for Textual Information Retrieval. In E.A. Fox, P. Ingwersen, and R. Fidel, editors, *Proceedings of the* 18^{th} *Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, in Poster session, Seattle Washington, USA, July 1995.

[10] A. Rungsawang and M. Rajman. Textual Information Retrieval Based on the Concept of the Distributional Semantics. In *Proceedings of the* 3^{th} *International Conference on Statistical Analysis of Textual Data*, Rome, Italy, December 1995.

[11] G. Salton, editor. The SMART RETRIEVAL SYSTEM, Experiments in Automatic Document Processing. Prentice-Hall, Englewood Cliffs, New Jersey, 1971.

TREC-7 Experiments: Query Expansion Method Based on Word Contribution

Keiichiro Hoashi, Kazunori Matsumoto, Naomi Inoue, and Kazuo Hashimoto
{hoashi,matsu,inoue,kh}@kddlabs.co.jp

KDD R&D Laboratories, Inc.
2-1-15 Ohara Kamifukuoka
Saitama 356-8502 JAPAN

1 Introduction

This is KDD R&D Laboratories' first participation in TREC. In this participation, we focused on experiments on a novel method of query expansion.

The query expansion method described in this paper is based on a measure we call "word contribution". Word contribution is a measure which expresses the influence of a word to the similarity between the query and a document. From our data, we figured that words which have highly negative contribution can be considered as to being expressive of the characteristics of the data (query or document) in which they exist. We proposed extracting such words from documents highly similar to a query, and adding them to the original query to generate an expanded query. We made experiments to evaluate this method, and reported the results in this paper.

We submitted 3 official ad hoc runs (**KD70000**, **KD71010q**, **KD71010s**) to TREC-7. However, the data we used for these runs were generated by a buggy morphological analysis program, which we consider a serious cause for our bad results. Since the official submission, we have fixed these bugs, and reconstructured our data. The results described in this paper are based on these new data, and some experiments made after the TREC-7 conference.

2 Retrieval Method

2.1 Indexing

For indexing the topics and the documents, we ran a morphological analysis program on the data, and extracted nouns, proper nouns, and undefined words. A frequency table was made for each datum consisted of the extracted terms and frequency. The morphologial analysis program was the program in which bugs were discovered after the official submission.

In our experiments, we used the data from the TREC CD-ROMs 4 and 5, excluding the Congressional Reports. The total number of terms extracted from this data was 772,659.

2.2 Similarity Calculation

For similarity calculation, we applied a probabilistic model proposed by Iwayama et al [1]. This model is based on an idea called *Single random Variable with Multiple Values* (SVMV), and was proved effective in text categorization compared to other existing methods.

The formula for similarity calculation between documents d_1 and d_2 for SVMV is described in Figure 1, where:

$Sim(d_1, d_2)$:	Similarity of documents d_1 and d_2
$F_d(d, w)$:	Frequency of word w in document d
$N_d(d)$:	Number of words in document d
$F(w)$:	Frequency of word w in all documents
N:	Number of words in all documents

3 Query Expansion Based on Word Contribution

In this section, we will make an explanation of our proposed method of query expansion based on word contribution.

$$Sim(d_1, d_2) = M(d_1, d_2) - U(d_1) - U(d_2)$$

$$U(d) = \log\left(\sum_{w \in d} \frac{\left(\frac{F_d(d,w)}{N_d(d)}\right)^2}{\frac{F(w)}{N}}\right)$$

$$M(d_i, d_j) =$$

$$\sum_{d \in d_i, d_j} \log\left(\sum_{w \in d} \frac{\frac{F_d(d,w)}{N_d(d)} \cdot \frac{F_d(d_i,w)+F_d(d_j,w)}{N_d(d_i)+N_d(d_j)}}{\frac{F(w)}{N}}\right)$$

Figure 1: Similarity calculation formula for SVMV.

3.1 Definition of Word Contribution

Word contribution is a measure which expresses the influence of a word (or term) to the similarity between the query and a document. It is defined by the following formula:

$$Cont(w, q, d) = Sim(q, d) - Sim(q'(w), d'(w))$$

where $Cont(w, q, d)$ is the contribution of the word w in the similarity between query q and document d, $Sim(q, d)$ is the similarity between q and d, $q'(w)$ is query q excluding word w, and $d'(w)$ is document d excluding word w. In other words, the contribution of word w is the difference between the similarity of q and d, and the similarity of q and d when word w is assumed to be non-existent in both data. Therefore, there are words which have positive contribution, and words which have negative contribution. Words with positive contribution increases the similarity, and words with negative contribution decreases the similarity. An example of word contribution data calculated from TREC-6 data is shown in Table 1.

3.2 Hypothesis

Figure 2 illustrates the contribution of all words from the query and document used in the example shown in Table 1. The document used in this example is relevant to the query. The data is sorted in descending order according to the contribution of each word.

From Figure 2, it is apparent that there are only a small number of words with highly positive contribution, and a small number of words

Table 1: Words with 10 highest/lowest contribution in Topic 313 and FT932-6259

Word	Contribution
levitation	0.39429400
discussion	0.00030683
plan	0.00012887
year	-1.33E-06
government	-5.21E-06
system	-0.0000053
city	-5.77E-06
take	-6.54E-06
development	-6.92E-06
agreement	-7.91E-06
...	...
narrative	-0.0009595
JAPANESE	-0.0009871
JFK	-0.0044444
Guardia	-0.0046405
superconductivity	-0.0107779
Nakamoto	-0.0114731
Michiyo	-0.0137173
flywheel	-0.039495
Grumman	-0.0424242
motor_car	-0.1363256

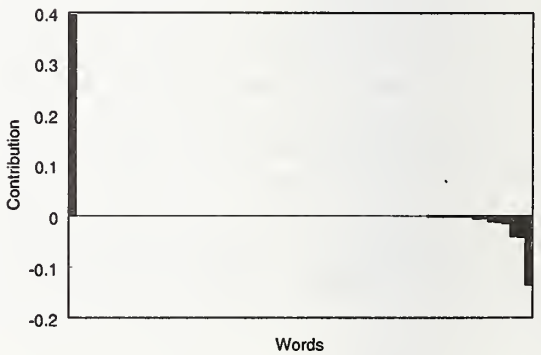


Figure 2: Word contribution between Topic 313 and FT932-6259

with highly negative contribution. On the contrary, most words have a contribution near zero, meaning most words do not have a significant influence on query-document similarity.

As obvious from the definition of word contribution, words with highly positive contribution are presumed to be words that co-occur in the query and document. Such words can be considered as informative words of document relevance to the query. On the contrary, words with highly negative contribution which do not occur in the original query can be considered as words which discriminate relevant documents from other non-relevant documents contained in the data collection.

Since the main objective of query expansion is to add words which are effective in distinguishing relevant documents from the data collection, we assumed that words with highly negative contribution are extremely suitable for expanding the original query. Moreover, we presumed that value of word contribution is a measure of the importance the word has for discrimination. Based on this presumption, the application of word contribution values as the weight of the extracted word for query expansion should also be effective.

3.3 Query Expansion Method

Based on our arguments in the previous section, we have developed the following query expansion method.

First, the word contribution of all words in the query and the set of documents from which the words for query expansion are extracted from are calculated. If there are Num documents which are included in the document set for query q, the relevant document set D_{qe} can be expressed as $D_{qe}(q) = \{d_1, \cdots, d_{Num}\}$. From each document d_i, N words with the lowest contribution are extracted.

Next, a score for each extracted word w is calculated by the following formula:

$$Score(w) = wgt \times \sum_{d \in D_{qe}(q)} Cont(w, q, d)$$

where wgt is a parameter with a negative value (since the contribution of extracted words are also negative). Finally, all extracted words and their scores are added to the original query. If any of the extracted words occur in the original query, that word is not added to the new query. Words with negative scores are also excluded from the expanded query.

4 Experiments

In this section, we will describe the experiments made to evaluate our query expansion method.

4.1 Preliminary Experiments

From the observation of word contribution data, we discovered that words which occur as a result of morphological analysis errors often have a highly negative contribution. Such "words" include terms with numbers, parantheses, or other punctuation marks. Examples of some of these data are: `[propose`, `0.1p`, `ID=JPRS-JST-003C-18A`, etc. These meaningless words must be deleted from the frequency tables of the documents in order to make an effective retrieval and query expansion.

Based on an empirical theory that these words do not occur frequently, words which occur less than a minimal number in all of the documents were excluded when calculating similarities. As a preliminary experiment, we set several minimal occurence thresholds, and made an evaluation of the text retrieval based on each threshold, by executing a search on TREC-6 data (Topics 301-350). The average precision, R-precision, and the number of retrieved relevant documents for each threshold are shown in Table 2.

Table 2: Retrieval results for baseline search on TREC-6 data

min	Avg Prec	R-Prec	Rel-ret
0	0.0388	0.0730	425
1	0.0418	0.0752	425
2	0.0423	0.0747	429
4	0.0394	0.0746	439
8	0.0439	0.0781	471
16	0.0442	0.0772	488
24	0.0452	0.0832	502
32	0.0435	0.0815	530
48	0.0449	0.0825	573
64	0.0459	0.0818	599

As apparent from the results in Table 2, there was not much difference between the results of these experiments. Therefore, considering the reasonability of threshold values, we decided to set

the minimum threshold to 16 and 32 for our following experiments. Therefore, the results for $min = 16, 32$ are used as the baseline for our evaluations on TREC-6 data. The baseline for TREC-7, i.e., TREC-7 retrievals when $min = 16$ and 32, are shown in Table 3.

Table 3: Retrieval results for baseline search on TREC-7 data

min	Avg Prec	R-Prec	Rel-ret
16	0.0442	0.0660	448
32	0.0435	0.0680	470

4.2 Query Expansion with Relevance Feedback

In order to examine the effectiveness of our query expansion method, we first made query expansion based on relevance feedback.

As described in previous sections, there are 4 parameters for our query expansion method: the minimum word occurence threshold (min), the number of documents for query expansion (Num), the number of words extracted from each document (N), and the weight applied to each extracted word and its contribution (wgt). For the description of the experiment results, we will use the following format:

rel.min.Num.N.wgt

For example, if $min = 16$, $Num = 10$, $N = 10$, and $wgt = -50$, the run for such conditions will be written as "rel.16.10.10.50".

The document set from which words for query expansion are extracted from is selected based on the results of the baseline search. Of all relevant documents for each query, the top Num ranked documents were extracted. If there were less than Num relevant documents for a query, then all relevant documents were included in the document set.

These experiments were made on TREC-6 and TREC-7 data. Results on TREC-6 data are described in Table 4.

As apparent from these results, we have achieved a significant improvement in both precision and recall compared to the baseline results. The influence of wgt was rather clear: the recall increases and the precision decreases as the absolute value of wgt increases.

Table 4: Retrieval results for query expansion with relevance feedback on TREC-6 data

Condition	Avg Prec	R-Prec	Rel-ret
rel.16.10.10.20	0.0877	0.1484	795
rel.16.10.10.50	0.0818	0.1414	954
rel.32.10.10.20	0.0932	0.1493	866
rel.32.10.10.50	0.0839	0.1422	992

For further analysis, the precision-recall curveline for the baseline and the query expansion results are presented in Figures 3 and 4.

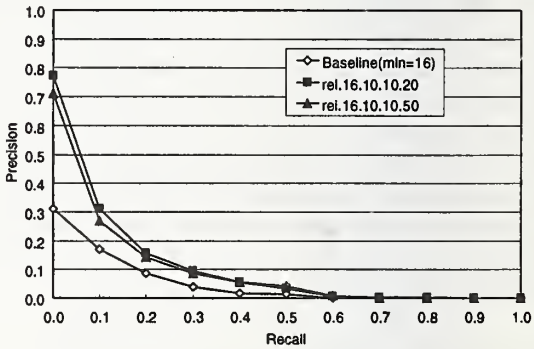


Figure 3: Precision-recall curveline for TREC-6 data ($min = 16$)

The results illustrated in these Figures also prove the effectiveness of our query expansion method.

Next, we will present the results of this experiment made on TREC-7 data. The average precision, R-precision, and number of retrieved relevant documents are shown in Table 5.

Table 5: Retrieval results for query expansion with relevance feedback on TREC-7 data

Condition	Avg Prec	R-Prec	Rel-ret
rel.16.10.10.20	0.0541	0.1134	652
rel.16.10.10.50	0.0551	0.1171	848
rel.32.10.10.20	0.0510	0.1154	740
rel.32.10.10.50	0.0513	0.1140	902

These results also show an improvement from the baseline retrieval, but the improvement is not as high as TREC-6 experiments. We will also present the precision-recall curveline for TREC-7 in Fig-

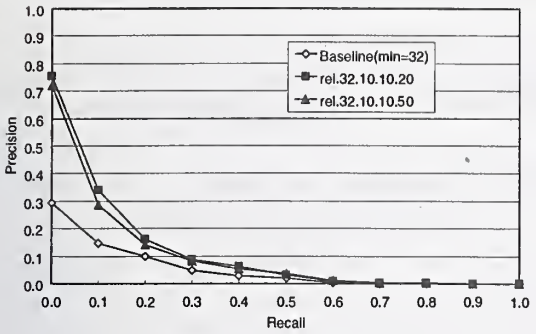


Figure 4: Precision-recall curveline for TREC-6 data ($min = 32$)

ures 5 and 6.

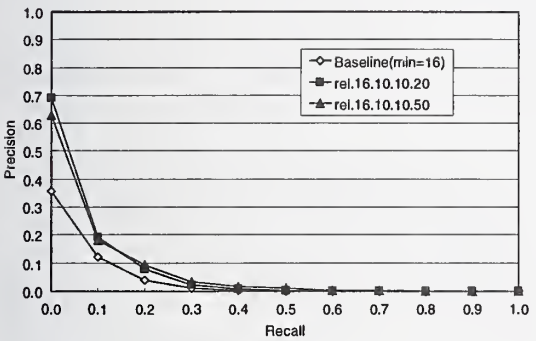


Figure 5: Precision-recall curveline for TREC-7 data ($min = 16$)

Observations from these Figures also show the lack of improvement compared to TREC-6 experiments. The most notable characteristic of TREC-7 results are the drastic descent of precision as the recall increases. Furthermore, the infuence of *wgt* is different from the results of TREC-6 experiments. As clear from Table 5, the increase of the absolute value of *wgt* results in the improvement of both the recall *and* precision.

However, overall results of the experiments described in this section proved the effectiveness of the query expansion method based on word contribution with relevance feedback.

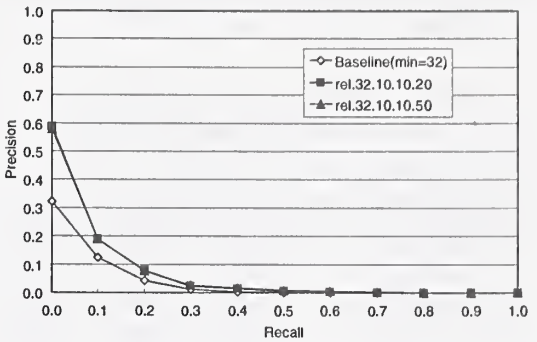


Figure 6: Precision-recall curveline for TREC-7 data ($min = 32$)

4.3 Query Expansion with Pseudo Feedback

Since the relevance of documents to a query is unknown in practical use of text retrieval systems, it is essential to develop an effective algorithm of retrieval without using relevance feedback information. Therefore, many systems using the idea of a pseudo feedback, i.e., selecting the top ranked documents from the pilot search as the document set used for query expansion, have been presented in recent years [2][3]. We have made experiments to apply pseudo feedback to our query expansion method. We will describe these experiments in this section.

The difference between the query expansion method described in the previous section based on relevance feedback and the query expansion method for this experiment is the extraction of the Num documents used for word extraction. In this experiment, we extracted the top Num documents of the baseline search as the set of documents used for query expansion, regardless of their actual relevance to the query.

We will use a format similar to the format used in the previous section for the expression of experiment conditions:

pse.$min.Num.N.wgt$

Similar to the previous section, we made experiments on both TREC-6 and TREC-7 data. Detailed experiments were made on TREC-6 data for analysis of the effects of various parameters, and a few experiments were made on TREC-7 data to

confirm results. In Table 6, the results for all experiments on TREC-6 data are shown.

Table 6: Retrieval results for query expansion with pseudo feedback on TREC-6 data

Condition	Avg Prec	R-Prec	Rel-ret
pse.16.10.10.20	0.0161	0.0387	416
pse.16.10.10.50	0.0137	0.0300	519
pse.16.10.10.100	0.0121	0.0232	518
pse.16.10.20.50	0.0140	0.0310	524
pse.32.5.10.20	0.0253	0.0543	445
pse.32.5.10.50	0.0142	0.0314	467
pse.32.10.5.20	0.0191	0.0191	468
pse.32.10.10.20	0.0198	0.0402	482
pse.32.10.10.50	0.0150	0.0280	547
pse.32.10.10.100	0.0131	0.0246	533
pse.32.10.20.50	0.0154	0.0280	558

As obvious from these results, our query expansion method did not improve the retrieval compared to the baseline search. The fact that the experiments with relatively high results used queries in which the expanded words had little influence, also back up the failure of our query expansion.

We will also present the results for the pseudo feedback experiment on TREC-7 data on Table 7. The conditions which had relatively good results from the experiments on TREC-6 data were selected for these experiments.

Table 7: Retrieval results for query expansion with pseudo feedback on TREC-7 data

Condition	Avg Prec	R-Prec	Rel-ret
pse.16.10.10.20	0.0125	0.0373	503
pse.16.10.10.50	0.0104	0.0285	594
pse.16.10.10.100	0.0085	0.0216	583
pse.32.10.10.20	0.0153	0.0387	536
pse.32.10.10.50	0.0101	0.0265	570
pse.32.10.10.100	0.0085	0.0216	583

As apparent from the data on Table 7, the results for experiments on TREC-7 were no better than those for the TREC-6 experiments. From these results, we presume that there are problems in our query expansion method using pseudo feedback.

5 Discussion

In this section, we will discuss the results of our evaluation experiments, and investigate the causes for the failure of our query expansion method.

5.1 Analysis of query expansion with relevance feedback

Although we have achieved significant improvement on our query expansion experiment using relevance feedback, consideration is necessary for improvement. As observed from the precision-recall curves illustrated in Figures 3-6, the precision of the retrieval descends rapidly as the recall increases. We examined the word contribution data used for query expansion in these experiments to investigate the cause of this phenomenon.

In Section 3, we explained that there are only a small number of words which have highly negative contribution. Further analysis of word contribution to the similarity between queries and relevant documents showed that, in many cases, there are 1 or 2 words per query-document set that have an extremely high absolute value of word contribution. An example of such data is illustrated in Figure 7.

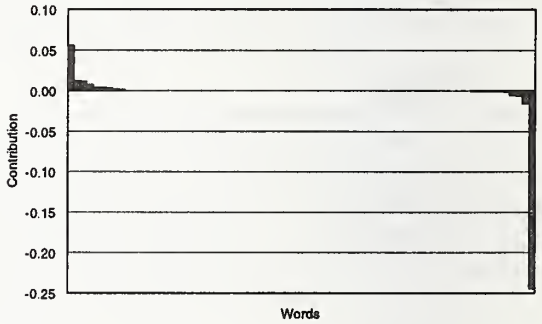


Figure 7: Word contribution between Topic 301 and FBIS3-10397

Since the values of *wgt* in our experiments were set so that the weight of extracted words would not be extremely higher than the frequencies of words in the original query, there is a strong possibility that the words other than the words with extremely high contribution did not apply sufficient influence on the expanded query.

For the investigation of this hypothesis, we ran query expansion experiments with $wgt = 1200$, so that the other words will have similar weights as

the original frequency table. The results are shown in Table 8.

Table 8: Retrieval results for query expansion with relevance feedback ($wgt = 1200$)

Condition	Avg Prec	R-Prec	Rel-ret
(*TREC-6*) rel.16.10.10.1200	0.0916	0.1454	1309
(*TREC-7*) rel.16.10.10.1200	0.0688	0.1390	1336

From the comparison of these results and the results presented in Section 4, it is clear that the drastic increasement of *wgt* has improved both the recall and precision of the retrieval. For comparison, we present the precision-recall curveline for the retrieval on TREC-7 data with *wgt* as 20 and 1200. This is illustrated in Figure 8.

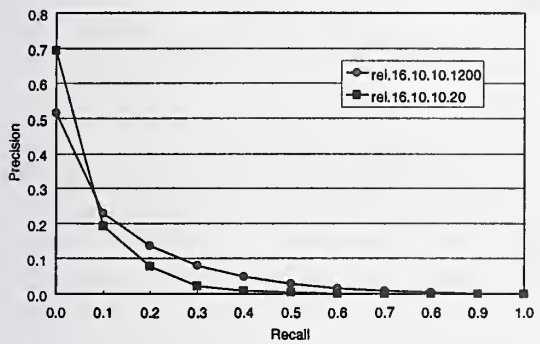


Figure 8: Precision-recall curveline for TREC-7 data ($wgt = 20, 1200$)

As observed from Figure 8, the precision at low recall of rel.16.10.10.1200 is not as good as rel.16.10.10.20. However, the precision of rel.16.10.10.1200 at higher recall is constantly higher than that of rel.16.10.10.20. From these results, we presume that the words with extremely high negative contribution are effective for retrieving documents at a low recall, while the other words extracted for query expansion are effective for retrieving a wide range of relevant documents. Therefore, it is necessary to merge these weights effectively in order to apply the characteristics of each set of words. A reduction or normalization of the extremely high contribution values, such as adapting a logarithm to the word contribution value, may be effective. We have yet to evaluate such methods.

5.2 Analysis on query expansion with pseudo feedback

One obvious cause of the failure of our query expansion method with pseudo feedback is the poor precision of the baseline search. Experiments were made on the TREC-6 data with the parameter Num set at 5, 10, and 20. The precision of the baseline retrievals at documents 5, 10, and 20 are shown in Table 9.

Table 9: Precision at 5,10,20 documents for baseline searches

	TREC-6		TREC-7	
	min=16	min=32	min=16	min=32
@ 5	0.1520	0.1560	0.1680	0.1480
@ 10	0.1480	0.1520	0.1520	0.1520
@ 20	0.1170	0.1160	0.1320	0.1200

As apparent from these results, 83%-88% of the documents used for query expansion were actually irrelevant to the query. This should have a negative effect on the results of query expansion.

In order to examine the effects of a poor baseline search, we simulated the TF*IDF based retrieval algorithm and the Rocchio feedback based query expansion method applied in the SMART system at TREC-7[3]. The Rocchio weights were calculated by the following formula:

$$\vec{Q_{new}} = \alpha \times \vec{Q_{org}} + \beta \times \frac{1}{R} \sum_{D \in Rel} \vec{D} - \gamma \times \frac{1}{N} \sum_{D \notin Rel} \vec{D}$$

where $\alpha = 3$, $\beta = 2$, $\gamma = 2$, and 20 new terms with the highest Rocchio weights for each query were added to the original query. These parameter values were set as the values presented in the SMART paper by AT&T on TREC-7. However, SMART also added 5 new phrases in this process. Since we do not have any indexing methods especially tuned for phrases, this function was not applied in our simulation.

The average precision, R-precision and number of retrieved relevant documents by the baseline search of SMART and the Rocchio feedback based query expansion are shown in Table 10, and the precision-recall curveline for these SMART retrievals are illustrated in Figure 9.

Table 10: Retrieval results for query expansion with pseudo feedback on TREC-7 data

Condition	Avg Prec	R-Prec	Rel-ret
Baseline	0.1433	0.1848	1887
Rocchio	0.1348	0.1691	1392

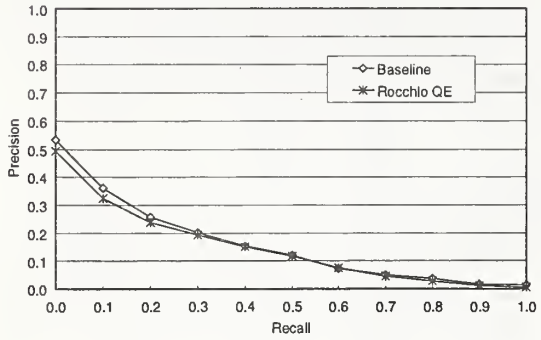


Figure 9: Precision-recall curveline for SMART algorithm

As apparent from Table 10, the result of the search by the SMART algorithm was good, but it is not as good as the results presented in the TREC-7 papers (AT&T's average precision was 0.2290.).

Furthermore, the results from Table 10 and Figure 9 show that the Rocchio feedback based query expansion could not exceed the baseline retrieval, which was similar to the case in our experiments. These results prove that a poor baseline retrieval has a negative impact on the results of query expansion, even when applying an established query expansion algorithm. Therefore, the lack of precision of the baseline search can be considered as a cause of our poor results.

However, further analysis on our data showed that queries with many relevant documents included in the document set for query expansion also had poor results after expanding the query. Table 11 is a list of topics of which the precision of the baseline retrieval at 10 documents were higher than or equal to 0.60, and the average precision of the baseline ($min = 16$) and query expansion (qe.16.10.10.50).

As obvious from these results, the high ratio of relevant documents included in the document set for query expansion did not improve the results of query expansion.

Table 11: Retrieval results for topics with high precision at 10 documents

Topic	Prec @ 10	Baseline	qe.16.10.10.50
302	0.90	0.2113	0.1286
314	0.70	0.2074	0.0468
353	0.60	0.0778	0.0823
357	0.80	0.1054	0.0361
368	0.60	0.0752	0.0508
398	1.00	0.1482	0.0095

The main idea behind our query expansion method may be an explanation of this result. As described in Section 3, the extraction of words with highly negative contribution was based on the hypothesis that such words are discriminant of the concerned document. If these words were extracted from documents which were not relevant to the original query, the expanded query will contain highly discriminant words of non-relevant documents. It is quite obvious that such query expansion will decrease the precision of retrieval.

Another cause may be the weighting problems which were pointed out in the previous analysis on relevance feedback experiments. In many cases, 1 or 2 extracted words have an extremely high weight after query expansion, as previously explained. This means that the discriminant words extracted from non-relevant documents will be extremely high weighted after query expansion. Therefore, the mere existence of non-relevant documents in the document set for query expansion can make a large negative influence on the final retrieval results.

However, it is difficult to make a strict failure analysis on the query expansion method if the indexing is erroneous. We suspect that this is the main cause of our poor results, since the text retrieval algorithm of SMART also did not achieve satisfying results with our frequency tables. Bugs on our dictionary are especially crucial with our current method, since words extracted for query expansion are observed to be words with relatively low term and document frequency, which may result from such bugs. Considering the fact that the contributions of the extracted words seem to be sensitive to the scarcity of the word, we believe that the improvement of our morphological analysis program is essential for strict evaluation.

6 Conclusion

In this paper, we have proposed a novel query expansion method based on word contribution, which is a measure of the influence of a word to query-document similarity. Through the analysis of word contribution on queries and relevant documents, we set a hypothesis that words with highly negative contribution are words which discriminate relevant documents from other documents in the data collection. Based on this hypothesis, we developed a query expansion method which adds such words and their weighted contribution to the original query.

First, we evaluated our query expansion by experiments using relevance feedback information. Results from these experiments proved the effectiveness of our proposed method. Second, we made evaluation experiments based on pseudo feedback. The results from these experiments were dissatisfying. Through the analysis of our results, we came to the conclusion that an improvement on the weighting of extracted words was necessary.

However, simulation of an established query expansion method on our data showed that an improvement on the indexing process, or, in other words, the dictionary used for our morphological analysis program was also necessary for the improvement of our results. We believe improvement of the morphological analysis program (or the application of a common-used program) is indispensable for future development.

One of our future studies will be the improvement of the weighting formula for extracted words. We consider it necessary to develop a new weighting method to cope with the words with extremely high contribution values. We also want to examine the word contribution of query-document similarity of non-relevant documents, which we have not made detailed analysis yet. Analysis on non-relevant documents should be helpful in our relevance feedback method.

Acknowledgments

The authors would like to appreciate the researchers in the Knowledge-Based Information Processing Lab of KDD R&D Laboratories for their fruitful opinions. We will also express gratitude to Shigeki Ohira of Waseda University and Marko Herzog of HTW Dresden for their tremendous efforts on our participation to TREC.

References

[1] M Iwayama, and T Tokunaga: "A Probabilistic Model for Text Categorization: Based on a Single Random Variable with Multiple Values", Proceedings of 4th Conference on Applied Natural Language Processing, pp.162-167, 1994.

[2] S Robertson, S Walker, S Jones, M Hancock-Beaulieu, and M Gatford, "Okapi at TREC-3", Overview of the Third Text REtrieval Conference, pp 109-125, 1994.

[3] A Singhal, J Choi, D Hindle, D Lewis, and F Pereira: "AT&T at TREC-7", The Seventh Text REtrieval Conference, 1998. (to be published)

Query Expansion and Classification of Retrieved Documents

C. de Loupy[1,2], P. Bellot[1], M. El-Bèze[1] and P.-F. Marteau[2]

(1) Laboratoire d'Informatique d'Avignon (LIA)
339 ch. des Meinajaries, BP 1228
F-84911 Avignon Cedex 9 (France)
{patrice.bellot,marc.elbeze}@lia.univ-avignon.fr

(2) Bertin & Cie
Z.I. des Gatines - B.P. 3
F-78373 Plaisir cedex
{deloupy,marteau}@boston.bertin.fr

Abstract: *This paper presents different methods tested by the University of Avignon and Bertin at the TREC-7 evaluation. A first section describes several methodologies used for query expansion: synonymy and stemming. Relevance feedback is applied both to the TIPSTER corpora and Internet documents. In a second section, we describe a classification algorithm based on hierarchical and clustering methods. This algorithm improves results given by any Information Retrieval system (that retrieves a list of documents from a query) and helps the users by automatically providing a structured document map from the set of retrieved documents. Lastly, we present the first results obtained with TREC-6 and TREC-7 corpora and queries by using this algorithm.*

keywords: ad-hoc information retrieval, automatic relevance feedback, synonymy, automatic classification, cluster-based and hierarchical methods.

1. Introduction

Our first goal in TREC-7 was to measure the performances of an Information Retrieval (I.R.) system, and the improvements brought by different methodologies. The basic tool of this system uses a Part Of Speech (POS) tagger and a lemmatizer[1]. Several modules have been tested: query enrichment techniques using stems or synonyms and two automatic relevance feedback methods: one using the TIPSTER corpus and the other using the World Wide Web.

The second goal was to evaluate a classification algorithm based on hierarchical and clustering methods. It is applied to the set of documents retrieved — and not to the collection as a whole — by using statistical techniques. Its purpose is to improve results given by any Information Retrieval system (that retrieves a list of documents from a query) and to help the users by automatically providing a structured document map from the set of retrieved documents.

This is our first participation in TREC evaluation. A lot of work had to be done and most of our efforts have been devoted to tune our system, so that not enough time was left for thorough testing. Nevertheless, among several achievements, it is possible to point out that some experiments have been very conclusive, as it will be reported in sections 2.5 and in 3.2. We have chosen to participate in the *ad hoc* task, using title or short queries.

2. Query expansion

It is well established that search procedures based only on words contained in a query cannot achieve high scores in Document Retrieval (DR) tasks. Indeed, one must deal with polysemy, synonymy. Furthermore, it is very important to identify the most relevant terms of the domain the query is referring to.

2.1. The basic methodology

The different words (obtained from the query or enrichment or relevance feedback procedures) are combined using fuzzy operators. The importance (information quantity) of a lemma λ depends on its frequency of occurrence [Maarek, 1991]:

$$I(\lambda) = -\log_2\big(P(\lambda)\big) \text{ with } P(\lambda) = \frac{K(\lambda)}{\sum_{\lambda'} K(\lambda')} \text{ where } K(\lambda) \text{ is}$$

the number of occurrences of λ in the corpus.

The quantity of information associated with a document D is then defined as:

$$I(D) = \sum_{\lambda \in D} \alpha(D, \lambda).I(\lambda)$$

where $\alpha(D, \lambda)$ is in fact a coefficient dependent on the size of the document and the number of occurrences of the lemma in this document.

[1] We use ECSTA, the Part of Speech tagger developed at the LIA[Spriet & El-Bèze, 1997]. Lemmatization is provided by lexical access.

The similarity between a document D and a query Q is:

$$S(D,Q) = \frac{I(D,Q)}{I(Q)} = \frac{\sum_{\lambda \in D \cap Q} \alpha(D,\lambda).I(\lambda)}{\sum_{\lambda \in Q} \alpha(Q,\lambda).I(\lambda)} \leq 1$$

The denominator is a constant and is used only for the normalization, it can be eliminated.

2.2. *Enrichment*

A first method to expand a query is to consider each of its lemmas independently, and to search for associated words: synonyms or words having the same stem. These expansions have been used to enrich *title* and *short* queries.

2.2.1. When and how to use associated words?

Query expansion with synonyms or stems must be handled carefully. Expanding very frequent lemmas could be dangerous. There is always a risk of a bad equivalence when expanding a lemma with its associated lemmas (that is to say synonyms or lemmas having the same stem). And the more frequent is the lemma, the more important are the consequences of an error in such equivalence. Therefore, words associated with a lemma are not taken into account if the latter appears in more than 5 % of the documents in the collection.

Moreover, polysemy is one of the most difficult problems in IR. Synonyms are related to the sense of the word and not to the word itself. If the sense of the word is not known, one will consider all the synonyms corresponding to its different senses as equivalent. Some experiments not reported here have shown that the precision would be very low in such a situation. Unfortunately, it is very difficult to determine the sense of a word in context. We are experimenting different methods to affect sense to words in context [Loupy et al., 1998b – Loupy et al., 1998c]. But since the different methods are not yet validated through reliable assessment, we prefer not to use them in an IR task. This validation is in progress within the SENSEVAL project [Kilgarriff, 1998]. Since a Word Sense Disambiguation (WSD) tool is not yet included in the IR tool, the system verifies the number of possible senses for one term, before deciding to expand it. If the word has no more than two senses, it is expanded with its synonyms.

At last, even if two lemmas can be highly related by the way of synonymy, a document containing the words of the query should be considered more relevant than documents containing their synonyms. In order to give more importance to the lemmas of the query, a coefficient δ ($\delta < 1$) is applied to the information quantity of the associated lemmas.

2.2.2. Stemming

Stemming can be very useful to expand queries, particularly to alleviate lacks in the lexicon. But stemming is not based on a high-level linguistic knowledge and, in many cases, the confusion involved by the use of stems leads to spurious effects. A guesser could solve some of the problems of Out Of Vocabulary (OOV) words. But there was not enough time to extend to English the one developed at the University of Avignon [Spriet et al., 1996] for French. Moreover, stemming can find out very close (semantically) words. In the experiments described in this paper, the Porter's stemmer[2] has been used [Porter, 1980].

For instance, consider the request 317 (TREC 6): "*Unsolicited Faxes*". The tagger finds that *unsolicited* is an adjective and *faxes* is an inflected form of the noun *fax*. The stemmer links the noun *fax* with the verb *to fax* and the OOV word *megafax* (!).

2.2.3. Synonymy

Many experiments were done with WordNet [Miller et al., 1993] to cope with the synonymy phenomenon (see for instance [Voorhees, 1993]). We have chosen to use this thesaurus. We have also done several experiments taking advantage from the hyponymy relation given in WordNet. But if this led to a slight improvement in the average precision/recall curve for all the queries in TREC-6, it was a real disaster for some requests. The reason is that the depth of the semantic inheritance tree can be too important. Therefore, only synonyms have been used.

The use of synonymy enrichment, for the request 317, linked *unsolicited* with the adjective *undesired* and *fax* with the noun *facsimile*.

2.3. *Lexical affinities*

A lexical affinity (LA) [Maarek, 1991] represents the affinity between *n* lemmas in a given corpus, that is to say, if they often appear in a near context. The system considers a context of 5 lemmas (content words) before and after a given lemma. According to [Martin et al., 1983 - cited in Maarek, 1991], 98 % of the lexical relations relate words contained in a 11 lemmas window. The LAs considered by the system relate 2 or 3 lemmas. The goal is to take into account the adjacency between the words of the request within the document.

First, the query is analyzed in order to extract the lexical affinities it contains. Let A a LA of the query *Q*. We can define a quantity of information for A and a similarity

[2] An implementation in C of the Porter's algorithm is available at http://www.cs.jhu.edu/~weiss/ir.html

between a document and a query using only LAs as done in 2.1 for lemmas. The similarity using single lemmas (S_{lem}) and the one using LAs (S_{LA}) are combined:

$S(D,Q) = \beta.S_{lem}(D,Q) + (1-\beta).S_{LA}(D,Q)$ where β is a coefficient ($\beta \leq 1$). We have used the empirical value $\beta = 0.7$.

In fact, when the query is expanded with synonyms or stems, an associated word is considered as the lemma itself (equivalence) for the construction of the LAs.

2.4. Automatic relevance feedback

Relevance feedback is a very classical way to automatically expand queries. Several methods are possible. The implemented methods search for documents containing all the terms of the query. Consequently, it is not possible to apply them on the so-called 'short' queries because they are too long. And, since we did not have time to develop a specific method, relevance feedback was used only with 'title'.

2.4.1. Relevance feedback using TIPSTER corpus

To get relevant terms, the texts containing all the lemmas of the query are analyzed in order to get the words appearing in the near context (a 10-word window) of these lemmas within the texts of the TIPSTER corpus. The list of all these words is arranged according to the number of times they were seen. Finally, in order to keep the ones that do not occur a lot of times in the corpus, but often in the context of the query terms, empirical thresholds have been used.

For example the enrichment processing from the documents containing *fax* and *unsolicited* returns the following lemmas: *advertisement, mail, machine, junk, ban, firm, ad,* which do not appear in the query.

2.4.2. Relevance feedback using the World Wide Web

As mentioned in the previous paragraph, it is useful to search for relevant terms in the context of query terms. We only tried such a method on the TIPSTER corpus, which is the textual database to search in for relevant texts. But the size of this corpus (2 Gb) is not very large compared to the amount of texts available on the Internet. Then, the second method used to enrich the query consists in searching for relevant lemmas in texts found in the World Wide Web. But, if the advantage of the WWW is the great amount of available information, this wealth also constitutes a serious drawback. We cannot use the same method to retrieve texts on the Web than the one applied on the TIPSTER corpus.

We have chosen to consider the query not as a set of words or lemmas, but as an indivisible entity. Pattern-matching retrieval has been used. Therefore the texts retrieved from the web with a TREC query contained 'exactly' this query[3]. The number of retrieved texts is not too large and the probability of their relevance is relatively high. For instance, considering the request 301, the retrieved texts contain exactly the **string**: "international organized crime". Of course, in this way, a lot (and even most) of pertinent documents are left aside, but what we need is to avoid unrelevant ones.

The method used to get the lemmas for the enrichment from the document is the same than in 2.4.1. But in this case, the validation process is not based on the number of occurrences. It is important to verify that these new lemmas, retrieved from every kind of sources, have real affinities with the words of the query. Hence, the context of these terms in the texts retrieved from the WWW is analyzed and a word is kept if one of the lemmas occurring in the query appears frequently enough in its context.

If we consider request "*unsolicited fax*", the following lemmas are taken into account to enrich the query: *sender, calling, bell, gt, e-mail, illegal, voice, spam, phone, machine, facsimile, email, check, anyone.*

2.5. Combining methods

In order to improve results, the different methods are combined. Table 1 hereafter shows the performances of the different modules. The first column indicates the method, the second one the number of relevant document retrieved (for all the queries), the third is the first point of the curve recall/precision, the fourth is the average precision (A-prec) and the last the R-precision (R-Prec).

	Rel.	at 0.10	A-Prec	R-Prec
Basic	1884	0.4029	0.1739	0.2256
Stems	2033	0.3925	0.1951	0.2381
Synonyms	1876	0.4060	0.1742	0.2269
Syn+Stems	2034	0.4140	0.2115	0.2535
L.As.	2006	0.4208	0.2165	0.2585
Rel. Feed. WWW	1920	0.3883	0.1834	0.2309
Rel. Feed. TIPSTER	1933	0.4105	0.1845	0.2308
Rel. Feed. (2 modules)	1937	0.4034	0.1880	0.2358
all modules (except LAs)	2078	0.4220	0.2188	0.2593

Table 1: Scores of the different modules

These figures show that, if each module can, more or less, improves some scores, the combination of several methods

[3] In fact, some characters are not submitted (like parenthesis) and others are replaced by 'and' or 'or' (like '/').

is the best way to increase both recall and precision. For example, on the one hand, the use of synonyms slightly improves precision (0.4060 at 0.10), but does not lead to a gain in recall (1876 relevant documents retrieved). On the other hand, stemming decreases precision (0.3925 at 0.10) but highly improves recall (2033 relevant documents retrieved). The combination of stems and synonyms clearly improves both precision (0.4140 at 0.10) and recall (2034 documents retrieved).

The following figure gives the recall/precision curves for the basic method (B.), the enrichment by associated words (L.) corresponding with Syn+Stem in Table 1, the relevance feedback method with both modules (R.F.) and all the modules together.

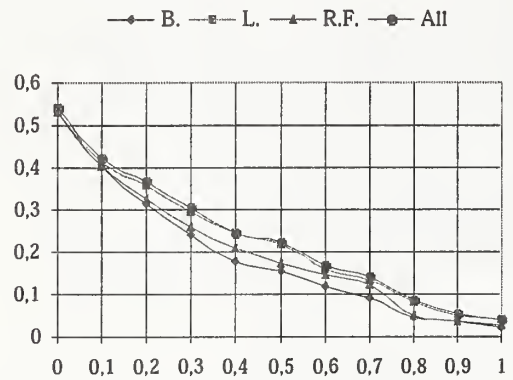


Figure 1: Recall/Precision curves

One can see that the most important improvement comes from synonymy and stemming enrichment procedures. But, although relevance feedback does not lead to a great improvement, the methodology used to get the relevant lemmas could be improved: we only take into account lemmas in the context of the query. Many other techniques could be used.

3. Clustering of retrieved documents

Classification algorithms have been used already in IR to improve the efficiency and effectiveness of retrieval by classifying the documents of a target corpus [VanRijsbergen,1979- Salton,1989- Rasmussen,1992]. A classical information retrieval system retrieves and ranks documents extracted from a corpus according to the computation of distances between the texts and a user query. The answer list is often so long that users cannot examine all the documents retrieved whereas some relevant items are badly ranked and thus never retrieved. In order to solve this problem, we have chosen to automatically cluster retrieved documents according to their topics. Indeed, one assumes that relevant documents are close just like in a relevance feedback scheme one thinks that a relevant document will help to retrieve the other ones (the "Cluster Hypothesis" [Van Rijsbergen,1979]). We present an algorithm combining hierarchical classification and cluster-based (K-means like) methods. They are applied to the set of documents retrieved — and not to the collection as a whole — by using statistical techniques. Hence, the classification is sensitive to the content of the queries. One can summarize this process of information retrieval as shown in Figure 2.

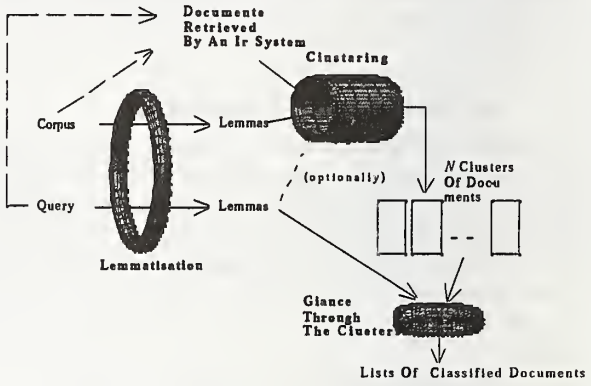


Figure 2 - Classification of documents retrieved

3.1. A Clustering algorithm with hierarchical and cluster-based aspects

An important criteria of an IR system is the time a user has to wait for an answer. Thus, we have chosen to use a K-means like method [Diday, 1982] to cluster the retrieved documents, in particular because its time and storage requirements are much lower than those required by hierarchical algorithms.

This algorithm aims to cluster items as follows:

- Find an initial partition (see 3.1.1)
- Do:
  1. Compute representatives of each existing cluster[4];
  2. assign each document to the most similar cluster.

  while a clustering quality criterion increases or until there is little or no change in cluster membership.

Since the cluster's representatives are computed only at the beginning of an iteration, the cluster's memberships are order independent.

Optionally, a document could be placed in a cluster only if the distance between the document and the cluster representatives does not exceed a given threshold[5].

[4] The number of clusters must be initially chosen.

Since this process may be seen as post-processing, it can be used with any IR-system which returns a list of documents to a user query. This method may only use the set of retrieved documents. Optionally, the knowledge of cumulative frequencies of each lemma in the corpus (and not only in the retrieved documents) improves the quality of the classification. Lastly, we can use queries in order to rank clusters.

3.1.1. Class representatives

It is required to compute centroids of each cluster so that the distances between a cluster and a document or between a cluster and a query can be calculated to allocate documents to their most similar cluster and to be able to rank clusters.

We choose to represent a cluster by the k documents[6] which are the closest to the geometric centre: we compute, for each document, the sum of distances from it to other texts belonging to the same cluster and choose the k documents corresponding to the k smallest distances as representatives.

3.1.2. Initial partition

Since the result of this cluster-based method depends on the initial set, the first problem one faces is to decide how to obtain a valid initial partition. A randomly attribution of documents in clusters is a simple idea but not a good one because clusters are consequently close, their representations similar and the number of iterations of the algorithm before convergence is too large.

We have made several experiments by using different methods [Bellot & El-Bèze,1998b] which will not be reported here. During our first tests on some TREC-5 corpora and queries [Bellot & El-Bèze, 1998a] we have used, among others, a partial hierarchical method to obtain the initial partition and thus strongly improved the quality of the final classification.

In order to obtain the initial classification, we do:

(a) for each couple of documents i and j such that $d(i,j) < threshold$[7]:
   - if i and j are not yet in a class, then create a new one;
   - if i and/or j are already assigned, merge all the documents of the class containing i (resp. j) with those containing j (resp. i);

(b) after this first step, the number of classes created may be greater than the preset number of clusters. So, while the number of classes is greater than the predefined one:
   - compute class representatives;
   - compute distance between every pair of classes (triangular matrix);
   - merge the two closest classes.

3.1.3. Distances

In order to be able to measure the quality of the partition and to assure convergence of the classification process, we must have a *real* distance (satisfying the triangular inequality). That is the case of the so-called *MinMax* distance described hereafter.

Let R and D be two documents, u a lemma and its syntactical tag, $N(u)$ the number of documents containing u in the corpus as a whole and not only in retrieved documents.

The *information quantity* of a lemma is based on its occurrences within the corpus:

$$I(u) = -\log_2 \frac{N(u)}{\sum_{u' \in \text{Corpus}} N(u')}$$

The information quantity of a document is the sum of information quantities of its lemmas:

$$I(D) = \sum_{u \in D} I(u)$$

We assume that the greater the information quantity of the intersection of lemmas of two documents is, the closer they are.

Let the distance between two documents R and D be:

$$d(R,D) = 1 - \frac{I(R \cap D)}{\text{Max}\left(I(R), I(D)\right)}$$

Let k be the number of representatives of a cluster C. Let $D_i \left(1 \le i \le k\right)$ be a representative[8] of C.

Let the distance between a document and a cluster be:

$$d(R,C) = \min_{1 \le i \le k} \left(d(R, D_i)\right)$$

[5] Tried values range from the average 'document to document' distance to 1 (to exclude documents which have not a common word with the cluster representatives).

[6] So far we have empirically selected $k = 3$.

[7] We choose the threshold value so as the quantity of documents assigned at the end of step '(a).' is greater than half the total number of documents.

[8] Let us recall that a representative of a cluster is a subset of documents.

In order to provide to the user a ranked list of documents from the partition or an arranged view of the clusters, we have to be able to compute distances between a cluster and a query ("what is the cluster which is the closest to the query ?"). This is accomplished using the above distance (R a query).

We have also to estimate what are the documents which are the closest to the query in each cluster so that one can rank them. This can be achieved using the similarity or distance values given by the IR system.

3.2. *Experiments with TREC corpora and queries*

We have assumed that a good classification allows to cluster documents according to their themes. If a query has only one theme, we should consider the best ranked cluster which should contain the relevant documents. But what to do if a query covers several topics ? We could look at the best ranked documents of each cluster *i.e.* at the documents for each theme which are the closest to the query or, merely at each cluster according to its rank. Lastly, we can present each cluster to the user so as he/she chooses those containing the largest number of relevant documents. In order to evaluate the classification process without taking into account the ranking process of clusters, we use the list of relevant documents (the *qrels* file) supplied by the TREC organization and select the best clusters for each query according to the number of relevant documents they contain (see [Hearst & Pedersen, 1996] and [Silverstein & Pedersen, 1997]).

We have chosen to assign a document to a cluster only if the distance between them is lower than a empirical threshold and to group together all remaining documents in a new cluster at the end of the process. Moreover, the documents in each cluster are ranked according to the similarity values between them and the queries.

3.2.1. TREC-6

By using the TREC-6 corpora and queries (from 301 to 350) and by classifying the 1000 best ranked documents retrieved by the "SynStem + LAs" method (cf. 2.5) for each query, we have been able to obtain some better results with classification rather than without it. We should be able to get better ones by choosing different parameters and by modifying the similarities used.

The graph and the table printed below show the recall-precision curves and results obtained:

(a) without classification ("SynStem + LAs" method alone);

(b) with 2 clusters ranked for each query according to the similarity d defined here;

(c) with 2 clusters ranked for each query according to the number of relevant documents they contain;

(d) with 8 clusters ranked for each query according to the number of relevant documents they contain.

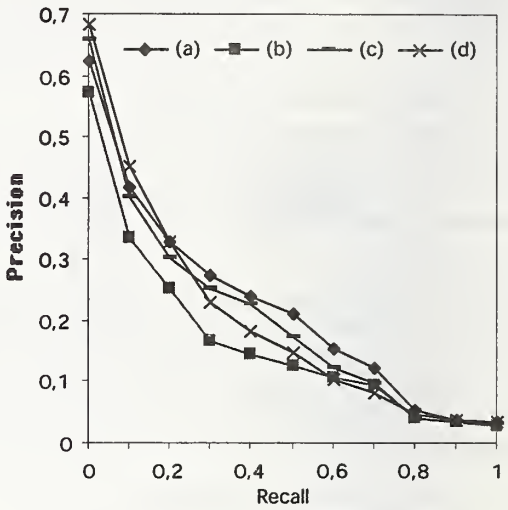


Figure 3 - Results with TREC-6 queries

	Precision at recall 0.00	Precision at recall 0.10	Precision at 5 docs	Precision at 10 docs
(a)	0,624	0,412	0,412	0,328
(b)	0,574	0,336	0,348	0,296
(c)	0,659	0,402	0,42	0,344
(d)	0,684	0,452	0,432	0,368

Table 2: Results with TREC-6 queries and corpora

By making a choice of 8 clusters and by ranking them according to the number of relevant documents they contain, the relative increase of precision for 0.00 and 0.10 recall values is 9.6% and 9.7% —curve (d)—. Precision is 4.8% better at 5 documents and 12.2% at 10 documents. By choosing 2 clusters and by ranking them according to the number of relevant documents they contain —curve (c)—, at 0.00 recall value, the relative increase is equal to 5.6% and at 0.1 recall value, precision is lower than those obtained without classification (-2.4%). However, precision is better: +2% at 5 documents and +4.9% at 10 documents.

Lastly, by choosing 2 clusters and by ranking them according to the similarity *d* defined above —curve (b)—, precision is always lower than without classification[9].

3.2.2. TREC-7

By using the TREC-7 corpora and queries (from 351 to 400) and by classifying the 1000 best ranked documents retrieved for each query, we have obtained better results with classification rather than without it.

Table 3 shows the results obtained:

(a) without classification;

(b) with 2 clusters ranked for each query according to the similarity *d* defined here;

(c) with 2 clusters ranked for each query according to the number of relevant documents they contain.

By choosing 2 clusters and by ranking them according to the number of relevant documents they contain —curve (c)—, the relative increase of precision at 5 documents is equal to 10.5% and at 10 documents, is equal to 6%. Lastly, by choosing 2 clusters and by ranking them according to the similarity *d* defined above —curve (b)—, precision is better with classification rather than without it only at 10 documents (+3%).

	Precision at recall 0.00	Precision at recall 0.10	Precision at 5 docs	Precision at 10 docs
(a)	0.57	0.37	0.38	0.34
(b)	0.58	0.38	0.38	0.35
(c)	0.63	0.35	0.42	0.36

Table 3: Results with TREC-7 queries and corpora

Moreover, for each query, the average ratio of retrieved relevant documents which are in the best class equals 76% (these classes contain 61% of the retrieved documents). That confirms the classification helps to regroup relevant documents.

4. Conclusion

It is clear that we have not reached the roots of all the methodologies described in this paper. Nevertheless, some interesting results have emerged from the different experiments. WordNet and stems can be used effectively together. Using Internet for relevance feedback seems full of promise, since the method we use to get relevant documents and lemmas is very simple. Concerning the classification process, we have obtained some improvements. Choosing different parameters and modifying the similarities used should lead to better results. We have shown that this classification method helps to regroup relevant documents. It increases the effectiveness of retrieval by providing to users a structure of texts and by allowing them a faster examination through the list of retrieved documents. Our participation to the AUPELF-UREF *Amaryllis-2* information retrieval project for the French language will permit to present some new results obtained with our tools.

[9] For recall values from 0.4 to 0.8, one can see that, whatever the classification evaluated, the best results are those obtained without classification. In order to resolve this problem we will try to provide a ranked list of documents which is not the entire succession of clusters contents but the first ones of the first cluster followed by the first ones of the second cluster and so on...

5. References

[Bellot & El-Bèze, 1998a] P. Bellot, M. El-Bèze, "*Classification Automatique et Recherche d'Informa-tion*", Technical Report, IR-06-1998, Laboratoire d'Informatique d'Avignon, 1998.

[Bellot & El-Bèze, 1998b] P. Bellot, M. El-Bèze, "*A Clustering Method for Information Retrieval and Text Segmentation*", to be submitted.

[Diday, 1982] E. Diday, J. Lemaire, J. Pouget, F. Testu, "*Eléments d'Analyse des Données*", Dunod Informatique,1982.

[Frakes, 1992] William B. Frakes, Ricardo Baeza-Yates (Editors), "*Information Retrieval, Data Structures & Algorithms*", Prentice-Hall Inc., 1992, ISBN-0-13-463837-9.

[Hearst & Pedersen, 1996] Marti A. Hearst, Jan O. Pedersen, "*Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results*", Proceedings of ACM-SIGIR 96, pp.76-84, 1996.

[Kilgarriff, 1998]: A. Kilgarriff; "*SENSEVAL: An Exercise in Evaluating Word Sense Disambiguation Programs*"; in Proceedings of the First International Conference on Language Resources & Evaluation; pp. 1255-1258; Granada, Spain; 28-30 May 1998.

[Loupy *et al*, 1998a] C. de Loupy, P.-F. Marteau & M. El-Bèze; "*Navigating in Unstructured Textual Knowledge Bases*"; in Proceedings of Nîmes'98 - La Lettre de l'IA; pp. 82-85; May 1998.

[Loupy *et al*, 1998b] C. de Loupy, M. El-Bèze & P.-F. Marteau; "*Word Sense Disambiguation using HMM Tagger*"; in Proceedings of the First International Conference on Language Resources & Evaluation; pp. 1255-1258; Granada, Spain; 28-30 May 1998.

[Loupy et al., 1998c] C. de Loupy, M. El-Bèze & P.-F. Marteau; "*WSD based on three short context methods*"; in Proceedings of SENSEVAL Workshop; Herstmonceux Castle, England; 2-4 September 1998.

[Maarek, 1991] Y. S. Maarek; "*Software Library Construction From an IR Perspective*"; SIGIR forum, Fall 1991, 25:2; pp. pp. 8-18; 1991.

[Marteau *et al*,1998] P.-F. Marteau, C. de Loupy, P. Bellot & M. El-Bèze, "Le Traitement Automatique du Langage Naturel Appliqué à l'Intelligence Economique: vers une Architecture "Push-Pull" d'Accès à l'Information", Système & Sécurité, submitted.

[Martin *et al*, 1983]: W.J.R. Martin, B.P.F. Al & P.G.G van Sterkenburg; "*On the processing of a text corpus: from textual data to lexicographic information*"; in R.R.K. Hartmann, editor, *Lexicography: Principles and Practice*; Applied Language Studies Series, Academic Press; London, 1993.

[Miller *et al*, 1993] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross & K. Miller; *Introduction to WordNet: An on-line lexical database*; http://www.cosgi.princeton.edu/~wn; August 1993.

[Porter, 1980]: M. F. Porter; *An algorithm for Suffix Stripping*; Program 14 (3); pp. 130-137; July 1980.

[Rasmussen,1992] Edie Rasmussen, "*Clustering Algorithms*", in [Frakes,1992], pp.419-442, 1992.

[Rissanen *et al.*,1992] Jorma Rissanen, Eric Sven Ristad, "*Unsupervised Classification with Stochastic Complexity*", Proceedings of the US/Japan Conf. on the Frontieres of Statistical Modeling, 1992.

[Salton,1989] G.Salton, "*Automatic Text Processing*", Reading, Mass. Addison-Wesley, 1989.

[Salton *et al*,1994] Gerard Salton, James Allan, Chris Buckley, "*Automatic Structuring and Retrieval of Large Text Files*", Communications of the ACM, Vol.37, No.2, 1994.

[Silverstein & Pedersen, 1997] C. Silverstein, Jan O. Pedersen, "*Almost-Constant-Time Clustering of Arbitrary Corpus Subsets*", Proceedings of ACM-SIGIR 97, p.60-66, 1997.

[Spriet *et al.*, 1996] T. Spriet, F. Béchet, M. El-Bèze, C. de Loupy & Liliane Khouri, *Traitement automatique des mots inconnus*; in Proceedings of TALN'96; 1996; Marseille; pp 170-179.

[Spriet & El-Bèze, 1997]: T. Spriet & M. El-Bèze; *Introduction of Rules into a Stochastic Approach for Language Modelling*; in Computational Models for Speech Pattern Processing, NATO ASI Series F, editor K.M. Ponting; 1997.

[VanRijsbergen,1979] C. J. Van Rijsbergen, "*Information Retrieval*", Buttherwords, London, 1979.

[Voorhees, 1993]: Ellen M. Voorhees; *Using WordNet to Disambiguate Word Sense for Text Retrieval*; ACM-SIGIR'93; pp. pp. 171-180; Pittsburg, PA, USA; June 1993.

Experiments in Query Processing at LEXIS-NEXIS for TREC-7

Ashwin G. Rao, Timothy Humphrey, Afsar Parhizgar, Christi Wilson, Daniel Pliske
Applied Research
LEXIS-NEXIS
(ashwin.g.rao,timothy.humphrey,afsar.parhizgar,christi.wilson,daniel.pliske)@LEXIS-NEXIS.com

Introduction

The purpose of this report is to provide an overview of LEXIS-NEXIS' entries to the TREC-7 competition. The report will describe the experiments we conducted, the results we obtained, and our future research directions. The report is divided into three sections. The first section describes the experimental setup and gives a brief account of some of the research activities that led to the TREC-7 entries. The second section explains how the techniques developed during our research culminated into the three entries that were submitted. Our experiences with these new techniques gave us insight into new research directions for improving query processing. In the third section, we conclude by sharing these ideas with the reader.

1 - TREC-7 Research at LEXIS-NEXIS

In the past [1] [2] [3], we concentrated on the evaluation of various query enhancement techniques to improve the quality of the final retrieval. This year, we decided to focus our attention on one specific technique. While preparing for our TREC submission last year, we came to the conclusion that retaining the focus of the original queries was critical during query expansion, and we could adopt co-occurrence analysis techniques to help us meet this need. Therefore, this year, we decided to spend a major portion of our time fine-tuning our co-occurrence metrics. Adding related terms to the query normally helps retrieve more documents, at the expense of precision (i.e. reducing the number of relevant documents at the top of the ranked list.) We believed that if we could improve our co-occurrence analysis process, we could add as many terms as possible to the queries and let the co-occurrence analysis process eliminate superfluous terms to bring documents that are more relevant into the ranked list. Our goal was to improve the precision of the final retrieval by 10% over last year's best official result. The best official result for TREC-6 was from the City University of London. The best unofficial result was from UMASS. Table 1 contains the precision values from the two universities and our final training results.

TITLE+DESC (TRAINING)				*Precision*		
Entry	***Avg. Precision***	***Exact Precision***	***Rel_Ret***	***At 5 docs***	***At 20 docs***	***At .20***
City University	0.2327	0.2595	2422	0.4360	0.3320	0.3892
UMASS (unofficial)	0.2730	0.3021	N/A	N/A	0.4200	N/A
LN (training)	0.2749	0.3080	2685	0.5080	0.3850	0.4771
(Improvement over City Univ.)	+18.1%	+18.7%	+10.9%	+16.5%	+16.6%	+22.6%
(Improvement over UMASS)	+0.7%	+1.95%	N/A	N/A	-8.33	N/A

Table 1. Comparison of results from the best performers last year vs. results after training.

In order to utilize our computing resources effectively, we re-engineered our processes so that we could distribute them and run them in parallel over a large set of hardware resources (around 200 Solaris workstations and servers).
Our initial plan was to participate in both Ad hoc and Filtering tracks, but we later decided to limit our participation to the Ad hoc task due to time constraints. We have submitted three entries, LNaTitleDesc7, LNaTitle7, and LNmanual7. As the names denote, the LNaTitleDesc7 entry used terms from both the title and the description fields of the TREC-7 topics to automatically retrieve documents. The LNaTitle7 entry used just the terms from the title field of TREC-7 topics

for automatic retrieval. Our manual entry, LNmanual7, used any information available to the human analysts for manual retrieval. All three entries will be described in detail in this report.

In any information retrieval system, text data goes through various stages before it can be used by the search engine. We used TREC-7 topics to make queries that are then used by the search engine to retrieve relevant documents from the TREC-7 corpus. Apart from the original topic terms, the queries can contain additional terms from external sources like dictionaries, thesauri and the LEXIS-NEXIS' REL[1] feature. The inverted files, corpus statistics, and queries become inputs into the search engine that creates a ranked list of documents. During the training process, we score the performance of our retrieval system by comparing the ranked list of documents output by the search engine, with a list of relevant documents provided by NIST. We use this scoring process to learn how to improve our retrieval processes.

This year we used the TREC-6 topics to train the system. We opted to use the unofficial results from UMASS as the benchmark for our training.

1.1 - Search Engine

To choose a search engine for this year's TREC competition, we revisited our best implementation of the well-known algorithms that were used at previous TREC conferences. We chose to implement Ocelot (based on City University's BM25 [4]), Inquery (based on UMASS' Inquery [6] [8]), and Panther (based on Cornell's Lnu.ltu [7]). Please refer to the original papers for more information about the algorithms.

During our initial training runs using the title and description fields of the TREC-6 queries, we highlighted nouns by incrementing their frequencies and we identified and added phrases to queries. We used these new queries to help us pick one or two of the best implementations of the algorithms described above. Ocelot turned out to be a clear winner as can be seen from Table 2. We therefore adopted it for the remainder of our experiments.

TITLE + DESC (TRAINING)				*Precision*	
Method	*Rel_Ret*	*Avg. Precision*	*Exact Precision*	*At 0.20*	*At 20 docs*
Ocelot	2542	.2560	.2885	.4747	.3640
Panther	2330	.2125	.2377	.3961	.2920
Inquery	2431	.2555	.2835	.3901	.3340

Table 2. Comparison of results of the three search algorithms.

We modified the Ocelot algorithm by incorporating query term coverage and query term dependency techniques. Both techniques are related to query processing and they will be described in the next section.

1.2 - Query Creation and Enhancement

This section will briefly describe various steps taken to enhance the queries. The first sub-section describes how we pre-processed our topics. Section 1.2.2 describes our attempts in adding more relevant terms to the topics to effectively improve the recall while keeping the precision as high as possible.

Section 1.2.3 describes two techniques that we have found to be effective for very short queries (consisting of 2-3 terms). The last sub-section explains the relevance feedback approach.

1.2.1 - Query Pre-processing

In our experiments, we have found that title terms carry much more information than the terms in the description, validating the observations by Voorhees [12]. In order to ensure the dominance of the title terms, we multiplied the within-query term frequencies by a factor. We arrived at the multiplier after some ad hoc experimentation.

[1] The LEXIS-NEXIS' REL (RELated concepts) feature of the LEXIS-NEXIS commercial system provides the user with a list of related terms that can be automatically incorporated into his search request. For this exercise, we logged on to the CURNWS file within the NEWS library of the online system, and transmitted the REL command along with a term selected from the query's title or the description field. The system returned a list of several dozen related terms, including multi-word terms.

For the automatic entries, we decided to repeat the query processing steps that gave us the most significant boost in TREC-6. During our preparation for TREC-6 [1], we experimented with highlighting query terms that were classified by WordNet [9] as nouns. We then added potential phrases and synonyms based on WordNet. This year we just performed the first step. The addition of synonyms consistently gave us poor results, so we skipped this step for our final processing. For the first query-processing step, we highlighted nouns that were identified by a table look-up of nouns found in WordNet by incrementing their frequencies by one. In addition, we also detected phrases in the query that were also present in the TREC-7 corpus.

The improvements due to phrase-detection were not as large as we had expected. Quite a few important phrases in the corpus were missed because of a bug in the phrase-detection program. Table 3 shows the improvement we gained in the ranking after applying the above technique.

TITLE + DESC (TRAINING)				*Precision*	
Run	***Rel_Ret***	***Avg. Precision***	***Exact Precision***	***At 0.20***	***At 20 docs***
Baseline	2377	.2560	.2741	.4749	.3620
Nouns+Phrases	2542	.2560	.2885	.4747	.3640

Table 3. Improvements gained by adding nouns and phrases.

1.2.2 - Adding More Terms Using LEXIS-NEXIS' REL Feature

Last year we found that adding related terms from the LEXIS-NEXIS online system helped us retrieve a larger number of relevant documents in the LNaShort (title only) entry [1]. This work was done after Voorhees [10] had found that related terms help by enhancing recall in the retrieval process. We decided to use this technique for the two automatic entries in TREC-7. We wanted to use the new co-occurrence analysis process to filter out noisy query terms added by the LEXIS-NEXIS REL feature. During training, we realized that precision and recall statistics were drastically effected by the inclusion of the related terms, but they seemed to bring in relevant documents not present in the original Nouns+Phrases run. We therefore retained this run to serve as input into the data-fusion step. Table 4 shows incremental improvements due to REL+Co-occurrence and data-fusion steps.

TITLE + DESC (TRAINING)				*Precision*	
Run	***Rel_Ret***	***Avg. Precision***	***Exact Precision***	***At 0.20***	***At 20 docs***
Nouns+Phrases	2542	.2560	.2885	.4747	.3640
Nouns+Phrases+REL+Co	2243	.2240	.2582	.3976	.3060
Data-Fusion	2526	.2620	.2898	.4493	.3810

Table 4. Incremental improvements due to REL + Co-occurrence and Data-Fusion.

1.2.3 - Query Term Dependency and Query Term Coverage Factors

For the title only entry (LNaTitle7), we also added term dependency and term coverage factors to improve the precision in the initial (pre-relevance feedback) run. The term dependency technique attempts to capture multiple concepts within the title. We give more weight to terms that are physically separated from each other than to those that are close together. The assumption behind this approach is that normally within short queries (2-3 terms), two terms located adjacent to each other are about the same concept. The probability of two terms not adjacent to each other being of different concepts increases as the distance between them increases. The query term coverage factor ranks higher the documents with a larger number of query terms. The assumption here is that the title fields of TREC topics are very specific. Hence, the larger the number of query terms in the document, the more on-point it is. These techniques can only be used in queries with 2-3 terms because longer queries become less focused and we can't assume that all query terms are equally important. To compound this problem, longer queries result in larger inter-term distance factors that confuse the original probabilities of relevance of various documents. The benefit of adding the term dependency factor and query term coverage processing can be seen in Table 5.

TITLE ONLY (TRAINING)				Precision	
Run	*Rel_Ret*	*Avg. Precision*	*Exact Precision*	*At 0.20*	*At 20 docs*
Nouns+Phrases	2231	.2250	.2576	.3988	.3300
Nouns+Phrases+trmDep	2265	.2386	.2791	.4291	.3450
Nouns+Phrases+trmDep+Cov	2235	.2382	.2792	.4299	.3450

Table 5. Incremental improvements due to query term dependency and coverage.

1.2.4 - Relevance Feedback

This year we were planning to experiment with both LDA [6] and Rocchio [11] relevance ranking approaches, but due to resource constraints, we were left with just enough time to submit the Rocchio runs. We performed Rocchio relevance ranking on the two automatic entries. We found that we got consistently better results after the inclusion of a factor representing non-relevant documents to our last year's Rocchio formula. Our Rocchio re-weighting formula was:

*8 * original query vector + 4 * average relevant vector - 4 * average non-relevant vector*

where *average relevant vector* consists of terms from the top ranked documents, and *average non-relevant vector* consists of terms from the documents ranked at the bottom of the ranked list. The results of our Rocchio processing and subsequent fusion with original ranking can be found in Table 6.

TITLE + DESC (TRAINING)				Precision	
Run	*Rel_Ret*	*Avg. Precision*	*Exact Precision*	*At 0.20*	*At 20 docs*
Nouns+Phrases	2542	.2560	.2885	.4747	.3640
Rocchio	2616	.2657	.3054	.4495	.3690
Data-Fusion	2685	.2749	.3080	.4771	.3850

Table 6. Incremental improvements due to Rocchio and Data-Fusion processes.

1.3 - Data Fusion

It is a well-known fact that different ranking algorithms and different query processing techniques retrieve different sets of documents. Belkin [16] used probability theory to arrive at a technique that merges results from different ranking techniques. If the fusion is done right, merging different sources of evidence (rankings) will improve the retrieval effectiveness, because the merged results will contain the best documents from all the sources. During preparation for the TREC-6 conference, we had experimented with various data-fusion techniques. We settled on a technique that gave us consistently better results. We used this technique for the TREC-7 conference too. We have found that data-fusion improves both precision and recall. Results in Table 6 show the improvement gained over the baseline algorithms as a result of the application of the data-fusion step.

1.4 - Co-occurrence Metrics

Most IR techniques are based on standard statistical measures that use the term frequency information within the text to determine whether a document is relevant to a query. Unfortunately, query terms can be used in multiple contexts with distinct meanings, so merely looking at term frequencies is not enough. When we try to add terms to the queries to improve recall, the problem is magnified. The reason is that we tend to add terms that expand the alternate meanings of the query terms along with the terms that belong to the same concept as the query. Several techniques exist that counter this effect. One of the most popular techniques has been the use of the mutual-information metric [17]. We investigated this and other techniques and arrived at a hybrid approach to evaluate the importance of various terms with respect to the original query terms.

Co-occurrence and Its Importance in Query Enhancement

To determine the best terms to add to a query, we began with the terms in the title of each TREC topic. These terms are known to be relevant to the respective query because of the way that they were created [12]. We use these terms as anchors to add new terms. We make an assumption that terms that frequently co-occur with all the title terms have a good probability of being relevant to the query, and are therefore added to the query. Terms that don't co-occur with all the title terms are eliminated.

There are many methods for measuring co-occurrence (e.g. Dice's coefficient, Jaccards's coefficient, cosine coefficient, the overlap measure, and many others [14]). Some of these measures work better than others in different situations. We experimented with several of these methods and derived one of our own that worked well on TREC data.

2 - Description of Entries

2.1 - LNaTitleDesc7

LNaTitleDesc7 entry used the title and the description fields of the TREC-7 topics. Two initial ranking runs were performed as a precursor to the Rocchio process. For the first ranking run, we enhanced the topics as described in section 1.2. We chose a multiplication factor of two to ensure that title terms were more significant for retrieval than the description terms.

For the second run, we added REL terms from the online system. To maintain the focus of the topics, weights of the title terms were tripled, the weights of the description terms were doubled, and both sets were added to the related terms from the online system. The new set of topics was pre-processed by the new co-occurrence metric before being ranked.

The output of the two initial rankings was combined by the data-fusion process. The combined ranking was then processed by a Rocchio relevance ranking process as described in section 1.2.4. We again processed the newly added terms through our co-occurrence analysis process to weed out terms that were deemed superfluous, or not on-point, before re-ranking the results. The results were later merged with the original Title+Description ranking to arrive at the final LNaTitleDesc7 ranking. Table 7 and Figure 1 show the results that were obtained after running the TREC-7 evaluation program after each intermediate step of the LNaTitleDesc7 entry.

LNaTitleDesc7					*Precision*	
Step	***Run***	***Rel_Ret***	***Avg. Precision***	***Exact Precision***	***At 0.20***	***At 20 docs***
1	Baseline	2442	.2030	.2559	.3712	.3950
2	1 + Nouns + Phrases	2581	.2127	.2628	.3672	.4050
3	2 + REL	2586	.2310	.2700	.3908	.3860
4	Co-occurrence on 3	2870	.2405	.2807	.3969	.3980
5	Data Fusion of (2+4)	2899	.2410	.2867	.4105	.4120
6	Rocchio on 5	3112	.2418	.2714	.3870	.3860
7	Co-occurrence on 6	3106	.2427	.2734	.3970	.3860
8	Data Fusion of (2+7)	3020	.2394	.2783	.4073	.4050

Table 7. The results of incremental steps in the LNaTitleDesc7 entry.

By re-weighting nouns and detecting phrases, we were able to get a 4.8% improvement in retrieval performance. Unlike the training run (Table 4), addition of REL terms from the LEXIS-NEXIS online system didn't hurt the precision as much while bringing more relevant documents to the top 1000. The co-occurrence analysis step (Step 4) helped us eliminate the noisy REL terms to improve the retrieval by 13% over Step 2. The data-fusion step (Step 5) improved the precision further especially among the top 20 documents. We then applied the Rocchio relevance feedback approach to Step 5 to get a new set of queries. The retrieval performance of Rocchio seems to be consistent with our earlier observations. We were able to bring in more documents to the ranked-list at the expense of precision at the top 20 documents. The co-occurrence analysis step improved the precision further although some of the relevant documents were lost because some good terms that did not co-occur with title-terms were also eliminated. The final data-fusion step was undertaken as a conservative measure to ensure that the focus of the original queries was not lost. This step undid

some of the improvements we had gained in our previous steps. As an after-thought, we could have done without this step.

The co-occurrence analysis steps in the LNaTitleDesc7 entry proved to be quite effective in improving the precision. Co-occurrence analysis provides an automatic approach for choosing terms that are more important than others for query expansion.

2.2 - LNaTitle7

The LNaTitle7 entry used the title field of the TREC-7 topics for retrieval. Figure 4 illustrates the processing steps that were taken to create the LNaTitle7 entry. Two initial ranking runs were performed before the Rocchio process. The first ranking run had as input the title terms with nouns being enhanced and phrases being identified as described in section 1.2. The ranking took into consideration term dependency and term coverage of the queries.

The second run used the REL terms from the online system and the title terms whose weights had been altered to maintain the focus of the original topics. The new topics were pre-processed by the co-occurrence analysis process to remove extraneous terms. The documents that were retrieved using these new topics were ranked based on the basic formula of the Ocelot algorithm (i.e. without the term dependency and term coverage processing).

The output of the two initial rankings were combined by the data-fusion processes, and the combined ranking was then processed by a Rocchio relevance ranking process as described above. We again processed the newly added terms through the co-occurrence analysis process before ranking the documents for the final time. The final ranking was then fused with the initial 'title' rank, and the output was named LNaTitle7. Table 8 and Figure 2 contain actual values obtained after each intermediate step in the LNaTitle7 entry.

LNaTitle7					*Precision*	
Step	*Run*	*Rel_Ret*	*Avg. Precision*	*Exact Precision*	*At 0.20*	*At 20 docs*
1	Baseline	2178	.1807	.2320	.3308	.3440
2	1 + Nouns + Phrases	2197	.1877	.2408	.3323	.3550
3	2 + REL	1740	.1577	.1981	.2946	.2960
4	Co-occurrence on 3	2618	.2302	.2667	.3860	.3800
5	2 + coverage + termDep	2137	.1892	.2420	.3368	.3810
6	2 + termDep	2241	.1955	.2452	.3413	.3820
7	Data Fusion of (6+4)	2699	.2310	.2728	.3915	.4040
8	Rocchio on 7	2923	.2444	.2700	.4043	.3790
9	Co-occurrence on 8	2916	.2410	.2702	.4096	.3790
10	Data Fusion of (6+9)	2791	.2338	.2691	.3887	.3910

Table 8. The results of the incremental steps in the LNaTitle7

Step 1 shows the ranked results of unmodified queries. After we detected phrases and re-weighted noun terms, we got a boost of just 3.8%. Addition of REL terms from the LEXIS-NEXIS online system reduced both the precision and the recall figures. However, application of the co-occurrence analysis process on the new REL-modified queries improved the average precision by 22.6% to .2302.

Steps 5 and 6 show the results of using term coverage and term dependency factors. Using the term dependency factor improved both the recall and the precision. We got better results by omitting the term coverage factor as can be seen when we compare Step 6 and Step 5.

The data-fusion step, (Step 7), resulted in a marginal improvement of both the precision and recall. The new Rocchio approach helped improve the precision and recall figures by 5.8%. The co-occurrence step after Rocchio and the subsequent data-fusion step didn't add any value to the ranking.

The final data-fusion step had a major negative impact on our results. We believe that the Rocchio relevance feedback run must have retrieved many on-point documents that are ranked higher up in the ranked list. So mixing relevance ranking with a noisier ranking (output of Step 6) automatically added noise to the final results.

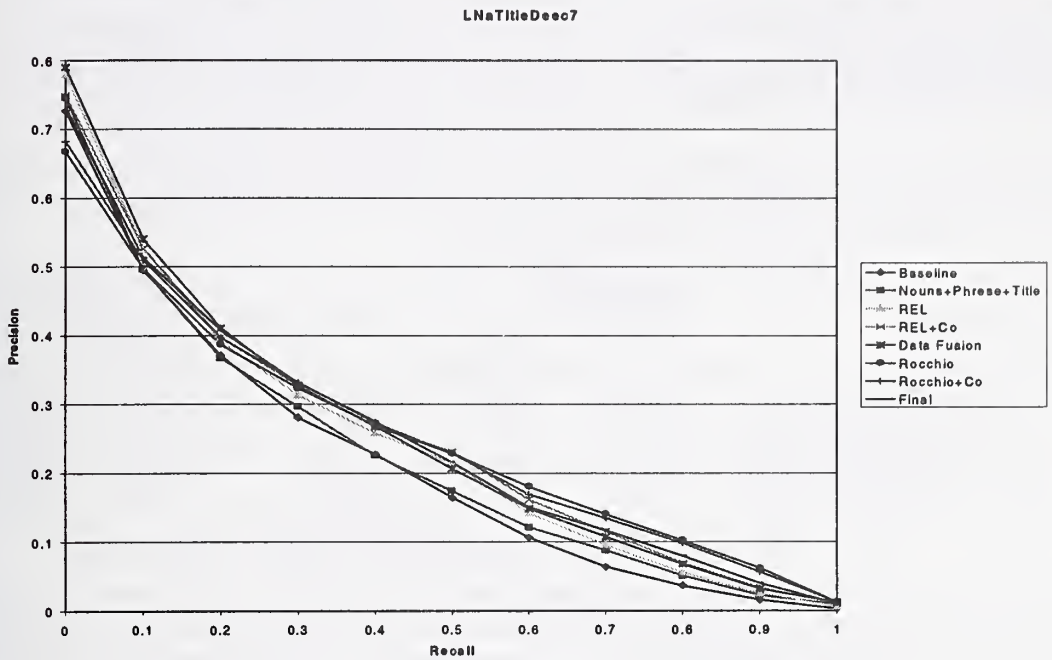


Figure 1. Recall/Precision graph of the processing steps for the LNaTitleDesc7 entry.

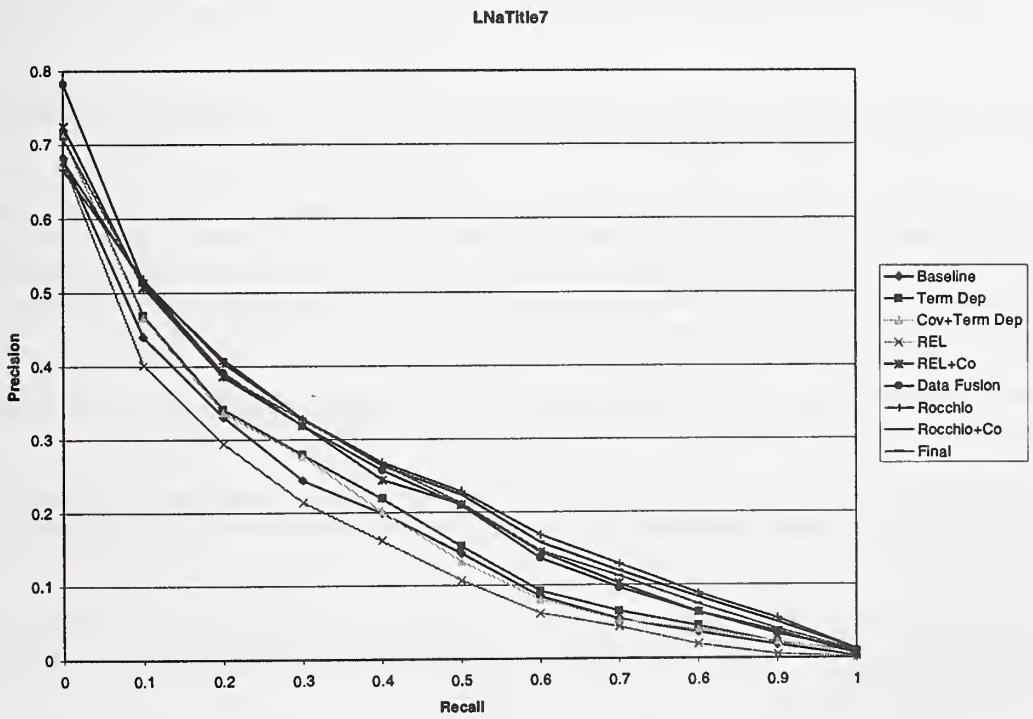


Figure 2. Recall/Precision graph of the processing steps for the LNaTitle7 entry.

2.3 - LNmanual7

We used the Ocelot algorithm to create a sample of 20 documents for each query that human analysts could use to refine the query terms for the manual entry. The algorithm used terms from the title and the description fields of TREC-7 topics. We doubled the weight of the terms in the title, incremented nouns by one and identified and added phrases. No terms from the narrative field were involved in this process. We evaluated these terms and either replaced, re-weighted, or supplemented them using terms from the top 20 documents, the narrative field, and the LEXIS-NEXIS' REL.

Table 9 and Figure 3 present the results of our manual entry. We gained a 28% improvement on the average precision by manually enhancing the queries. The largest gain is at 0.20 precision (i.e. 32%) indicating that the manual intervention improved the ranking of some on-point documents.

LNmanual7					Precision	
Step	**Run**	**Rel_Ret**	**Avg. Precision**	**Exact Precision**	**At 0.20**	**At 20 docs**
1	Baseline	2442	.2030	.2559	.3712	.3950
2	1 + Nouns + Phrases	2581	.2127	.2628	.3672	.4050
3	Manual on 2 (Final)	3005	.2722	.3190	.4848	.4490
4[2]	Co-occurrence on 3	2922	.2606	.3089	.4770	.4510

Table 9. The results of the incremental steps in theTREC-7 manual entry.

The co-occurrence analysis process (Step 4) which has improved the performance of the algorithm in the automatic entries did not have the same contribution as the human judgment. One explanation is that the human analysts made a judgment about each word based on their world knowledge and experience. The co-occurrence analysis process increases precision by eliminating terms that do not co-occur with the title terms. Some of the terms added by the analysts are relevant to the topic even though they don't co-occur with the title terms. These are the terms that are eliminated by the co-occurrence process. Analysis at the query level indicated that the average precision has increased when the co-occurrence analysis was applied to the manual run (Step 3), but because important terms were eliminated, the actual number of relevant documents retrieved was lower. This resulted in higher recall values in Step 3 versus Step 4 as indicated in Figure 3.

Because time was short, we had only one chance to select terms and re-run the document selection. It was not possible to fine-tune the term selection or have multiple iterations to see what kinds of terms were useful.

We could only add phrases to the query that have been recognized by the indexing process. In many occasions, the existence of one unambiguous phrase in a document could indicate relevance but we could not add it to the query because the indexing program hadn't recognized it as a phrase. For example, the phrase *human cargo* is a unique combination of terms for a query about *human smuggling* (TREC-7 topic #362). Unlike the phrase *human cargo*, the two individual terms *human* and *cargo* are very common and they are repeated in many documents.

The absence of proximity indicators was another limiting factor. For example, "technology transfer" and "illegal" may occur within a document but if they are in close proximity, the meaning and relevance change.

We couldn't use numbers although they were significant relevance indicators in some queries. For example, documents about international waters disputes often mentioned the "12 mile" limit and relevant documents about blood alcohol fatalities required a report on the blood alcohol level of the driver.

One of the decisions that the human reader had to make for each added query term was the weight that had to be assigned to it. Terms from the title proved to be more relevant than either the description or the narrative [12]. Therefore, they were usually assigned a much higher weight.

[2] Not submitted

Figure 4 compares our three entries in TREC-7. The contribution from terms in the description field was minimal this year because the LnaTitle7 and LnaTitleDesc7 have almost identical results. A subjective survey of TREC-7's 50 topics showed that in at least 9 topics, the description section did not contribute any non-noise words to the query. The median number of useful words in all 50 topics was just 3.

The performance of our manual entry exceeded the automatic entries. This is due to the differences in the term expansion processes. The expansion process in the automatic entries is blind to terms in the expansion set. These terms are pulled in simply by the virtue of occurring in the top ranked documents. On the other hand, the query expansion terms in the manual entry were highly filtered through human experience.

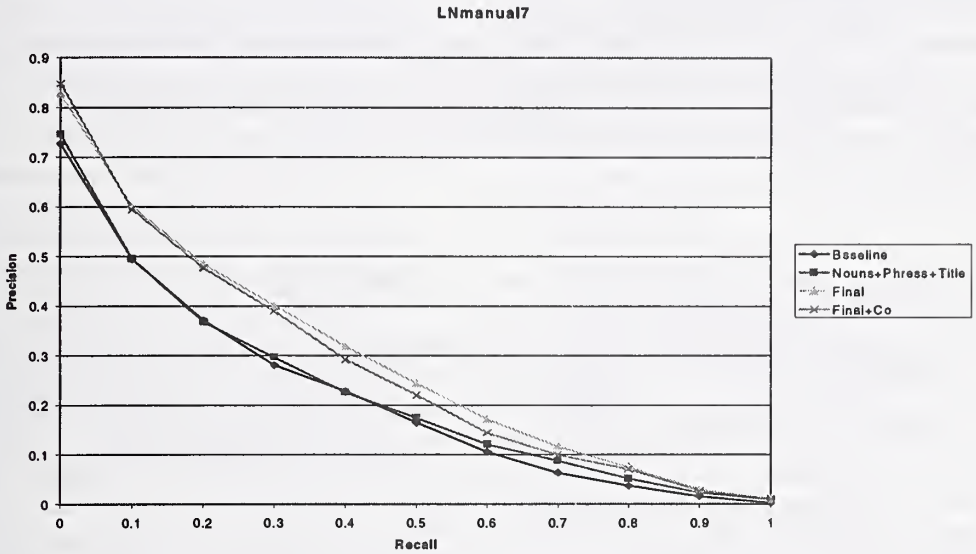


Figure 3. Recall/Precision graph for processing steps for the LNmanual7 entry.

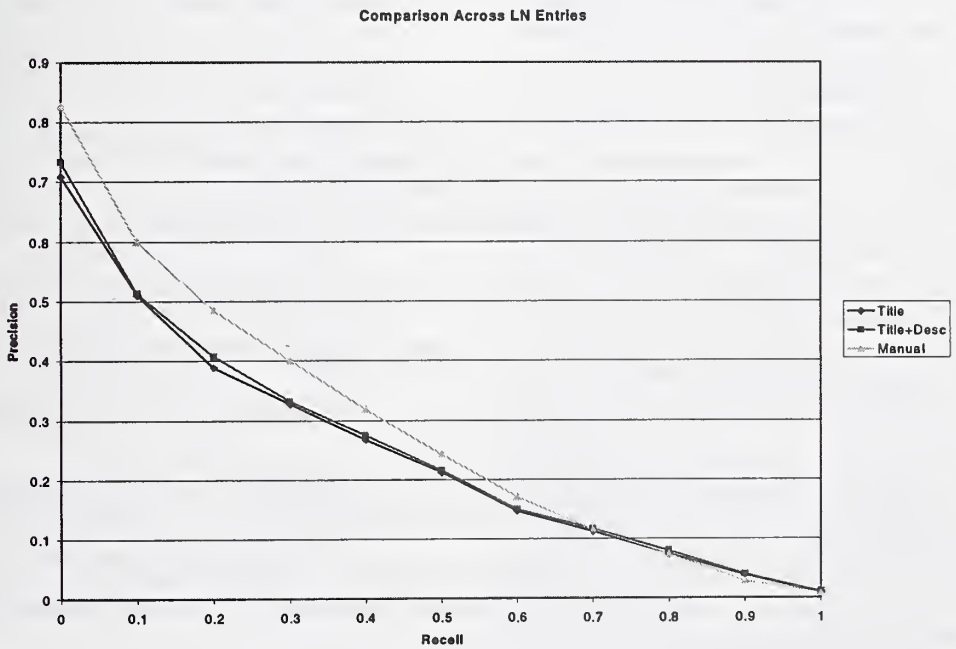


Figure 4. Comparison across all LEXIS-NEXIS entries.

3 - Conclusion

This report described our research effort for TREC-7. We performed a comparative analysis of several well-known search algorithms, and we improved our query enhancement techniques. While we were researching these issues, we developed two new tools to help us create more robust and scaleable search solutions. The first tool helps us fine-tune the co-occurrence metrics. This technique has many applications in the commercial world such as improving text filtering, identification of related concepts and text classification. The second tool helps us distribute the TREC processing over multiple systems to increase the processing speed.

In summary, we found that the new co-occurrence analysis step improved our precision without significantly degrading the recall. We believe that the experience gained by participating in TREC-7 was instrumental in answering some of our research questions but there are still other research issues that we need to explore. We experimented with variations of query term coverage factors, but we didn't come up with a consistent way of improving ranked-results. Another research issue that was raised during this study was the need for finding better ways to identify on-point documents for the relevance feedback process. We also need to find out ways to adjust the performance of our system based on the specific genre of the corpus.

References

[1] Lu, A., Meier, E., Rao, A., Miller, D., and Pliske, D. in "Query Processing in TREC-6", The Sixth Text Retrieval Conference, TREC-6 Notebook, 1997.

[2] Lu, X. A., Ayoub, M., and Dong, J. in "Ad Hoc Experiments using EUREKA" The Fifth Text Retrieval Conference, NIST Special Publication 500-238, pp. 229-240, edited by Voorhees E.M. and Harman D.K.

[3] Lu, X.A., and Keefer, R.B. in "Query expansion/reduction and its impact on retrieval effectiveness", The Third Text Retrieval Conference, NIST Special Publication 500-225, pp. 231-239, edited by Harman D., 1995.

[4] Robertson, S.E., Walker, S., Jones, S., Hancock-Beaulieu, M.M., and Gatford, M. in "Okapi at TREC-3", The Third Text Retrieval Conference, NIST Special Publication 500-225, pp. 109-126, edited by Harman D, 1995.

[5] Broglio, J., Callan, J.P., Croft, W.B., and Nachbar, D.W. in "Document Retrieval and Routing Using the INQUERY System", The Third Text Retrieval Conference, NIST Special Publication 500-225,, pp. 29-38, edited by Harman D, 1995.

[6] Allan, J., Ballesteros, L., Callan, J.P., Croft, W.B., and Lu, Z. in "Recent Experiments with INQUERY", The Fourth Text Retrieval Conference, NIST Special Publication, pp. 49-64 edited by Harman D. 1996.

[7] Singhal, A., Salton, G., Mitra, M., and Buckley, C. in "Document length normalization," Information Processing & Management, Vol. 32, pp.619-633, 1996.

[8] Allan, J., Callan, J., and Croft, W.B. in "INQUERY does battle with TREC-6", The Sixth Text Retrieval Conference, TREC-6 Notebook, 1997.

[9] Miller, G. in "WordNet: An online lexical database", International Journal of Lexicography, Vol. 3(4), 1990

[10] Voorhees, E.M. in "On expanding query vectors with lexically related words," The Second Text Retrieval Conference, NIST Special Publication 500-215, pp. 223-231, 1994.

[11] Rocchio, J.J. in "Relevance Feedback in Information Retrieval", The SMART Retrieval System, pp. 313-323, edited by Salton G.

[12] Voorhees, E.M., and Harman, D. in "Overview of the Sixth Text REtrieval Conference (TREC-6)", The Sixth Text Retrieval Conference, TREC-6 Notebook, 1997.

[13] Robertson, S.E., and Sparck, J.K. in "Relevance weighting of search terms", Journal of the American Society for Information Science 27: p129-146, 1976.

[14] Boyce, B.R, Meadow, C.T., and Kraft, D.H., Measurement in Information Science, Academic Press, pp. 86-87, 1994.

[15] Turtle, H.R., and Croft, W.B in "Inference Networks for document retrieval", Proceedings of the 13th International Conference on Research and Development in Information Retrieval, pp. 1-24, New York: Association for Computing Machinery, 1990.

[16] Belkin, N.J., Cool, C., Croft, W.B., and Callan, J.P. in "The effect of multiple query representations on information retrieval performance", Proceedings of the 16 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 339-346, 1993.

[17] Church, K., and Hanks, P. in "Word association norms, mutual information, and lexicography", Proceedings of the ACL Meeting, pp. 76-83, 1989.

READWARE® TEXT ANALYSIS AND RETRIEVAL IN TREC 7

By Tom Adi, O. K. Ewell and Patricia Adi
Management Information Technologies, Inc.
Email: mitioke@readware.com

ABSTRACT

This paper describes Management Information Technologies, Inc.'s (MITi) first involvement in the TREC program. We limited our participation to manual adhoc although our multilingual algorithms can be used for automatic query generation and refinement and are suited for most TREC tasks.

We used our commercially available text analysis and retrieval Readware technology to perform the manual adhoc task of finding the documents relevant to fifty specified topics in a pool of more than half a million documents. Readware uses concepts (groups of related words), superconcepts (groups of concepts), Readware query elements (query building blocks) and document subjects to form queries. This is complemented by word search, phrase search and Boolean logic.

One MITi analyst performed the task. She formulated an average of 18 queries per topic. The queries were derived intuitively from topic specifications (title, description and narrative). First, a baseline pool of documents was identified for every topic using a few simple queries. Then the analyst queried the baselines using as often as possible Readware query elements related to elements of the topic specification. On average, few hits were returned per query. The analyst also had the advantage of seeing the exact responsive text spots highlighted in every hit document. Queries were adjusted and expanded using information from the neighborhood of the highlighted hit spots. There was no intrinsic ranking of hits. All hits were full semantic matches. The hits were ranked higher after the fact if the queries contained more items.

In the "Best Manual Adhoc" figure of the TREC 7 evaluation results, MITi's graph is above all other participants' graphs at most points.

1. INTRODUCTION

MITi has been developing text analysis and retrieval techniques under the trade name Readware for over a decade. This is our first participation in TREC. We submitted one run in the manual ad hoc category.

Readware currently uses the following conceptual sets to analyze text:

a) a few thousand concepts (groups of related words)
b) a few dozen superconcepts (groups of concepts)
c) a few hundred query elements (query building blocks) composed of superconcepts, concepts, words and phrases
d) 46 document subjects identified by concepts

Readware query elements include:

a) query helpers: frequent questions such as "who," "where," "why," "what does it mean"
b) Readware topics: useful extended concepts such as courts, leaders, safety, medicine, military and business
c) issues people use to make critical decisions, such as "emerging needs," "potential trouble," and "checking on those in charge."

There are three basic search strategies: word search, concept search and superconcept search. Readware's selectable query elements simplify the art of asking questions.

Users may mix strategies using a different strategy for every item. A variable-size sliding search window scans each document for certain words, phrases, concepts and superconcepts. The window size (context size) can be set in the query to values ranging from one tenth the query size to 20 times the query size.

Queries may also contain document-level inclusion or exclusion of words and phrases. And finally, Readware incorporates Boolean

logic to combine words, phrases, concepts and query elements into a compound query.

MITi participated in the manual adhoc task. One MITi analyst used Readware to prepare and query over 500,000 test documents. The goal was to identify all the documents discussing each of the fifty TREC 7 topics. Hit documents were required to be ranked and a maximum of 1000 hit documents were expected per topic.

Since all Readware hits must have a complete set of full semantic relations with the query and Readware no longer ranks hits by semantic points, we only looked for "good hits" without ranking. After establishing a rather small baseline set of documents for every topic, queries made to the baseline returned few hits on average. The exact hit spots were highlighted and the analyst was able to judge the hits rather quickly. Queries were refined and in the end, most of the hit documents we delivered were already judged likely relevant from the analyst's point of view.

We delivered for evaluation a total of 5898 ranked hit documents. For the purposes of TREC evaluation, we ranked the hits by the complexity of the queries used. The more items and positions the generating queries contained, the higher the hit rank. We did not rank hit documents higher if they contained more than one hit spot.

2. DATA PREPARATION

We used a Pentium II (266 MHz) with 128 MB of RAM and a 4 Gigabyte disk. A fully automatic data preparation took about 8 hours of CPU time.

Once the TREC 7 files were decompressed, they were ready for automatic processing by Readware. It took some minor programming to exclude certain fields such as subject, headline and header. Our compiler split the files into documents using the <DOC> and <DOCNO> tags. This was done without physical duplication by keeping track of document lengths and their positions in the original files. Our default tag filter made sure that tag contents were skipped.

Every document was analyzed to determine the positions of words, concepts, phrases and query elements. Identifying Readware query elements meant asking about a million questions to every document using a variable-size sliding search window. Document subjects were identified using concept frequencies at the tops of documents.

The results of the analysis were stored in 4 files:

docs._ (42 MB):	table of vital data per document: document number, file number, position in file, document subject, document issues, etc.
list._ (71 k):	TREC file list containing documents
sigs._ (931 MB):	Readware signature database of positions of words, concepts and query elements in every document.
optdx._ (155 MB):	Optimized index

The text analysis consulted the 2 MB Readware Concept Base which consists of several files.

3. QUERY CONSTRUCTION

To save search time, we first limited the scope of search to a small pool of topic-related documents. To identify such documents, we asked simple questions.

> For TREC topic 372 "Native American casino," we searched for the words casino or gambling combined with Indians, Native Americans, tribes or reservations. The search identified about 260 TREC documents. This became the pool for further searches and we called it a "baseline."

Most TREC queries were asked within a baseline pool, and sometimes the baseline was expanded during the search. Setting a baseline also established the maximum possible number of relevant documents. Baselines were very efficient. For example, searching the baseline of topic 372 for the

word "casino" brought a majority of good hits.

Then, we looked at TREC topic specifications composed of title, description and narrative and tried to identify the basic elements.

> TREC topic 372 "Native American casino" contains the elements: growth of gambling, social implications, economic effects on community and tribe, and legal aspects related to tribal autonomy.

Our search got off to a quick start by clicking on the Readware query elements which corresponded to the basic elements of the TREC topic. Experience showed this was the fastest way to find relevant documents.

> The query "Indian casino" combined with query helper CONSEQUENCES or query helper WHY found implications and effects. Combined with the Readware topic POLICE or the collective issue ALL ISSUES, this search captured many hits related to community disruption. We did not have to ask specific questions about social implications, economic effects and legal aspects.

Refining queries by combining (AND) increased precision and recall.

> The queries "Indian gambling" and "tribal casino" within the baseline pool brought a mix of good and bad documents. But "Indian gambling" AND the Readware topic POLICE brought 7 clean hits. And "tribal casino" AND the query helper WHY found 8 good hits.

Queries were also refined by excluding (NOT) unwanted words, concepts, phrases, query elements (also at document level) and excluding document subjects.

> "Indian gambling affairs" but NOT the word (Trump) brings 14 clean results.

Using alternative query formulations and strategies made the search more exhaustive.

> Alternative queries included "reservation gambling," "Indian gambling" and "Native American casino." The queries can be searched with word search or concept search. The search "window" (context size) may be small or large. Mixed strategies are possible in one query-- partly word, partly concept and partly document wide search.

Compound queries show relatedness and focus the search process.

> Casino/gamble/gaming AND (query helper PROBLEMS/FAILURE OR query helper SUCCESSES)

In the neighborhood of the exact responsive text spots highlighted by Readware, we found words, concepts and elements which we used in more queries.

> We learned from the text around highlighted responsive spots that Native Americans from the Mohawk tribe ran a casino. When we asked the simple question "Mohawk" within our baseline pool, we got 4 clean hits.

A series of focused queries which bring back a manageable number of results (say, from 1 to 50 at a time) are more satisfying for an analyst to work with than a strategy that requires him to sift through hundreds or thousands of irrelevant responses for a few good ones.

> A total of 14 questions was asked for TREC topic 372. The average number of hits per query was 6. The maximum number of hits was 17; the minimum was 1. Good document hits were marked. The analyst stopped asking questions when no more relevant new documents were discovered.
>
> Out of the 46 hits we delivered for this topic, 43 were judged relevant. Judges found a total of 49 relevant documents.

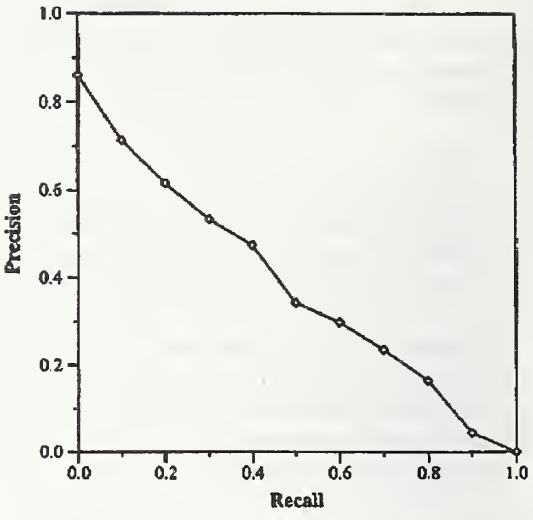
For all 50 topics, we constructed a total of 918 queries, an average of 18 queries per topic. We used document subjects and Readware query elements over 650 times in the queries, i.e. in 7 out of 10 queries.

4. PERFORMANCE

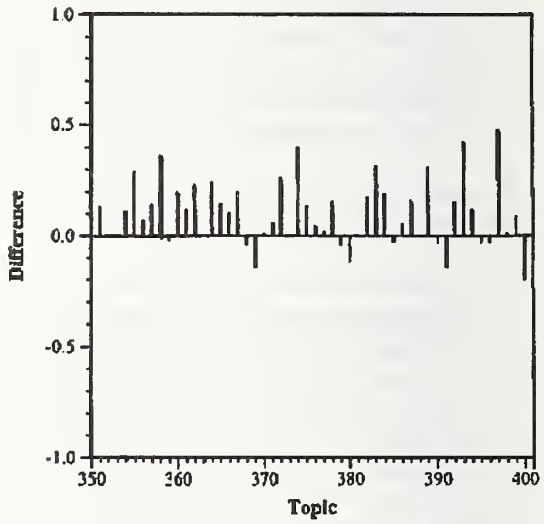
In the TREC 7 evaluation results, MITi is listed as one of the best manual ad hoc runs. Our graph (t7miti1) appears above all other graphs at most points of the comparative recall/precision figure.

MITi scored the highest R-Precision (precision after R documents retrieved) at 0.4392 (second-highest is Claritech at 0.4140). We achieved the second-highest average precision over all relevant documents at 0.3675 (just below Claritech's 0.3702).

MITi delivered the smallest number of retrieved documents, 5,898 (followed by University of Waterloo's 16,617) but we had the second-highest number of unique contributions, more than 160 (following Waterloo's 200 or so).



Recall-Precision Curve



Difference from Median in Average Precision per Topic

HIGHLIGHTS

Summary Statistics	
Run Number	t7miti1
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	**5898**
Relevant:	4674
Rel-ret:	2520

Recall Level Precision Average and Selected Document Level Averages

Average Precision over all relevant documents	
Non-interpolated	**0.3675**
R-Precision (after Relevant docs retrieved)	
At 5 docs	0.6640
At 10 docs	0.6400
At 15 docs	0.6213
At 20 docs	0.5780
At 30 docs	0.5433
At 100 docs	0.3512
Exact	**0.4392**

TREC 7 Ad Hoc, Speech, and Interactive tracks at MDS/CSIRO

Michael Fuller* Marcin Kaszkiel* Dongki Kim†‡ Corinna Ng* John Robertson†§ Ross Wilkinson†¶
Mingfang Wu* Justin Zobel*

1 Overview

For the 1998 round of TREC, the MDS group, long-term participants at the conference, jointly participated with newcomers CSIRO. Together we completed runs in three tracks: ad-hoc, interactive, and speech.

2 Ad-hoc task

In TREC-5 we used document retrieval based on arbitrary passages [8, 9], or fixed-length passages that could start at any word position. Although far from the best runs in TREC-5, these results were promising, in particular for long documents. In TREC-6 we continued with arbitrary passages, but our main emphasis was on comprehensive factor analysis of successful automatic query expansion and refinements methods in the context of the vector space model [5]. This year we have refined the MG retrieval system to include Rocchio-based relevance feedback. Also, phrase matching has been added. We have continued to use arbitrary passages and combination of evidence for document retrieval.

2.1 System description

An in-house version of the MG retrieval system has been used for all experiments. All experiments were carried out on an Intel Pentium II (300 Mhz) with a single processor and 256 Mb of physical memory.

Queries and documents were matched using the Okapi formulation [13]:

$$sim(q,d) = \sum_{t \in q \wedge d} w_{d,t} \cdot w_{q,t} \qquad (1)$$

with $w_{d,t}$:

$$\frac{(k_1+1)\cdot f_{d,t}}{k_1 \cdot [(1-b)+b\cdot \frac{W_d}{avr\_W_d}] + f_{d,t}}$$

and $w_{q,t}$:

$$\frac{(k_3+1)\cdot f_{q,t}}{k_3+f_{q,t}} \cdot \log \frac{N-f_t+0.5}{f_t+0.5}$$

where k_1, k_3, and b are constants set to 1.2, 1000, and 0.75 respectively, as recommended by the City University group [13]. The value W_d is the length of document d in bytes and avr_W_d is the average document length in the entire collection. The value N is the total number of documents in the collection, f_t is the number of documents in which term t occurs, and $f_{x,t}$ is the frequency of term t in either document d or query q.

Okapi is not easily adaptable to arbitrary passage ranking because parameters k_x and b are tuned to document ranking. Queries and passages are matched using a non-normalised version of the cosine similarity function:

$$sim(q,p) = \sum_{t \in q \wedge p} (w_{q,t} \cdot w_{p,t}) \qquad (2)$$

with weights that have been shown to be robust and give good retrieval performance [1]: $w_{q,t} = (log(f_{q,t})+1)\cdot log(\frac{N}{f_t}+1)$ and $w_{p,t} = log(f_{p,t})+1$.

Automatic relevance feedback was based on the Rocchio formula [12]:

$$Q_{new} = \alpha \cdot Q_{orig} + \frac{\beta}{|R|}\sum_{r \in R} r + \frac{\gamma}{|R'|}\sum_{r' \in R'} r' \qquad (3)$$

where Q_{orig} is a weighted term vector for the original query; R is the set of relevant documents; R' is the set of non-relevant documents; and r and r' are weighted term vectors for relevant and non-relevant document, respectively. Parameters α, β, and γ control the contribution of terms from original query, relevant documents, and non-relevant documents, respectively.

For indexing purposes, documents and queries have been stopped using the stop-list used in our TREC-6 experiments.[1] Single terms have been stemmed with the Lovins algorithm [10]. Two-word phrases are indexed if they satisfy the following conditions:

- individual words of the phrase occur at least 30 times in collection, and
- the phrase occurs at least 10 times in collection.

A detailed description of two-term phrase extraction can be found elsewhere [3].

2.2 Ad-hoc runs

This year we have concentrated on short queries, and have submitted official runs for *title* and *title+description* queries. For the first time we have not submitted a full-topic run.

[1] See Appendix A of the MDS TREC-6 report for a list of stopped terms [5].

* Department of Computer Science, RMIT,
GPO Box 2476V, Melbourne VIC 3001, Australia
{msf,martin,cln,ross,ming,jz}@mds.rmit.edu.au
† CSIRO, Division of Mathematical and Information Science
‡ GPO Box 664, Canberra ACT 2601, Australia
Dong.Ki.Kim@cmis.csiro.au
§ Locked Bag 17, North Ryde, NSW 1670, Australia
John.Robertson@cmis.csiro.au
¶ 723 Swanston St, Carlton VIC 3053, Australia
Ross.Wilkinson@cmis.csiro.au

	5 docs	10 docs	20 docs	200 docs	Avg. Prec.	%Δ
title						
Document	0.424	0.382	0.332	0.124	0.161	0.0
Passage-300	0.424	0.388	0.322	0.127	0.162	+0.6
mds98t	**0.436**	**0.422**	**0.365**	**0.159**	**0.220**	+36.6
mds98t2	**0.440**	**0.426**	**0.359**	**0.159**	**0.218**	+35.4
mds98t-p300	0.432	0.404	0.355	0.159	0.218	+35.4
title+desc						
Document	0.532	0.486	0.397	0.145	0.204	0.0
Passage-300	0.556	0.458	0.375	0.140	0.194	-4.9
mds98td	**0.572**	**0.536**	**0.446**	**0.187**	**0.281**	+37.7
mds98td-p300	0.540	0.508	0.423	0.180	0.261	+27.9
title+desc+narr						
Document	0.580	0.536	0.450	0.167	0.240	0.0
Passage-300	0.524	0.472	0.394	0.154	0.214	-10.8
mds98tdn	0.616	0.554	0.483	0.196	0.285	+18.8
mds98tdn-p300	0.580	0.518	0.444	0.190	0.271	+12.9

Table 1: *TREC-7 ad-hoc results.*

A complete set of results for TREC-7 is shown in Table 1. For completeness, full-topic runs are included. Official runs are shown in bold face.

Runs *mds98t* and *mds98td*, which correspond to title and title+description queries, used the following approach. Single terms and phrases were used; weights of phrases were scaled down by 0.3 to compensate for single-term contributions of the terms of the phrase. Documents were ranked using the Okapi formulation (equation 1) and 1000 documents retrieved. The top ten documents retrieved (that is, set R) were assumed to be relevant and the last 250 documents of the 1000 retrieved (that is, set R') were assumed to be non-relevant. Using the parameter values $\alpha = 1.0$, $\beta = 2.0$, and $\gamma = 1.0$, equation 3 was used to select an *additional* 40 single terms and an *additional* 5 phrases. Each new single term had to appear in at least 2 relevant documents in order to be considered. With the original query terms re-weighted and new terms added to the query, a final set of 1000 documents was retrieved using document ranking.

Run *mds98t* also a simplified form of this approach: phrases were not used, since there was not enough evidence in very short queries to justify their use; as short queries are not likely to retrieve many relevant documents in top 10, only the first 5 documents were assumed to be relevant; and 80 new terms were added to original query with no restriction of minimum occurrence in relevant documents.

Run *mds98t2* (title queries) is an experimental run that explores combination of evidence, as we have done in past TRECs [5, 8]. The scores of documents in mds98t2 are based on a weighted sum of the document scores from mds98t and the document scores based on the arbitrary passage of 300 words. The document scores for passage-based ranking were downweighted by 0.3 to take into account poorer retrieval performance with respect to the relevance feedback run of mds98t.

It is interesting to observe that, for short queries, document retrieval based on arbitrary passages is as effective as sophisticated document ranking. However, as the query length increases, query terms' proximity is not as important as occurrence of multiple query terms in the entire document.

A two-stage relevance feedback achieved a significant improvement for different types of queries. For title and title+description queries, the improvement for average precision was at least 36%. We believe that the improvement is not as significant for full topic queries because the parameter tuning had been based on queries of medium length.

	Best	≥ Median
mds98t	2	29
mds98t2	1	28
mds98td	4	42

Table 2: *Number of queries that had highest average precision or had at least median average precision.*

Table 2 compares our official runs with the automatic runs submitted by other TREC-7 participants. The table shows the number of queries for MDS runs that achieved at least a median average precision or had the highest average precision for a topic.

For short queries, Table 2 does not reflect well the relative performance because all automatic runs were used to derive the best and median values. Despite this, at least 30 title queries performed better than the median run. For the title+description run, only four queries fell short of the median. This is a positive result and indicates that simple relevance feedback can be highly effective.

To test the robustness of Rocchio-based relevance feedback, the same approach has been used on last year's data. Table 3 compares our TREC-7 strategy with the results from last year's TREC. A 37% increase has been achieved over last year's results and a 25% improvement over the Okapi measure. This result is consistent with our TREC-7 results.

Run mds98t2 fell short of expectations. There was no improvement from combining mds98t with arbitrary passage ranking.

2.3 Further experiments

In order to further evaluate document retrieval based on arbitrary passages, we have used expanded queries from document-based retrieval and ranked documents based on best passage. The average precision for title+description queries has decreased by 7.1% from mds98td but for title queries there was no differences. See Table 1 for run mds98t-p300 and mds98td-p300 (note: phrases were not used in those runs).

We have explored many parameters of the Rocchio formulation on past TREC data. Our TREC-7 runs have used

Experiment	5 docs	10 docs	20 docs	200 docs	Avg. Prec.	%Δ
Last year's	0.400	0.370	0.260	0.120	0.204	0.0
Document	0.464	0.426	0.347	0.128	0.224	+9.8
Passage-150	0.496	0.424	0.331	0.122	0.242	+18.6
TREC7 method	0.530	0.460	0.400	0.159	0.280	+37.3

Table 3: *TREC-6 experiments (title+description queries).*

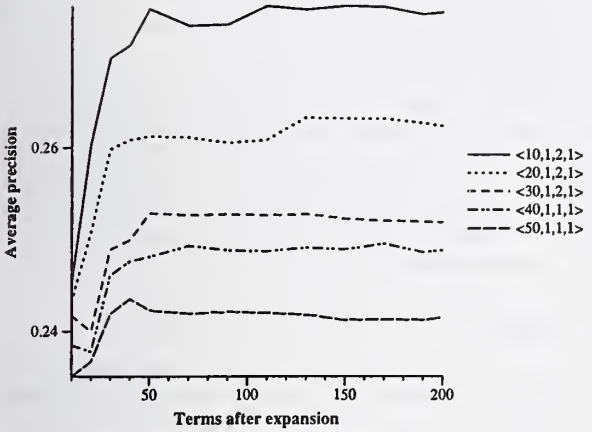


Figure 1: *Rocchio-based relevance feedback on TREC-6 data (title+desc queries).*

parameters that were most consistent over the training data. Here we summarize why particular parameters have been used in TREC-7.

For each R, the best combination of α, β and γ has been identified and presented on single graph. The results are summarised in Figure 1. Each line corresponds to a combination of $\langle |R|, \alpha, \beta, \gamma \rangle$. The figure illustrates that the most consistent results were achieved when R was set to 10, and α, β, and γ set to 1, 2, and 1, respectively. For all cases $|R'|$ was 250; we found in our experiments that negative feedback was useful.

2.4 Analysis

Overall, we conclude the following:

- Using a simple two-stage retrieval, up to 36% gain has been achieved over a single stage retrieval;
- Statistical phrases has improved average precision by 4.1%;
- Document retrieval based on single arbitrary passage is at least as effective as entire document ranking, confirming our past results [9].

3 Interactive retrieval track

The purpose of these experiments was to examine how different organizations of query results affect the ability of users to resolve information needs, focusing on retrieval coverage and efficiency. In particular for TREC-7, the MDS group focused on comparing a *cluster*-based organization with a simple *list*-based organization. Two interactive systems were tested, with the main distinction that where one displayed an ordered list of document titles, the other displayed an ordered list of clusters and their descriptors. The performance of the two systems could be evaluated in two ways: how effective each system was at helping an interactive user identify relevant documents, and how efficient each system was at helping an interactive user identify relevant documents.

To isolate this comparison from other interactive effects, the interfaces of the two systems were kept as consistent as was possible, no querying facility was provided, no relevance feedback was sought, and no query reformulation was possible. As a result, for each topic, the same pool of candidate documents was offered to each subject. Note that these restrictions greatly diminish the role of the user in contrast with other interactive experiments, at the cost of lowering overall performance.

3.1 Retrieval engine

The MG [15] retrieval system was used to identify the pool of candidate documents for each topic. Documents were casefolded, stemmed (Lovins [10]), and stopped.[2] To form queries, the description portion (not title, not narrative) of each TREC-7 topic was case-folded, stemmed, and stopped. Term weights in documents was calculated by using $tf.idf$; term weights in queries used idf. Queries were matched against the document collection using the cosine measure. The top 300 ranked documents formed the pool of candidate documents available to the experiment subjects for each topic.

For the clustered-organization, the pool of 300 candidate documents was clustered using two passes of a single-pass clustering algorithm [4, 16]. The number of clusters for each query was controlled between 7 and 10; the size of each cluster was not controlled. Within a cluster, documents were ranked according to their similarity to the query. Cluster descriptors were formed from the ten highest-weighted terms from the cluster vector, the five most frequent word pairs from all documents in the cluster, and the titles of the three documents in the cluster most similar to the query.

3.2 User interfaces

Given the goal of comparing two alternate organizations of the same data, it was important that the two interfaces be as consistent as possible, differing only in their presentation of the alternate organizations. The design of the interfaces also assumed that relatively large monitors would be available for the interactive experiments, sufficient to permit side-by-side viewing of documents and result organizations. For ease of development, a suite of perl CGI scripts was used to generate the HTML and JavaScript that implemented the interfaces. No mechanism for providing relevance feedback or for supplying a new query was provided; subjects were restricted to exploring the pool of pre-selected candidate documents.

For the list-based organization, the viewport was divided into two parts. (See Figure 2.) The left half displayed a

[2] See Appendix A of the MDS TREC-6 report for a list of stopped terms [5].

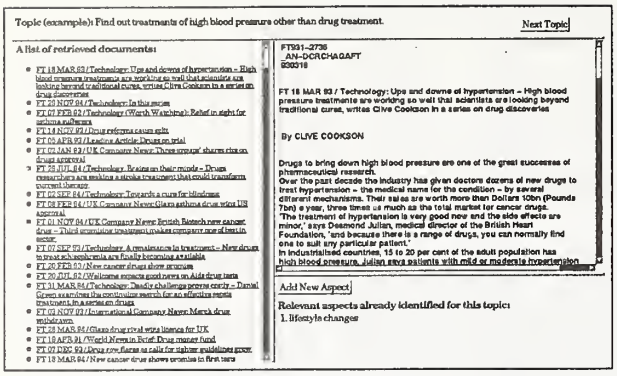


Figure 2: *List-based user interface.*

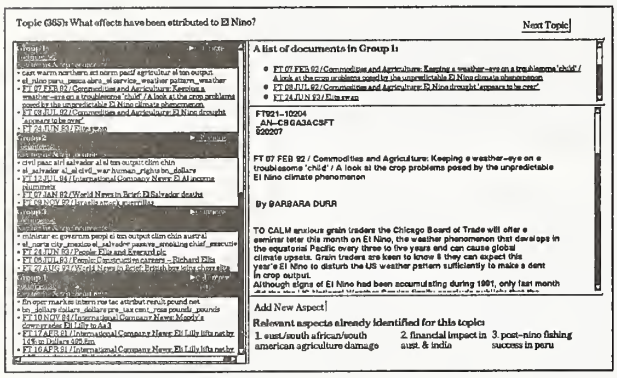


Figure 3: *Cluster-based user interface.*

ranked, scrollable list of the titles of the top 300 documents for a topic; each title could be selected by single-clicking. The upper part of right half of the viewport, initially blank, displayed a scrollable view of any document selected in the left-hand panel. The lower part of right half of the viewport, the aspect selection panel, displayed a list of currently known document aspects; subjects could use this panel to record that a document contained an aspect relevant to the topic and to add a description of the aspect via a pop-up dialogue box.

For the cluster-based organization, the left-hand panel was replaced by an ordered, scrollable list of cluster descriptors; each cluster could be selected by single-clicking. (See Figure 3.) Each cluster descriptor identified the cluster by number, indicated how many documents it contained, provided a list of representative terms and a list of representative term pairs, and listed the three titles of the three 'top' documents from the cluster. The scrollable document view of the list-based interface was divided into two parts. The upper part, initially blank, was used to display a scrollable, ranked list of the titles of documents in any cluster selected in the left-hand panel; each title could be selected by single-clicking. The lower part, initially blank, displayed a scrollable view of any document selected in the upper panel. The aspect selection panel was unchanged.

3.3 Subjects

Sixteen subjects undertook the experiment, according to the Latin Square arrangement stipulated by the TREC-7 Interactive Track guidelines.[3] Their task was "to save documents, which, taken together, contain as many different instances as possible of the type of information the topic expresses a need for—within a 15 minute time limit" from a pool of candidate documents selected from the TREC-7 "Financial Times of London 1991-1994" collection, for each of eight (slightly modified) TREC-7 adhoc topics.

The subjects were undergraduate computer science students, recruited via an internal RMIT newsgroup. The subjects aged from 17 to 23, and had on average 3.3 years of on-line search experience.

The subjects attempted four searches using each system. Pre- and post-experiment, post-search, and post-system questionnaires were administered to each subject, as was the psychometric test FA-1 (Controlled Associations) from ETS "Kit of Reference Tests for Cognitive Factors"' (1976 Edition). Documents identified as relevant by subjects, as well as other "significant" events in each session, were logged and time-stamped automatically.

3.4 Results

The effectiveness of the two organizations was measured in terms of aspectual recall and aspect coverage; the efficiency was measured by the time taken to locate each new aspect. The distribution of the assessed aspects for each topic in the pool of candidate documents is shown in Table 4 and Table 5.

To measure effectiveness, the list of documents whose full text was viewed during each search was extracted from the experiment logs. The aspectual recall of this list can be calculated using either the judgement of the independent NIST assessors, as shown in Table 6, or the judgement of the experimental subjects, as shown in Table 7. The assessors' judgement provides an objective assessment of the quality of the documents that were chosen for viewing, whereas the subjects' judgement reflects their own concept of document's relevance to the topic in terms of their own understanding of the information need.

To measure efficiency, the time to locate each aspect was calculated, again according to both the assessors' judgement, as shown in Figure 4, and the subjects' judgement, as shown in and Figure 5.

There were no significant differences for overall aspectual recall between the two organizations, nor for the time to locate each aspects (paired, one tail t-test).

The pre-experiment psychometric test attempted to gauge subjects' associational fluency in terms of the number of synonyms for a set of eight stimulus words. The mean score for the 16 subjects was 23.2 correct of 34.9 total terms, with a standard deviation of 8.1 terms. There appeared to be a linear correspondence between subjects' FA-1 score and their average aspectual recall; no correlations were found with performance with either interface.

Previous work has indicated that clustering can be an effective mechanism for identifying rich groups of candidate documents [6], and in particular, that clustering can be used to improve ad-hoc query performance in terms of recall-precision [16]. To see if a similar effect would be observed in terms of aspectual recall, we have selected only the documents that are part of clusters from which subjects saved documents. Table 8 shows the aspectual recall of those documents when ordered by their original cluster-ranking and within-cluster ranking. Compared with the baseline (Table 4), aspectual

[3] `http://www.nist.gov/itl/div893/894.02/projects/t7i/`

Topic	Number of documents								
	5	10	15	20	50	100	150	200	300
352	0.000	0.000	0.000	0.036	0.107	0.107	0.107	0.143	0.143
353	0.091	0.182	0.364	0.364	0.364	0.545	0.545	0.545	0.545
357	0.385	0.385	0.462	0.462	0.462	0.615	0.692	0.692	0.692
362	0.000	0.167	0.167	0.167	0.167	0.250	0.333	0.333	0.333
365	0.750	0.750	0.750	0.750	0.750	0.750	0.750	0.792	0.792
366	0.000	0.000	0.000	0.000	0.286	0.571	0.571	0.571	0.857
387	0.111	0.111	0.111	0.222	0.556	0.889	0.889	1.000	1.000
392	0.111	0.417	0.472	0.500	0.639	0.750	0.750	0.806	0.806
Avg.	0.181	0.251	0.291	0.312	0.416	0.560	0.580	0.610	0.646

Table 4: *Aspectual recall for the candidate document pool of the 300 highest-ranked documents.*

	Topic number							
	352	353	357	362	365	366	387	392
Total aspects in all documents	28	11	13	12	24	7	9	36
Total documents containing aspects	120	69	86	116	40	39	88	88
Aspects in candidate documents	4	6	9	4	19	6	9	29
Candidate documents containing aspects	3	13	27	10	3	9	35	30

Table 5: *Aspect coverage for each topic, as judged by the NIST assessors.*

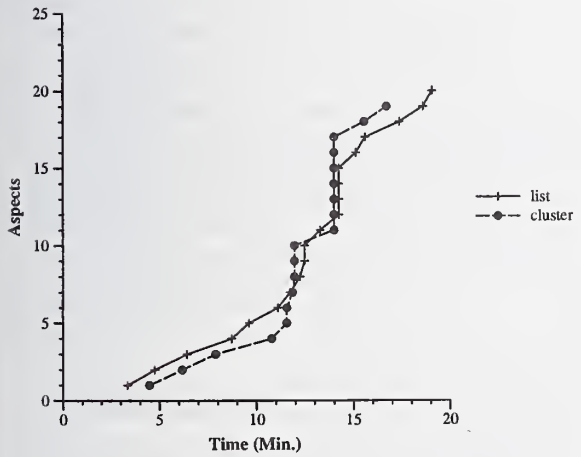


Figure 4: *Time to get each assessor-determined aspect for documents whose full text was viewed.*

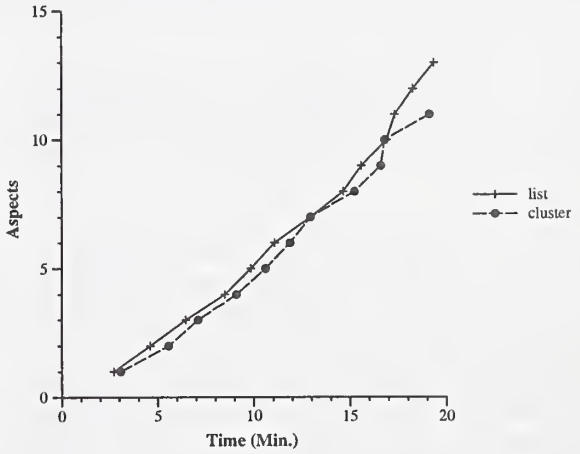


Figure 5: *Time to get each subject-determined aspect for documents whose full text was viewed.*

Topic	Number of docs retrieved				
	5	10	15	20	50
352	0.058	0.103	0.112	0.121	0.125
353	0.057	0.159	0.205	0.227	0.250
357	0.241	0.270	0.318	0.337	0.346
362	0.073	0.146	0.156	0.156	0.167
365	0.490	0.495	0.495	0.495	0.495
366	0.125	0.304	0.304	0.339	0.411
387	0.028	0.069	0.111	0.153	0.194
392	0.080	0.167	0.215	0.233	0.278
Avg.	0.144	0.214	0.239	0.258	0.283

(a) Cluster organization

Topic	Number of docs retrieved				
	5	10	15	20	50
352	0.009	0.049	0.058	0.058	0.058
353	0.057	0.114	0.193	0.239	0.250
357	0.318	0.337	0.366	0.366	0.366
362	0.042	0.136	0.146	0.146	0.156
365	0.693	0.693	0.693	0.693	0.693
366	0.071	0.107	0.161	0.197	0.232
387	0.097	0.139	0.236	0.236	0.305
392	0.226	0.382	0.399	0.413	0.420
Avg.	0.189	0.245	0.281	0.293	0.310

(b) List organization

Table 6: *Cumulative aspectual recall (as judged relevant by assessors) for each topic, at increasing numbers of documents viewed, of documents whose full text was displayed. Note that subjects viewed varying numbers of documents for each topic.*

Topic	Number of docs retrieved				
	5	10	15	20	50
352	0.440	0.688	0.771	0.844	0.875
353	0.320	0.568	0.627	0.669	0.697
357	0.480	0.621	0.701	0.763	0.763
362	0.257	0.632	0.679	0.679	0.798
365	0.369	0.431	0.431	0.431	0.431
366	0.376	0.494	0.607	0.607	0.732
387	0.242	0.575	0.833	0.858	0.929
392	0.355	0.495	0.646	0.705	0.779
Avg.	0.355	0.563	0.662	0.695	0.750

(a) Cluster organization

Topic	Number of docs retrieved				
	5	10	15	20	50
352	0.179	0.408	0.621	0.662	0.737
353	0.531	0.698	0.760	0.760	0.823
357	0.560	0.742	0.861	0.861	0.874
362	0.354	0.651	0.666	0.682	0.724
365	0.692	0.692	0.692	0.692	0.692
366	0.146	0.271	0.396	0.458	0.521
387	0.285	0.600	0.758	0.783	0.840
392	0.294	0.494	0.578	0.634	0.658
Avg.	0.380	0.570	0.667	0.692	0.734

(b) List organization

Table 7: *Cumulative aspectual recall (as judged relevant by subjects) for each topic, at increasing numbers of documents viewed, of documents whose full text was displayed. Note that subjects viewed varying numbers of documents for each topic.*

recall improves over the first 20 documents; the results are truncated at 20 documents as not all clusters contained 50 documents (although results at higher levels still indicate improved performance).

Although no significant difference in terms of either efficiency or effectiveness was found between organizations for the overall results, inspection of the topic-by-topic results suggested that there was in fact a variation in performance for a subset of the topics. Table 9 shows that, for the four topics for which users of the list organization saved the fewest aspects, users of the cluster organization saved highly significantly more topics (paired, one tail t-test). Conversely, for the four topics for which users of the list organization saved the most aspects, significantly fewer topics were saved by users of the cluster organization. The same result can be observed in the assessor-based aspectual recall levels of the two subgroups of topics, although not at a statistically significant level (see Table 10). Note that the break-up of topics approximately corresponds to average familiarity with each topic, as determined by post-search questionnaire. No equivalent effect was discernible in per-topic efficiency results.

Additionally, although for both list and cluster organizations subjects saved similar numbers of aspects on average (μ of 33.4 for the list, 31.3 for the cluster), subjects' behaviour when using the cluster interface was far more consistent than with the list interface (σ of 19.1 for the list, 8.2 for the cluster). Apparently, when using the cluster interface, the subjects saved on average the same number of aspects for each query, regardless of query familiarity or the number of aspects to be found.

Searchers' comments

From the exit questionnaire, 12 of the 16 subjects preferred the clustered organization to the list organization, and 13 subjects rated the clustered organization as easy to use.

A fairly clear preference for the cluster organization was shown by the subjects, who made comments such as:

> Clustering interface is easier, because it's grouped.
>
> Everything was nicely organized into group and I could skip some stuff and get directly to the point.
>
> It showed me all the list of the topic in a screen.

In contrast, comments on the simple list organization included:

> Too many links in one list.
>
> Everything was just in a list and it was difficult to concentrate on the actual topic.
>
> Long list, sometimes frustrated in couldn't find suitable topic.
>
> Hard to search, depends on the topic.
>
> Very simple interface - no confusion.

However, subjects did note some inadequacies of the cluster organization as implemented, such as:

> The keywords in each group are not clear. They will make users confused for the first time.

—the cluster descriptor terms were stemmed, rather than complete words—and

Topic	Number of docs retrieved			
	5	10	15	20
352	0.036	0.107	0.107	0.143
353	0.091	0.182	0.364	0.364
357	0.308	0.308	0.385	0.385
362	0.167	0.167	0.167	0.167
365	0.750	0.792	0.792	0.792
366	0.286	0.286	0.571	0.714
387	0.111	0.111	0.444	0.444
392	0.111	0.417	0.472	0.472
Avg.	0.233	0.296	0.413	0.435

Table 8: *Aspectual recall (as judged relevant by assessors) for documents from clusters from which a document was saved, at increasing numbers of documents viewed.*

	352	353	365	366	Avg.
List organization	23	21	22	12	19.5
Cluster organization	30	26	32	21	27.3

(a) "Hard" Topics

	357	362	387	392	Avg.
List organization	43	49	27	70	47.3
Cluster organization	35	33	25	48	35.3

(b) "Easy" Topics

Table 9: *Number of aspects saved, per topic.*

The group is not exactly you want.

—presumably indicating either the failure of the clustering algorithm, or the shortcomings of the cluster ranking algorithm.

3.5 Analysis

Although most subjects liked the clustered organization, we did not find any significant difference overall in either effectiveness or efficiency between the ranked list organization and the clustered organization, However, a statistically significant difference was observed in the number of aspects saved when the set of topics was divided into harder and easier groups; this result carried over to aspectual recall for the two groups, albeit not at a statistically significant level. This suggests that the cluster organization may be helpful for "hard" topics, but of less value for "easy" topics; confirming the general validity of this result and characterizing applicable situations must be the subject of further work.

The cluster hypothesis—that relevant documents tend to cluster—has been verified here; however, the algorithm used was not able to cluster documents into topic aspects. Subjects tended to browse all clusters, but generally saved documents from clusters that contained many aspects (as determined by the NIST assessors); in contrast, subjects generally did not save documents from clusters that contained few topic aspects. This suggests that, while clustering helped subjects identify useful groups of documents, without further aid subjects experienced some difficulty in identifying relevant documents. This is borne out by an improvement over the baseline in aspectual recall when considering documents from relevant (suggested by a subject having saved a document) clusters only. Secondary clustering passes, perhaps in the style of

	352	353	365	366	Avg.
List organization	.053	.068	.667	.071	.215
Cluster organization	.089	.102	.687	.214	.273

(a) "Hard" Topics

	357	362	387	392	Avg.
List organization	.279	.135	.250	.285	.237
Cluster organization	.231	.146	.111	.184	.168

(b) "Easy" Topics

Table 10: *Aspectual recall per topic of saved documents, as judged relevant by assessors.*

Scatter/Gather [6], once users have located clusters of interesting documents may aid the identification of specific relevant documents or distinct aspects.

4 Spoken document retrieval

For TREC7 we again chose to explore phoneme-based methods for spoken document retrieval (SDR). We believe that the phonetic approach is required for SDR with data sets containing more than one dialect of English. In our case, we are interested in approaches that can manage American, Australian, and British variations of English. MDS, in collaboration with CSIRO, participated in the full SDR run, which included two speech runs, *mds-s1* and *mds-s2*. The first speech run was submitted by the CSIRO team while the other retrieval runs were submitted by the RMIT team.

The two key processes involved in SDR are speech recognition followed by textual retrieval. The recognition process used the HTK toolkit [18]. The documents were recognised as phoneme sequences. For the reference and baseline retrieval runs, the word-based documents and queries were translated to phonemes using the CMU pronunciation dictionary [2]. The document collection contained 100 hours of News Broadcast obtained from LDC. It contained 2866 documents with an average length of 275 words. A set of 23 queries was used for evaluation. The average length of a query was about 16 words.

4.1 Speech recognition system

Based on our decision to use phone models as the basic units for recognition as well as text retrieval, 39 phones in the CMU dictionary were used for training the continuous density, left-to-right HMMs. As manually segmented and labelled American-accented speech data was not available for training those models, five files were arbitrarily selected from each different news program on the TREC-6 spoken document collection and were then partitioned into smaller files. Some of the partitioned files included noise, music and non-speech. These were filtered out. Finally we obtained trainable speech data of 18.4 hours and converted the corresponding word sequence of each speech file into sequences of phones, using the CMU dictionary and their grapheme-to-phoneme software, which were then used for training the models.

The acoustic parameters used were 12 mel-frequency cepstral coefficients, 12 delta coefficients and two normalised log-energy values and were extracted every 10 ms using a window frame of 25 ms (Hamming windows with pre-emphasis). With these acoustic feature vectors, initially 39 context-independent phone models with one mixture were trained and then 1521

mds-r1	Retrieval using reference transcriptions
mds-b1	Retrieval using baseline 1 transcriptions (35% wer)
mds-b2	Retrieval using baseline 2 transcriptions (50% wer)
mds-s2	Retrieval using phoneme-based transcriptions from our own team's phoneme-based recogniser.

Table 11: *Submitted runs for speech retrieval.*

words	Find reports of fatal air crashes
phonemes	F y N D R I P X R T S h V F x T h L E R K R a s I Z
quad-grams	FyND yNDR ... TShV ShVF ... ThLE ... ERKR ... asIZ
quad-grams, bounded	FyND RIPX ... XRTS hV FxTh xThL ER KRas ... asIZ

Table 12: *Example of differences between an unbounded quad-gram and a bounded quad-gram query.*

Total Number of files used for testing	:	960
% Correct	:	53.43 %
% Accuracy	:	43.34 %

Table 13: *Results of phone recognition.*

	Quad-gram queries
mds-r1	0.3107
mds-b1	0.2753
mds-b2	0.1937
mds-s2	0.1063

Table 14: *Average precision of submitted runs.*

right-context dependent models were trained incrementally with the number of mixtures from one to three, depending on the amount of training data. We chose the right-context models because our informal experiments showed that these models produced slightly better results than the left-context models, and triphone models were not trained due to insufficient training data. The right-context models that had less than 100 training speech tokens were cloned from the context-independent models.

For language modelling, the backoff bigram for the right-context dependent phones was computed from the label files used for training described above. Table 13 shows the results of phone recognition, which was over part of the training data consisting of 11,520 files.

Text retrieval was performed on the speech database of TREC-7 and details are in the subsequent sections.

4.2 Spoken document retrieval experiments from RMIT

Four runs were submitted by RMIT, shown in Table 11. Our phoneme-recognised documents, which had an error rate of about 50%, can be said to be highly corrupted with respect to the other types of transcriptions. Tri-grams and quad-grams of the phonetic transcriptions of the documents were formed and combined prior to indexing by our retrieval system. Queries were also translated to phonetic quad-grams such that no quad-grams were created across word boundaries.

Previous experiments using the TREC-6 spoken document collection, as well as subword unit experiments by Ng et al. [11], indicated that both tri-grams and quad-grams performed well on their own. The term weights of the combined tri-gram and quad-gram phonetic transcriptions would result in a retrieved rank set of documents which would be different to that obtained if they were not combined or by combination of indices [7]. Bounded quad-gram queries, where quad-grams were not created cross word boundaries, caused the formation of quad-grams as well as other shorter n-grams. An example of the differences between a bounded quad-gram and an unbounded one is shown in Table 12.

MG [15], developed at RMIT and University of Melbourne, was the retrieval engine used. The version used included the Okapi similarity formulation (equation 1). The occurrence of negative weights if $f_t \geq \frac{N}{2}$ is handled such that f_t is set to $\frac{N}{2} - 1$.

Four runs based on the different versions of the document collections were submitted. Three of the transcriptions, reference, baseline 1, and baseline 2, were word-based, while our speech recognised run was phoneme-based. The word-based documents were translated to phoneme sequences using the CMU pronunciation dictionary [2]. The word-based queries were translated as well. The original pronunciation dictionary contained approximately 118,000 words. For the training collection, about 1350 words were added and a further 2580 words were added to translate the test collection into phoneme sequences. Prior to translation, the documents were neither stopped nor stemmed.

For the submitted runs, tri-grams and quad-grams of the phonetic transcriptions were created. These were combined for each document prior to indexing. The Okapi similarity function described in the previous section was used for retrieval. Table 14 showed the average precision values for the submitted runs. Using the same retrieval system, the average precision values were also obtained for word-based documents and queries as well as stopped queries. Stopped queries were created using a stopped list of 368 words. The average length of the original queries was about 16 words while the average length of a stopped query was down to about 9 words. The results are shown in Table 15. Further experiments showed that optimal average precision values for the combination of tri-gram and quad-gram phonetic transcriptions can be obtained using a combination of the tri-gram and quad-gram query set. The average precision of these is shown in Table 16.

We translated all word-based documents to phoneme sequences and combined the tri-grams and quad-grams of these sequences for each document prior to retrieval. The queries used were the bounded quad-grams of the translated query where they were not created across word boundaries.

The results indicate that phoneme-based retrieval using n-grams is feasible. Compared to retrieval using word-based

	Original Queries	Stopped Queries
Reference	0.4464	0.4542
Baseline 1	0.4049	0.4171
Baseline 2	0.3403	0.3259

Table 15: *Average precision of word-based baseline retrieval experiments.*

	Original Queries	Bounded Queries
Reference	0.3411	0.3290
Baseline 1	0.3071	0.2886
Baseline 2	0.2314	0.2046
Speech	0.0818	0.1050

Table 16: *Average precision of combined tri-gram and quad-gram queries on combined tri-gram and quad-gram documents.*

documents, as shown in Table 15, phonetic n-gram retrieval did not perform well. This was because there was significant loss of contextual information due to the lost of boundary information after the word documents were translated to phoneme sequences. The use of fixed size combinations of tri-gram and quad-gram for the documents and queries also increased the noise of the collection. There was no significant improvement in retrieval performance using stopped queries in the word-based case, and, although results were not shown here, the same was observed for stopped queries translated to n-grams and used for retrieval.

Using a combination of tri-gram and quad-gram, we found that bounded queries did not improve retrieval performance of the phonetic transcriptions of word-based documents, but there was a slight improvement for phoneme-based documents. This meant that documents which were recognised using a word-based recogniser performed better using the phoneme queries where n-grams were created across boundaries; and documents recognised using a phoneme-based recogniser performed better using bounded queries. A typical result comparing bounded and unbounded queries is shown in Table 16.

The combination of tri-grams and quad-grams showed a slight improvement compared to its performance individually, as shown in Table 17. This was due to increases in numbers of relevant documents found using quad-grams although there was also an increase in noisy matches due to tri-grams.

4.3 Spoken document retrieval experiments from CSIRO

CSIRO submitted a single run, *mds-s1*. The retrieval system developed for this experiment was composed of Perl scripts and C programs developed by CSIRO. The mds-s1 run used the recognition stream developed for mds-s2 but applied a different retrieval technique.

Regardless of what speech recognition system or approach is utilised during the recognition phase, errors will occur, resulting in the creation of a recognition stream with an incorrect representation of the voice data. In the mds-sr1 run, we tested the use of approximate string matching to compensate for these index errors and thus improve retrieval performance. Our hypothesis is that, by supplementing establish speech recognition techniques with approximate string matching techniques, improved retrieval performance will result. As discussed above, the phoneme-based recognizer produced significant levels of errors in the recognition stream (53% accuracy). Because of these inaccuracies an exact string matching approach could not identify any occurrences of a sample set of search terms in the spoken document set's recognition stream. During the HMM (HMM) training phase, HTK produces a table containing performance information, known as the confusion matrix. This table contains detailed information about the types of recognition errors produced by the recognizer (phone substitution, insertion or deletion). By utilizing each phone's performance profile, the approximate string-matching task can eliminate unlikely matches, thus improving matching performance and accuracy.

After the recognition process had been completed, the retrieval phase started with a series of perl scripts used to modify and convert both the query and recognition stream's formats. Upon completion of the conversion process, the retrieval system combined the use of approximate string matching with the confusion matrix to identify occurrences of each search term within the recognition stream.

Several steps were used to convert the queries supplied by TREC into a format acceptable to the mds-s1 retrieval system. Initially, SGML tags and punctuation were removed from the query sentences. Non-noun words were removed from the query using the Moby Part-of-Speech Dictionary [14]. The Carnegie Mellon Pronouncing Dictionary [2] was then used to convert the remaining query terms into their phonetic equivalence. The phonetic label set used by the Carnegie Mellon Pronouncing Dictionary varied from the label set used by the mds-s1 retrieval system. For this reason a conversion script was applied to the query set to translate from the Carnegie Mellon based phonetic query terms to a representation appropriate for the mds-s1 matching algorithm. We referred to the query terms produced through this process as a "processed query term".

The file produced by the recogniser contains a continuous stream of labels representing the phonetic content of the voice track. A sliding window was used to move over the continuous phonetic string one label at a time to parse and generate a set of fixed length sub-strings. The size of the sliding window used was the number of characters in the processed query term. Agrep [17] was then employed as a fast filtering tool to quickly eliminate all sub-strings within the phonetic stream that could not contain the phonetic representation of the search term. Since agrep does not provide insight into which edit operations are utilized during its approximate string matching operation, it was necessary to develop a module that would include in its output the number, order, and type of edit operations required to establish a match. The output of this module includes information about which symbols were deleted, which symbols were inserted and which symbols were replaced by another symbol.

The matching algorithm can derive multiple matches for a single recognition sub-string because the same match can be produced using different sets of operations. Therefore, the output of the matching algorithm typically contains numerous occurrences of the same match, with different combinations of insertion, deletion and substitution operations. It was proposed that the greater the match redundancy for a recognition stream sub-string, the more likely the match is correct. An algorithm was devised that takes into account a match's redundancy, a threshold factor, and the number of nouns within the query that were matched for the spoken document, yielding a ranking for each query and each spoken document.

The experimental results are disappointing. Overall average precision for the mds-s1 run was 0.0223. Due to limited time and resources, the reference and baseline runs were not

	Tri-gram		Quad-gram	
	Original Queries	Bounded Queries	Original Queries	Bounded Queries
Reference	0.3065	0.3033	0.3230	0.3005
Baseline 1	0.2599	0.2757	0.2936	0.2531
Baseline 2	0.2051	0.2015	0.2265	0.1871
Speech	0.0421	0.0561	0.1013	0.0978

Table 17: *Average precision of n-gram transcripts using n-gram queries.*

conducted. Preliminary analysis of the results shows that, while recall over the total document set was reasonable (of the 390 relevant documents 268 were retrieved – 68.7%), precision was woefully inadequate. It was recognized at the submission time that the ranking algorithm was deficient.

Acknowledgements

The work reported in this paper has been partially funded by the Cooperative Research Centres Program through the Department of the Prime Minister and Cabinet of Australia, and by the Australian Research Council.

References

[1] C. Buckley, G. Salton, and J. Allan. The effect of adding relevance information in a relevance feedback environment. In W. Croft and C. van Rijsbergen, editors, *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 292–300, July 1994.

[2] Cmudict.0.4: Carnegie Mellon University Pronouncing Dictionary. Available from: `http://www.speech.cs.cmu.edu/cgi-bin/cmudict/`, 1995.

[3] J. Fagan. *Experiments in Automatic Phrase Indexing for Docuement Retrievel: A comparison of syntactic and nonsyntactic methods.* PhD thesis, Cornel University, September 1987.

[4] W. B. Frakes and R. Baeza-Yates, editors. *Information Retrieval: Data Structures and Algorithms.* Prentice-Hall, Inc., Englewood Cliffs, NJ, U.S.A., 1992.

[5] M. Fuller, M. Kaszkiel, C. L. Ng, P. Vines, R. Wilkinson, and J. Zobel. MDS TREC6 report. In E. M. Vorhees and D. K. Harman, editors, *Proceedings of the Sixth Text Retrieval Conference*, Gaithersburg, MD, U.S.A., 1997. Department of Commerce, National Institute of Standards and Technology.

[6] M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In H.-P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, editors, *Proceedings of the 19th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 76–84, Zurich, Switzerland, 18-22 August 1996. ACM.

[7] G. J. F. Jones, J. T. Foote, K. S. Jones, and S. J. Young. Retrieving spoken documents by combining multiple index sources. In *Proceedings of the 19th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 30 – 38, 1996.

[8] M. Kaszkiel, P. Vines, R. Wilkinson, and J. Zobel. The MDS Experiments for TREC5. In D. Harman, editor, *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, pages 209–216, November 1996.

[9] M. Kaszkiel and J. Zobel. Passage retrieval revisited. In N. J. Belkin, D. Narasimhalu, and P. Willett, editors, *Proceedings of the 20th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 178–185, July 1997.

[10] J. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computation*, 11(1-2):22–31, 1968.

[11] K. Ng and V. W. Zue. Subword unit representations for spoken document retrieval. In *Proc. ESCA Eurospeech Conference*, pages 1607 – 1610, Rhodes, Greece, 1997.

[12] G. Salton, editor. *The Smart retrieval system: experiments in automatic document processing.* Prentice-Hall, 1971.

[13] S. Walker, S. Robertson, M. Boughanem, G. Jones, and K. S. Jones. Okapi at TREC-6 Automatic ad hoc, VLC, routing, filtering and QSDR. In E. Voorhees and D. Harman, editors, *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*, pages 125–136, November 1997.

[14] G. Ward. Moby part-of-speech dictionary. `http://www.dcs.shef.ac.uk/research/ilash/Moby/`, 1996.

[15] I. Witten, A. Moffat, and T. Bell. *Managing Gigabytes: Compressing and indexing documents and images.* Van Nostrand Reinhold, 1994.

[16] M. Wu and R. Wilkinson. Evaluation of indexing methods for clustering. In J. Kay and M. Milosavljevic, editors, *Proceedings of the Third Australian Document Computing Symposium*, pages 14–19, August 1998. Proceedings available as Technical Report 518, Basser Department of Computer Science, University of Sydney.

[17] S. Wu and U. Manber. Fast text searching with errors. Technical report, Department of Computer Science, University of Arizona, 1991. Technical Report TR-91-11.

[18] S. Young, J. Jansen, J. Odell, D. Ollason, and P. Woodland. *The HTK Book.* Entropic Cambridge Research Laboratory, 1995.

Ad Hoc Retrieval Experiments Using WordNet and Automatically Constructed Thesauri

Rila Mandala, Takenobu Tokunaga, Hozumi Tanaka,
Akitoshi Okumura and Kenji Satoh

NEC Corp. and Tokyo Institute of Technology
Japan

Abstract

This paper describe our method in automatic-adhoc task of TREC-7. We propose a method to improve the performance of information retrieval system by expanded the query using 3 diffferent types of thesaurus. The expansion terms are taken from hand-crafted thesaurus (WordNet), co-occurrence-based automatically constructed thesaurus, and syntactically predicate-argument based automatically constructed thesaurus.

1 Introduction

A critical problem in information retrieval is that the vocabulary that the searchers use is not the same as the one by which the documents have been indexed. The word synonymy is one example of this problem. If a user use a synonym of a word which document has been indexed in his/her query, then that documents could not be retrieved.

Query expansion [3] is one method in information retrieval to avoid this problem. The expansion terms can be taken from thesaurus [4, 9, 11]. There are many research of query expansion using thesaurus in the literature. Briefly there are two types of thesaurus, i.e. hand-crafted thesaurus and automatically constructed thesaurus. WordNet [10] is one example of hand-crafted thesaurus which is publically available in machine readable form.

Corpus-based thesaurus is a thesaurus which is constructed automatically from the corpus without intervention of human. There are two different method to extract thesaural relationships from corpora predicate-argument (also called head-modifier) method [6, 5, 8, 7, 13] and co-occurrence statistical method [1, 2, 12, 15]

We propose the use of WordNet, co-occurrence-based and predicate-argument-based automatically constructed thesauri for query expansion in automatic-adhoc task of TREC-7.

2 Method

2.1 Co-occurrence-based Thesaurus

The general idea underlying the use of term co-occurrence data for thesaurus construction is that words that tend to occur together in documents are likely to have similar, or related, meanings.

Co-occurrence data thus provides a statistical method for automatically identifying semantic relationships that are normally contained in a hand-made thesaurus. Suppose two words (A and B) occur f_a and f_b times, respectively, and cooccur f_c times, then the similarity between A and B can be calculated using a similarity coefficient such as the Dice Coefficient

$$\frac{2 \times f_c}{f_a + f_b}$$

2.2 Predicate-Argument-based Thesaurus

In contrast with the previous section, this method attempts to construct a thesaurus according to predicate-argument structures. The use of this method for thesaurus construction is based on the idea that there are restrictions on what words can appear in certain environments, and in particular, what words can be arguments of a certain predicate [7]. For example, a *cat* may *walk*, *bite*, but can not *fly*. Each noun may therefore be characterized according to the verbs or adjectives that it occurs with. Nouns may then be grouped according to the extent to which they appear in similar constructions.

First, all the documents are parsed using the Apple Pie Parser, which is a bottom-up probabilistic chart parser developed by Satoshi Sekine [16]. Its grammar is a semi-context sensitive grammar and it was automatically extracted from Penn Tree Bank syntactically tagged corpus made at the University of Pennsylvania. Its performance is 0.71 of precision, 0.70 of recall and 3.03 of average crossing.

Using this parser, the following syntactic structures are extracted

- Subject-Verb
- Verb-Object
- Adjective-Noun

Each noun has a set of verbs and adjective that it occurs with, and for each such relationship, a dice coefficient value is calculated.

- $C_{sub}(v_i, n_j) = \frac{2 \times f_{sub}(v_i, n_j)}{f(v_i) + f_{sub}(n_j)}$,
  where $f_{sub}(v_i, n_j)$ is the frequency of noun n_j occurring as the subject of verb v_i, $f_{sub}(n_j)$ is the frequency of the noun n_j occurring as subject of any verb, and $f(v_i)$ is the frequency of the verb v_i
- $C_{obj}(v_i, n_j) = \frac{2 \times f_{obj}(v_i, n_j)}{f(v_i) + f_{obj}(n_j)}$,
  where $f_{obj}(v_i, n_j)$ is the frequency of noun n_j occurring as the object of verb v_i, $f_{obj}(n_j)$ is the frequency of the noun n_j occurring as object of any verb, and $f(v_i)$ is the frequency of the verb v_i
- $C_{adj}(a_i, n_j) = \frac{2 \times f_{adj}(a_i, n_j)}{f(a_i) + f_{adj}(n_j)}$,
  where $f(a_i, n_j)$ is the frequency of noun n_j occurring as argument of adjective a_i, $f_{adj}(n_j)$ is the frequency of the noun n_j occurring as argument of any adjective, and $f(a_i)$ is the frequency of the adjective a_i

We define the similarity of two nouns with respect to one predicate, as the minimum of each dice coefficient with respect to that predicate, i.e.
$SIM_{sub}(v_i, n_j, n_k) = min\{C_{sub}(v_i, n_j), C_{sub}(v_i, n_k)\}$

$SIM_{obj}(v_i, n_j, n_k)$=$min\{C_{obj}(v_i, n_j), C_{obj}(v_i, n_k)\}$
$SIM_{adj}(a_i, n_j, n_k)$=$min\{C_{adj}(a_i, n_j), C_{adj}(a_i, n_k)\}$

Finally the overall similarity between two nouns is defined as the average of all the similarities between those two nouns for all predicate-argument structures.

2.3 Expansion Term Weighting Method

A query q is represented by a vector $\vec{q} = (q_1, q_2, ..., q_n)$, where the q_i's are the weights of the search terms t_i contained in query q.

The similarity between a query q and a term t_j can be defined as belows [12] :

$$simqt(q, t_j) = \sum_{t_i \in q} q_i * sim(t_i, t_j)$$

Where the value of $sim(t_i, t_j)$ can be defined as the average of the similarity values in the three types of thesaurus. Since in WordNet there are no similarity weights, when there is a relation between two terms in WordNet, their similarity is taken from the average of the similarity between those two terms in the co-occurrence-based and in predicate-argument-based thesauri.

With respect to the query q, all the terms in the collection can now be ranked according to their *simqt*. Expansion terms are terms t_j with high $simqt(q, t_j)$.

The $weight(q, t_j)$ of an expansion term t_j is defined as a function of $simqt(q, t_j)$

$$weight(q, t_j) = \frac{simqt(q, t_j)}{\sum_{t_i \in q} q_i}$$

where $0 \leq weight(q, t_j) \leq 1$.

An expansion term gets a weight of 1 if its similarity to all the terms in the query is 1. Expansion terms with similarity 0 to all the terms in the query get a weight of 0. The weight of an expansion term depends both on the entire retrieval query and on the similarity between the terms in the thesauri.

The query q is expanded by adding the following query

$$\vec{q_e} = (a_1, a_2, ..., a_r)$$

where a_j is equal to $weight(q, t_j)$ if t_j belongs to the top z ranked terms. Otherwise a_j is equal to 0.

The resulting expanded query is

$$\vec{q}_{expanded} = \vec{q} \circ \vec{q_e}$$

where the $\circ$ is defined as the concatenation operator.

The method above can accommodate the polysemous word problem, because an expansion term which is taken from a different sense to the original query term is given very low weight.

3 Experiments

As a retrieval engine we used SMART [14] version 11.0. SMART is an information retrieval system based on the vector space model in which term weights are calculated based on term frequency,

inverse document frequency, and document length normalization. We used *lnc* for document's term weighting and *ltc* for query's term weighting.

We ran experiments in the automatic-adhoc task framework using only title, only description, and all terms of the topics. The results are shown belows

	Title	Description	All
	=====	==========	===
Total number of documents over all queries			
Retrieved	50000	50000	50000
Relevant	4674	4674	4674
Rel_ret	2435	3149	3106
Interpolated Recall - Precision Averages			
at 0.00	0.6957	0.7782	0.8161
at 0.10	0.4528	0.5643	0.5783
at 0.20	0.3622	0.4377	0.4511
at 0.30	0.2864	0.3519	0.3575
at 0.40	0.2148	0.2981	0.2899
at 0.50	0.1438	0.2300	0.2177
at 0.60	0.1017	0.1786	0.1618
at 0.70	0.0530	0.1212	0.1121
at 0.80	0.0321	0.0749	0.0636
at 0.90	0.0049	0.0159	0.0306
at 1.00	0.0005	0.0067	0.0054
Average precision (non-interpolated) over all rel docs			
	0.1898	0.2584	0.2565
Precision			
At 5 docs	0.4720	0.5840	0.5800
At 10 docs	0.4260	0.5460	0.5480
At 15 docs	0.4080	0.5013	0.4973
At 20 docs	0.3700	0.4640	0.4700
At 30 docs	0.3320	0.4113	0.4147
At 100 docs	0.2012	0.2406	0.2484
At 200 docs	0.1395	0.1771	0.1771
At 500 docs	0.0791	0.1038	0.1023
At 1000 docs	0.0487	0.0630	0.0621
R-Precision (precision after R (= num_rel for a query) docs retrieved)			
Exact	0.2403	0.2993	0.2989

Figure 1 shows the 11 point interpolated precision graph for the retrieval result using title only, description only, and title+description+narrative.

4 Discussion of Result

As expected, the performance of retrieval using only title of topic yields a worst performance. The use of only description of topic has a higher retrieval performance than the use of all sections of topic. This can be explained that the narrative section of topic has some negation statements

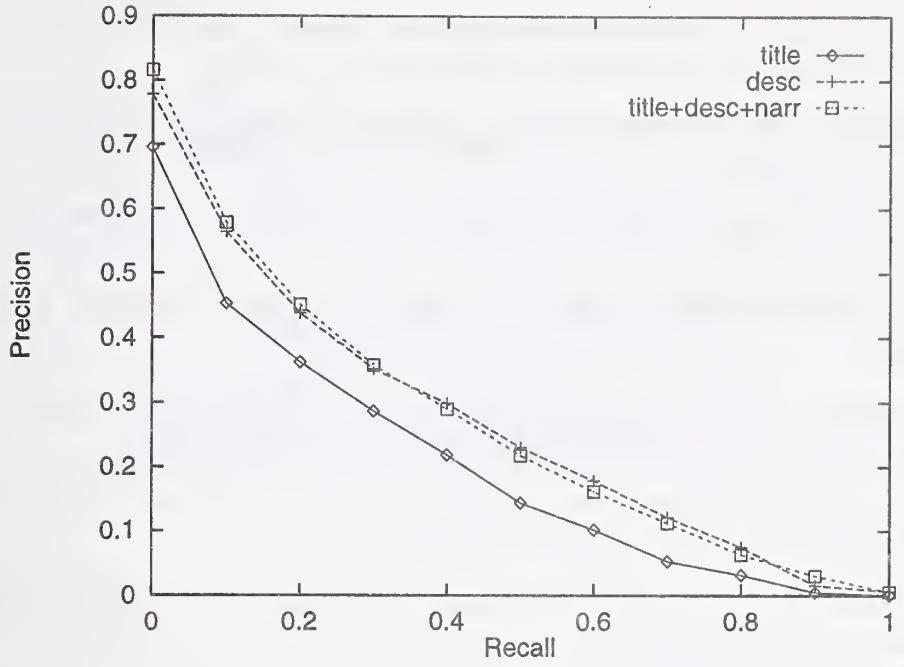


Figure 1: 11-point interpolated precision using title, description, and title+description+narrative

which could not be handled properly by our system yet. Expanding terms occur in the negation statement degraded the performance very much.

5 Conclusion

We have implemented and experimented a method for query expansion using WordNet and corpus-based thesauri. To avoid the wrong expansion terms, a weighting method is utilized whereby the weight of expansion terms depends on the similarity value of those terms in the various thesauri and on the weight of all terms in original query.

In the future, we will investigate the proper method to handle term expansion in the negation statement.

6 Acknowledgements

The authors would like to thank Mr. Timothy Baldwin (TIT, Japan) for his comments on the earlier version of this paper. We also grateful to Dr. Chris Buckley (SabIR Research) and Dr. Satoshi Sekine (New York University) for providing the SMART information retrieval engine and Apple Pie Parser respectively.

References

[1] Carolyn J. Crouch. An aproach to the automatic construction of global thesauri. *Information Processing and Management*, Vol. 26, No. 5, pp. 629–640, 1990.

[2] Yang B. Crouch, C.J. Experiments in automatic statistical thesaurus construction. In *Proceeding of the Fifteenth ACM SIGIR Conference*, pp. 77–82, 1992.

[3] Ekmekcioglu F.C. Effectiveness of query expansion in ranked-output document retrieval systems. *Journal of Information Science*, Vol. 18, pp. 139–147, 1992.

[4] Edward A. Fox. Lexical relations enhancing effectiveness of information retrieval systems. *SIGIR Forum*, Vol. 15, No. 3, pp. 6–36, 1980.

[5] Gregory. Grafenstette. *Explorations in Automatic Thesaurus Discovery.* Kluwer Academic Publ., 1994.

[6] G. Grefenstette. Use of syntactic context to produce term association lists for text retrieval. In *Proceedings of the 15th ACM SIGIR Conference*, pp. 89–97, 1992.

[7] D. Hindle. Noun classification from predicate-argument structures. In *Proceedings of 28th Annual Meeting of the ACL*, pp. 268–275, 1990.

[8] Y. Jing and B. Croft. An association thesaurus for information retrieval. In *Proceedings of RIAO*, pp. 146–160, 1994.

[9] Joana Kristensen. Expanding end-users query statements for free text searching with a search-aid thesaurus. *Information Processing and Management*, Vol. 29, No. 6, pp. 733–744, 1993.

[10] G.A Miller. Special issue, wordnet an on-line lexical database. *International Journal of Lexicography*, Vol. 3, No. 4, 1990.

[11] Chris D. Paice. A thesaural model of information retrieval. *Information Processing and Management*, Vol. 27, No. 5, pp. 433–447, 1991.

[12] Qiu and H.P. Frei. Concept based query expansion. In *Proceedings of the 16th ACM SIGIR Conference*, pp. 160–169, 1993.

[13] G Ruge. Experiments on linguistically-based term associations. *Information Processing and Management*, Vol. 28, No. 3, pp. 317–332, 1992.

[14] G. Salton. *The SMART Retrieval System Experiments in Automatic Document Processing.* Prentice-Hall, 1971.

[15] H. Schutze and J.O. Pederson. A cooccurrence-based thesaurus and two applications to information retrieval. In *Proceedings of RIAO Conference*, pp. 266–274, 1994.

[16] S. Sekine and R. Grishman. A corpus-based probabilistic grammar with only two non-terminals. In *Proceedings of the International Workshop on Parsing Technologies*, 1995.

NTT DATA at TREC-7: system approach for ad-hoc and filtering

Hiroyuki Nakajima, Toru Takaki, Tsutomu Hirao and Akira Kitauchi
Laboratory for Information Technology
NTT DATA Corporation
Kowa Kawasaki Nishi-guchi Bldg., 66-2 Horikawa-cho,
Saiwai-ku, Kawasaki-shi, Kanagawa 210-0913 Japan
{nakajima,takaki,hirao,kitauchi}@lit.rd.nttdata.co.jp

1 Introduction

In TREC-7, we participated in the ad-hoc task (main task) and the filtering track (sub task). In the ad-hoc task, we adopted a scoring method that used co-occurrence term relations in a document and specific processing in order to determine which conceptual parts of the documents should be targeted for query expansion. In filtering, we adopted a machine-readable dictionary for detecting idioms and an inductive learning algorithm for detecting important co-occurrences of terms. In this paper, we describe the system approach and discuss the evaluation results in brief for our ad-hoc and filtering in TREC-7.

2 Ad-hoc Track

This section describes the method we adopted that allowed by the ad-hoc task to obtain the output results.

2.1 System description

Figure 1 shows the processing procedure in our system. The structure of the databases and the action of each processing module are explained as follows.

(a) Databases

As for each data set of the *Financial Times*-1991-1994 (FT), *Federal Register*-1994 (FR94), *Foreign Broadcast Information Service* (FBIS) and the *LA Times* (LATIMES) retrieved by TREC-7 ad-hoc, the index is made respectively.

(b) Processing module

(1) Query term extraction module

First, a term that will be used for retrieval and scoring is extracted from the input topics, and a list of retrieval terms is made. The stopwords (550 words) are then deleted, and the extracted terms are converted into root words.

(2) Query term limitation module

To do the score calculation processing efficiently, the terms are limited from the term list that was obtained by the extraction module. In this term limitation processing, a term's degree of importance is defined by the *idf* (Inverse Document Frequency) value, and terms which have a low degree of importance are deleted from the query term list before the retrieval is processed. The *idf* value is

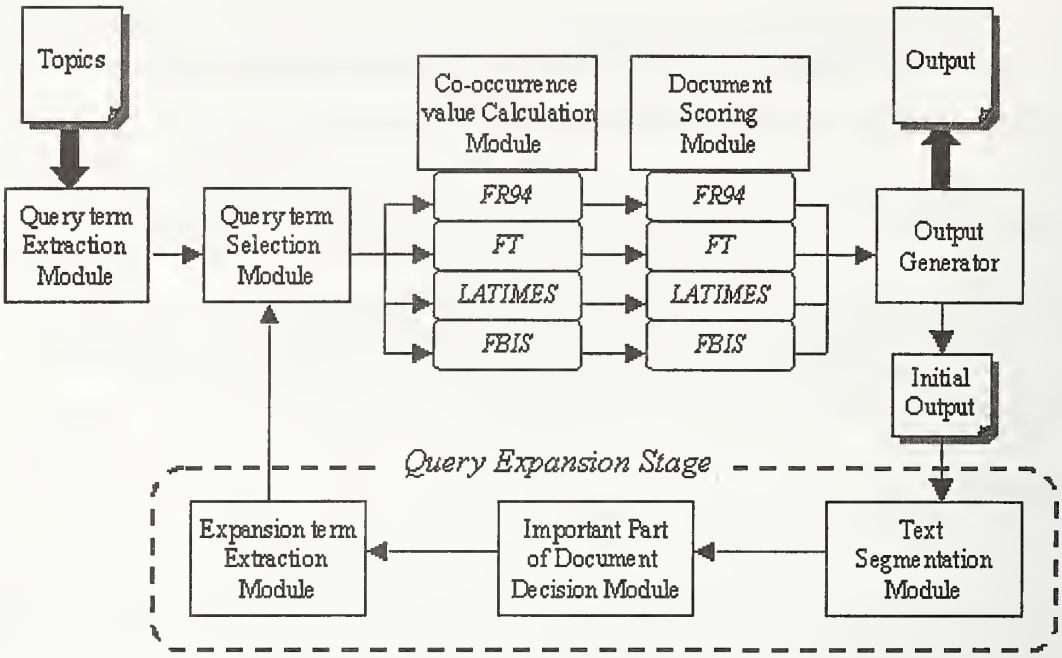


Figure 1: Flow of the processing procedure(ad-hoc task)

calculated from the corpus that consists of all data sets used for TREC-7 ad-hoc. Terms that have low degrees of importance are disregarded because it is assumed that they do not have a big influence on the score of each document for topics. In addition, the degree of importance degree of each division, title, description and narrative, in the input topics is determined, and these values were used to set the degree of importance of the query term and in the scoring. When the degree of importance of a term which appeared in the title was assumed to be 1.0, this system adjusted the degree of importance of description and narrative to 0.1.

(3) Co-occurrence value calculation module

The terms used for retrieval and scoring are extracted from the input topics. Processing is done to add to the score of the document in consideration of the importance of co-occurrence, when two related terms appear in the same document. In this module, the degree of importance of each co-occurrence in a document is calculated. Details of the processing method are described later.

(4) Document scoring module

We used a scoring method based on $tf - idf$ as a basis of the score calculation. Here, the degrees of importance of the co-occurrence terms calculated in the previous module are added, and a final document score is calculated. The score of each document is calculated using the document frequency of the data set to which the document belongs.

(5) Output generator module

The output generator module merges each result from a data set into one result. This system does not normalize the score between documents included in each data set. Each document is sorted in order of score, and the module generates a formatted output.

(6) Query term expansion module

To decrease retrieval leakage, we adopted the query term expansion. The basis of this query term expansion is a local feedback method, which involves the top ranked documents retrieved by the original query. Terms to be included in the expanded term are selected from previously the processed parts of the document that are considered to be important enough. We will discuss how to determine the level of importance later.

2.2 Degree of importance of co-occurrence appearance

We think that when query terms co-occur in the documents, it is indication that the document is more relevant to the query than documents in which more than once query terms appear. The degree of importance of the co-occurrence is defined according to this co-occurrence condition, and processing by which a co-occurrence important value adds to the score calculated by usual frequency of the term is executed further. The co-occurrence important values are decided according to the following parameters:

- The distance between adjacent terms: ρ (It is important if two terms appear near each other)
- The relative relationship between terms: σ (It is not important even if no related terms co-occurred)
- The importance of the co-occurred term: τ (Co-occurrence with a term that is not considered important is not crucial)

The degree of importance $cw(t_i, t_j)$ of the co-occurrence of terms t_i and t_j is defined by the next expression.

$$cw(t_i, t_j) = \rho(t_i, t_j) \times \sigma(t_i, t_j) \times \tau(t_i) \times \alpha \tag{1}$$

where,

$$\rho(t_i, t_j) = \begin{cases} \dfrac{\lambda - d_{ij}}{\lambda} & if : \lambda > d_{ij} \\ 0 & :\text{otherwise} \end{cases} \tag{2}$$

$$\sigma(t_i, t_j) = \frac{rtf_{ij}}{atf_i} \tag{3}$$

$$\tau(t_i) = \log(\frac{N}{df_j}) \tag{4}$$

Here,
λ is a parameter of the adjacent appearance distance,
d_{ij} is the distance between t_i and t_j (in number of words),
rtf_{ij} is frequency in the database of t_i that t_j appears with an adjacent appearance distance of λ words or less,
atf_i is the appearance frequency of t_i in the database,
N is the total number of the documents in the database,
df_j is the number of documents in which appears t_j

The degree of the co-occurrence importance of these words is calculated between two words.

2.3 Query term expansion

We applied the local feedback method as a query term expansion. This local feedback analyzes the document retrieved by the initial query and usually obtains the expansion terms. An extended term can be obtained from the entire document using this method but we assumed that its could be obtained from an important part of the document. An important part is determined from the segment which is the unit of a consecutive sentence. There is a method of expanding the query term from the sentence, from which the degree of importance of each sentence is calculated and the importance degree is high. However, sentences which do

not include the query term cannot be extracted by this method. Moreover, there is a method of deciding which parts of the document are important by dividing the document into chapters and paragraphs, etc. This method has a problem in that parts containing unrelated subjects may be used to expand the query term when multiple subjects are included in the unit of the document structure.

The degree of importance of each sentence is calculated, and the change in the degree of importance is used to determine the segment's range. In general, the degree of importance of the sentence in one document performs the change. The range of the segment is determined by assuming the part where the degree of importance of the sentence is low to be a gap in meaning and then dividing the document into the segments. The sum total of the degrees of importance of the sentences in each segment is assumed to be the degree of importance of the segment, and the segment with a high degree of importance becomes a query term extended object. The terms included in the selected segment become the candidate of extended terms. But all terms are not used for query expansion. We assumed the term which has higher value of *idf* to be an extended word.

2.4 Results

We submitted three processing results to NIST. The method of each processing of the submitted result is as follows. Table 1 shows the evaluation result of TREC-7 ad-hoc by each method.

[A] *nttdata7Al0* (judged)

The query term was generated from all fields of topics (title, description, and narrative). The method of the term frequency base ($tf - idf$) was used for scoring, and, in addition, co-occurrence information was applied to the score element. Here, we did not use feedback for the expansion of the query term.

[B] *nttdata7Al2* (judged)

This query expansion processing was executed after processing the *nttdata7Al0*. Twenty high-ranking documents of the initial retrieval result were targeted for local feedback. In addition, we executed the query term expansion by specifically processing an important part in the document. Thirty words were selected in order of the value of idf and these words were assumed to be extended query terms.

[C] *nttdata7At1* (submitted but NOT judged)

The processing method was similar to that of *nttdata7Al2*. Only the title filed of the topics was used for query term generation.

[X] *nttdata7Anorm* (NOT submitted: for comparison)

This method only scored by the tf-idf base by processing the *nttdata7Al0*. The degree of importance of the co-occurrence was not considered.

nttdata7Anorm vs *nttdata7Al0*

First, we compared the *nttdata7Anorm* and *nttdata7Al0* which applied the degree of importance of the co-occurrence. The average precision improved by 4.5% when the degree of the importance of the co-occurrence was used. Moreover, the relevant-retrieved number improved from 2580 to 2624. When the degree of importance of the co-occurrence was applied, a decrease in the precision was observed in the lower recall part (0.0,0.1,0.2) and the top document part.

RUN ID	[X] *nttdata7Anorm* (NOT submited)	[A] *nttdata7Al0* (judged)		[B] *nttdata7Al2* (judged)	
Relevant-Retrieved	2580	2624		2656	
Recall		(%Change vs [X])		(%Change vs [A])	
at 0.00	0.7384	0.6738	(-8.75%)	0.6715	(-0.34%)
at 0.10	0.4495	0.4366	(-2.87%)	0.4476	(+2.52%)
at 0.20	0.3592	0.3469	(-3.42%)	0.3523	(+1.56%)
at 0.30	0.2675	0.2816	(+5.27%)	0.2947	(+4.56%)
at 0.50	0.1567	0.1818	(+16.02%)	0.1934	(+6.38%)
at 0.70	0.0764	0.1007	(+31.81%)	0.1090	(+8.24%)
at 1.00	0.0005	0.0013	(+160.00%)	0.0003	(-76.92%)
Average	0.1943	0.2032	(+4.58%)	0.2113	(+3.99%)
At 5 docs	0.4800	0.4400	(-8.33%)	0.4280	(-2.73%)
At 10 docs	0.4340	0.4100	(-5.53%)	0.4080	(-0.49%)
At 20 docs	0.3490	0.3480	(-0.29%)	0.3973	(+6.03%)
At 30 docs	0.3053	0.3080	(+0.88%)	0.3200	(+5.46%)
At 100 docs	0.1932	0.1972	(+2.07%)	0.2050	(+3.90%)
At 500 docs	0.0832	0.0525	(+1.92%)	0.0862	(+1.38%)
R-Precision	0.2398	0.2493	(+3.96%)	0.2483	(-0.40%)

Table 1: TREC-7 ad-hoc result

nttdata7Al0 vs *nttdata7Al2*

Retrieval accuracy was improved when query term expansion was used. When *nttdata7Al2* was compared with *nttdata7Al0*, the average precision improved by 4%. In particular, an improvement in recall was observed in the middle recall range. The improvement of the recall of 30 top-ranking documents is high. In general, it is called 20 or 30 top-ranking documents of the retrieval result at most that the user reads, thus this method might be useful.

3 Filtering Track

Our filtering track system is based on Rocchio feedback[8] and dynamic feedback optimization (DFO)[1]. Rocchio feedback and DFO have shown fine results in past routing tasks in TREC. In recent years, several methods have been proposed that enhance the Rocchio feedback and DFO so that they are able to handle the weights of co-occurring pairs of terms, and they succeeded in improving the precision of results.

We think that co-occurring pairs of original query terms are especially important in relevance feedback, because:

1. We can use filtering profiles to show important terms to users. But users find it is very hard to imagine the relationships between terms. If we can show important co-occurrence pairs to users, they will find it easier to grasp the relationships between terms.

2. Once indices of terms are read into the memory of system, the system can check the co-occurrence of terms in memory, and this doesn't strongly affect processing time. In many cases, the terms in original queries are important, and therefore, they should be used in relevance feedback, and read into the memory. If the system can correctly find important pairs of query terms, these pairs improve the results in a short processing time.

We have constructed a system that is very similar to that reported by AT&T at TREC-6[9], but with additional features to handle co-occurrence pairs.

1. We added an idiom dictionary that is constructed using an English-Japanese dictionary to enable idiom indexing. We consider idioms to be a special case of term co-occurrence.

2. We added an inductive-learning algorithm to detect the co-occurrence of more than two terms.

3.1 System

We give a brief description below.

3.1.1 Building inverted files

Stopwords and stemming

We used the stemming algorithm of Porter's[6], and removed terms in stopword lists of freeWAIS[5] and SMART.

Idioms and phrases

In general, we construct inverted files of documents by using term-based indexing. However, some terms have special meanings that are quite different from the meaning of each term, i.e., their meaning is idiomatic. If we index idioms by their constituent terms, we lose the opportunity to use their special meanings.

The meanings of terms and idioms can often be clarified by paraphrasing them, or, translating them into another language. At TREC-7, we used an English-Japanese dictionary to first translate expressions into Japanese and then translate their constituent terms. If the meanings are quite different, the expression is identified as idiomatic, and it is added to our idiom dictionary.

This idiom dictionary is then used when we build inverted files from documents. The idiomatic expressions are marked, and these markers are used to build idiom indices. The constituent terms are not indexed.

After this idiom processing, any pair of adjacent words that are neither stopwords nor idioms is regarded as a phrase, and both term and phrase indices are built.

Term weighting

We used 'lnc' and 'ltc' schemes, as in SMART in TREC2[2].

3.1.2 Refining term weights through feedback

We used 'multi-pass' Rocchio feedback[9] and 2-pass DFO for refining term weights. We used 1000 terms and 100 phrases, and set the size ratio of the vectors of original query (α), relevant documents (β), and non-relevant documents (γ) to be 1:8:-8 in Rocchio feedback.

3.1.3 Detecting term pairs and weightings

Co-occurrence of 2 terms

After 1st-pass of Rocchio feedback, we detected co-occurrence pairs of 2 terms. We regarded any pairs of original query terms and top 100 terms (weighted by the Rocchio formula) as co-occurrence pairs, and calculated their weights in the same way that the term weights of Rocchio feedback are calculated. 'tf' and 'idf' of pairs are calculated as follows:

tf : the smaller value of tf of the two terms.
idf : calculate the number of documents (N_{pair}) which include the pair of the term#1 and term#2 as follows:

$$\begin{aligned} N_{pair} &= collection\_size \\ &\times (number\_of\_document\_including\_term\#1/collection\_size) \\ &\times (number\_of\_document\_including\_term\#2/collection\_size) \end{aligned} \quad (5)$$

We used 200 positively weighted pairs.

Detecting co-occurrence of more than 2 terms and phrases

Co-occurrence of 2 terms has reported to be effective in improving the precision of results. We assumed if we apply the same methods for the co-occurrence of more than two terms, it will improve the precision. However, calculating weights of co-occurrence of many terms needs a lot of computation, and most pairs have quite small weights.

To find pairs that have large weights, we used an inductive learning algorithm (based on $C4.5$[7], added some modifications) to detect terms and phrases pairs ('rules') that appear frequently in relevant documents and do not appear in irrelevant documents. We calculated tf and idf of each pairs (rules) as follows:

tf : smallest value of tf in the rule (We neglected negative terms in calculating tf).
idf : calculate the number of documents N_{rule} that include the rule as:

$$N_{rule} = collection\_size \times \prod_{each\_term\_appears\_in\_the\_rule} f(term) \tag{6}$$

We defined $f(term)$ as below:

If the term is positive one in the rule,

$$f(term) = number\_of\_documents\_including\_term / collection\_size$$

If the term is negative one in the rule,

$$f(term) = (collection\_size - number\_of\_documents\_including\_term) / collection\_size$$

3.2 Results

We submitted results of routing and filtering tracks to NIST. In the routing track, we made a slight mistake in computing the idf of pairs ('nttd7rt1'), so we later put the fixed result ('nttd7rt2').

Table 2 shows the results of the routing track. Our results were the best of all participants in TREC-7 routing track. (The results of nttd7rt1 are the same as those of nttd7rt2).

Run	Average Precision	Best	> average	Average	< average	Worst
nttd7rt2	.5139	30	11	7	2	0

Table 2: Results for nttd7rt2, routing

We tested the effects of our methods experimentally. Table 3 shows the effects of various parameters on the Rocchio feedback, without using our new methods. (Parameters '$\alpha : \beta : \gamma = 2 : 4 : -1$' are used in SMART at TREC2).

$\alpha : \beta : \gamma$	$1 : 8 : -8$			$2 : 4 : -1$		
number of terms	100	300	1000	100	300	1000
Ave.Prec	.4475	.4579	.4618	.4280	.4396	.4430

Table 3: Effects of various parameters on Rocchio feedback

Table 4 shows the effects of using our idiom processing ('+idiom') and detecting the co-occurrence of two terms ('2pair') before weight refining.

Run	1000 terms($\alpha : \beta : \gamma = 1 : 8 : -8$)	1000 terms + idiom	1000 terms + idiom + 2pair
Ave.Prec	.4618	.4726	.4771

Table 4: Effects of idiom processing and detection of two-term co-occurrences

In filtering track, the results are slightly better than the average of participants (nttd7bf1) and near the best of participants (nttd7bf2).

Run	Best	> average	Average	< average	Worst
nttd7bf1	6	7	23	7	7

Table 5: Results for nttd7bf1, filtering F1 measure

Run	Best	> average	Average	< average	Worst
nttd7bf2	16	4	22	6	1

Table 6: Results for nttd7bf2, filtering F3 measure

4 Conclusion

We described our system approach and discussed the evaluation results for ad-hoc and filtering in TREC-7. Results in filtering track were quite fine, especially in routing track.

The inductive-learning algorithm is used to detect co-occurrence pairs in the filtering track. This method is only effective when sufficient training documents are used. If only a few training documents are available, using non-judged documents as provisional irrelevant documents might be effective[3, 4].

Our investigation of idiom processing is still in progress, and we have not tried it with any languages other than Japanese. However, languages that have linguistic ancestors in common with English may not be suitable, because they have common lexical borrowings, including idioms.

References

[1] Chris Buckley and Gerard Salton. Optimization of relevance feedback weights. In *SIGIR*, pages 351–357, 1995.

[2] Chris Buckley, Gerard Salton, and James Allan. Automatic routing and ad-hoc retrieval using SMART: TREC2. In *TREC-2*, pages 45–55, 1994.

[3] Hiroyuki Nakajima and Tsuyosi Kitani. Inductive learning using document frequency for relevance feedback (in Japanese). Technical Report FI-45-97, Information Processing Society of Japan(IPSJ), 1997.

[4] Hiroyuki Nakajima, Tsuyosi Kitani, and Mamoru Okada. Improving relevance feedback through recognizing co-occurences of query words (in Japanese). *Transcations of IPSJ*, March 1999. (To be appeared).

[5] Ulrich Pfeifer and Tung Huynh. Freewais-sf, 1994.
ftp://ls6-www.infomatik.uni-dortmund.de/pub/wais/freeWAIS-sf-1.0.tgz.

[6] Porter, M.F. An algorithm for suffix stripping. *Journal of the Society for Information Science*, 3(14):130–137, 1980.

[7] Quinlan, J. R. *C4.5: Programs for machine learning.* Morgan Kaufman, 1993.

[8] Rocchio, J. J. Relevance feedback in information retrieval. In *The SMART Retrieval System*, pages 313–323. Prentice-Hall, 1971.

[9] Amit Singhal. AT&T at TREC-6. In *TREC-6*, pages 215–226, 1998.

A Large-Scale Comparison of Boolean vs. Natural Language Searching for the TREC-7 Interactive Track

William Hersh, Susan Price, Dale Kraemer, Benjamin Chan, Lynetta Sacherek, Daniel Olson
Division of Medical Informatics & Outcomes Research
Oregon Health Sciences University
Portland, OR, USA

Studies comparing Boolean and natural language searching with actual end-users are still inconclusive. The TREC interactive track provides a consensus-developed protocol for assessing this and other user-oriented information retrieval research questions. We recruited 28 experienced information professionals with library degrees to participate in this year's TREC-7 interactive experiment. Our results showed that this was a highly experienced group of searchers who performed equally well with both types of systems.

Introduction

The goal of the Oregon Health Sciences University (OHSU) TREC-7 Interactive Track experiment was to continue our investigation of Boolean vs. natural language searching. This year our experiments featured a large study sample as well as the use of experienced information professionals with a library degree.

Previous research in comparing Boolean and natural language systems has yielded conflicting results. The first study to compare Boolean and natural language searching with real searchers was the CIRT study, which found roughly comparable performance between the two when utilized by search intermediaries (Robertson and Thompson 1990). Turtle found, however, that expert searchers using a large legal database obtained better results with natural language searching (Turtle 1994). We have performed several studies of medical end-user searching comparing Boolean and natural language approaches. Whether using recall-precision metrics in bibliographic (Hersh, Buckley et al. 1994) or full-text databases (Hersh and Hickam 1995), or using task-completion studies in bibliographic (Hersh, Pentecost et al. 1996) or full-text databases (Hersh, Elliot et al. 1995), the results have been comparable for both types of systems. Our TREC-6 interactive experiments showed a trend towards better results for natural language searching, though the small sample size precluded achievement of statistical significance (Hersh and Day 1997).

The analysis in this paper focuses on system-oriented comparisons. A subsequent paper will focus on user-oriented factors associated with successful searching. This analysis looks specifically at three categories of data:

1. User characteristics
2. Statistical analysis of instance recall
3. Comparison of systems

Methods

The OHSU TREC-7 experiments were carried out according to the consensus protocol as described elsewhere in the proceedings. We used all of the instructions, worksheets, and questionnaires developed by consensus, augmented with some additional instruments. Our experiments compared Boolean and natural language searching systems as used by experienced librarian searchers.

Performance measures

The performance measures used in the TREC-7 interactive track were instance recall and instance precision. The searcher was instructed to look for instances of each topic (e.g., the number of discoveries by the Hubble telescope). Relevance assessors at NIST defined the instances from pooled searching results from all experimental groups. Instance recall was defined as the proportion of true instances identified during a topic, while instance precision was defined as the number of true instances identified divided by all instances identified by a user.

Systems

Both the Boolean and natural language systems were accessed via Web-based interfaces as shown in Figures 1 (Boolean) and 2 (natural language). There was a common IR system behind both interfaces, MG, a publicly available system with Boolean and natural language features (Witten, Moffat et al. 1994). MG was run on a Sun Ultrasparc 140 with 256 megabytes of RAM running the Solaris 2.5.1 operating system. Each interface accessed MG via CGI scripts which contained JavaScript code for logging search strategies, documents viewed (title displayed to user), and documents seen (all of document displayed by user). Searchers accessed each system with either a Compaq DeskPro 200 MHz Pentium Pro machine running Windows 95 and Netscape Navigator 3.0 or an Apple PowerMac 9600 with a 180 MHz PowerPC 604 running System 7.5.5 and Netscape Navigator 3.0. Figure 3 shows the documents displayed for viewing by both interfaces.

Experiments

Subjects were recruited by advertising over the American Society for Information Science Pacific Northwest Chapter listserv. The advertisement explicitly stated that participation would be limited to information professionals with a library degree and participants would be paid a modest remuneration for their participation. As subjects confirmed their participation, they were classified by type of library setting in which they worked: special (e.g., corporate, professional, or scientific), academic, and public.

The experiments took place in a computer lab at OHSU in the first half of August, 1998. An experimental session took four hours, with the first half used for personal data and attributes collection and the second half used for searching. All instructions, worksheets, and questionnaires developed by the consensus process are underlined in the remainder of this section.

The personal data and attributes collection consisted of the following steps:
1. Orientation to experiment (10 minutes)
2. Turn in Pre-Search Questionnaire (distributed by mail and completed before session)
3. Cognitive test administration (40 minutes)
4. Meyers-Briggs Personality Test (15 minutes)
5. Orientation to searching session and both retrieval systems, giving subjects the Searcher Instructions and demonstrating a search to them with each system (20 minutes)
6. Practice search using Hubble telescope search on second page of Searcher Instructions with both systems (10 minutes)

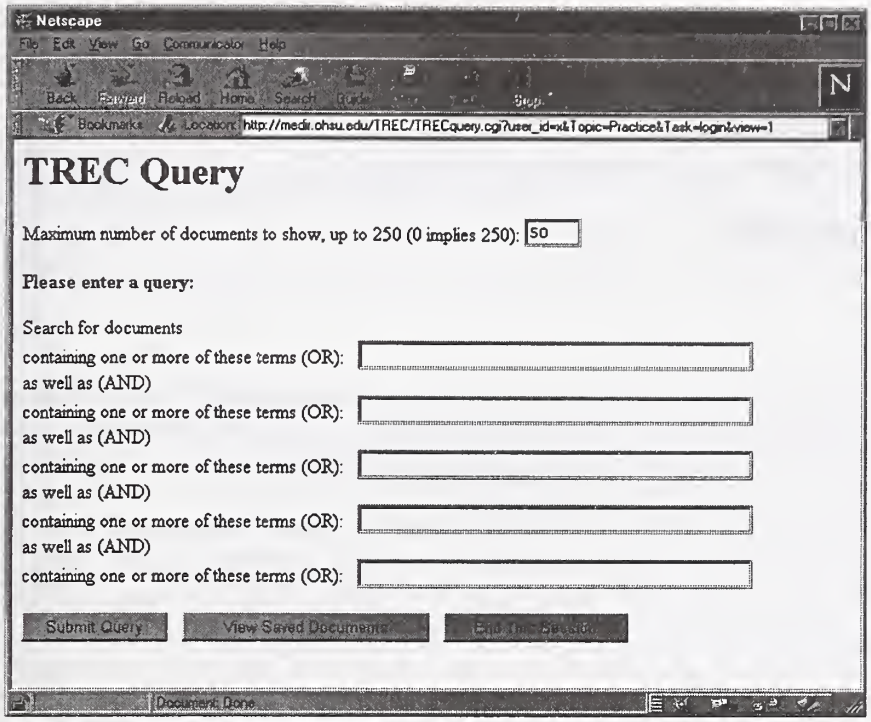


Figure 1 – Boolean searching interface.

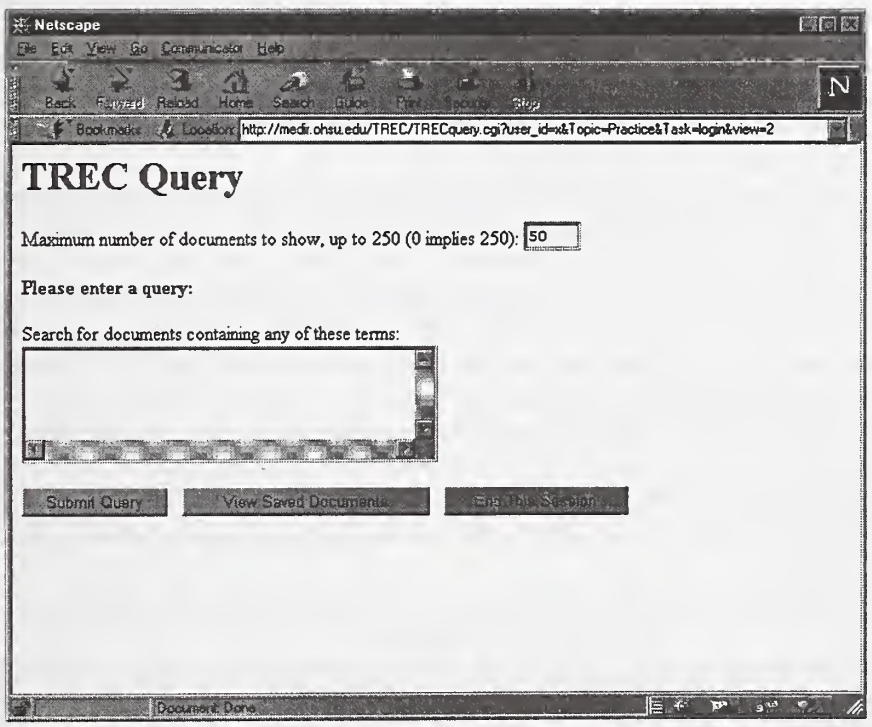


Figure 2 – Natural language searching interface.

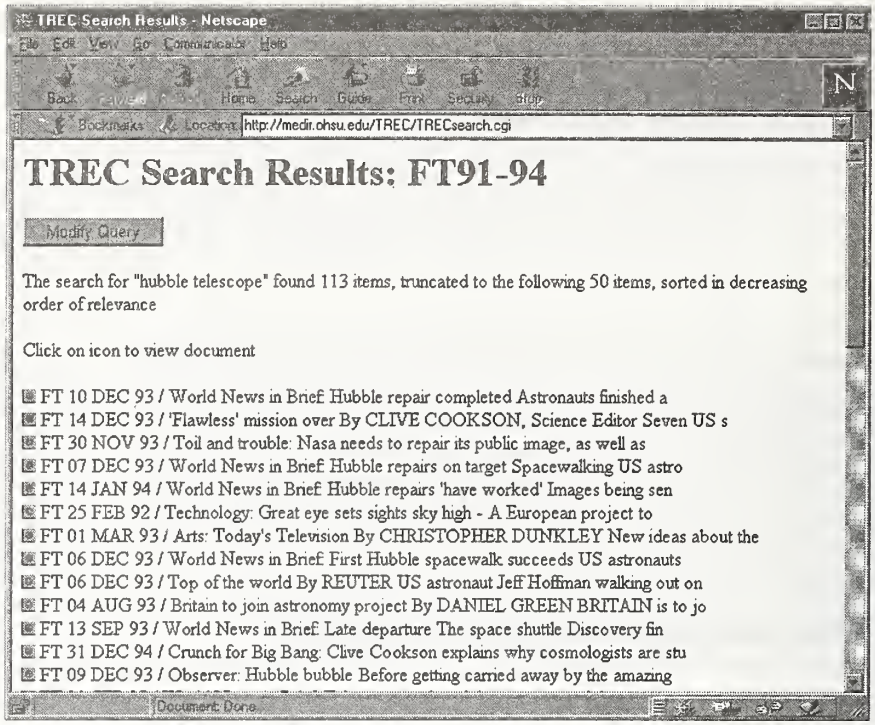


Figure 3 – Documents displayed for viewing.

The cognitive test administration consisted of four cognitive tests from the Educational Testing Service shown in past IR research to be associated with some aspect of successful searching. They included (ETS mnemonic in parentheses):

1. Paper folding test to assess spatial visualization (VZ-2)
2. Nonsense syllogisms test to assess logical reasoning (RL-1)
3. Advanced vocabulary test I to assess verbal reasoning (V-4)
4. Controlled associations test to assess associational fluency (FA-1)

The FA-1 test was used by all TREC-7 interactive groups per the consensus protocol.

The personal data and attributes collection was followed by a 10-15 minute break. The searching portion of the experiment consisted of the following steps:

1. Searching on first 4 topics with assigned system using Searcher Worksheet and Post-Topic Questionnaire (60 minutes)
2. Post-System Questionnaire for system used on first 4 topics and stretch (5 minutes)
3. Searching on second 4 topics with assigned system using Searcher Worksheet and Post-Topic Questionnaire (60 minutes)
4. Post-System Questionnaire for system used on second 4 topics and Exit Questionnaire (5 minutes)

Each subject was randomized into one of four blocks based on system (Boolean vs. natural language) and topic block group. Topic block 1 consisted of topics 365i, 357i, 362i, and 352i. Topic block 2 consisted of topics: 366i, 392i, 387i, and 353i. Users could be assigned as shown in Table 1.

Per the consensus protocol, each subject was allowed 15 minutes per query. Subjects were instructed to identify as many instances as they could for each query. They were also instructed for each query to write

each instance on the Searcher Worksheet and save any document associated with an instance (either by using the "save" function of the system or writing its document identifier down on the Searcher Worksheet).

The Post-System Questionnaire was augmented from the consensus protocol to include the Questionnaire for User Interface Satisfaction (QUIS) 5.0 instrument (Chin, Diehl et al. 1988). QUIS provides a score from 0 (poor) to 9 (excellent) on a variety of user factors, with the overall score determined by averaging responses to each item.

Analysis of Results

As noted above, the analysis of results consisted of three categories of data: user characteristics, statistical analysis of instance recall, and comparison of system factors. User characteristics were obtained from the Pre-Search Questionnaire, cognitive tests, and Meyers-Briggs Personality Test.

The statistical analysis was performed on a modified model of the TREC-6 statistical analysis (Lagergren and Over 1998). The experimental design and, hence, the appropriate analysis of variance model (ANOVA) for our study was more complex than proposed in the TREC-7 instructions and the TREC-6 analyses. One reason for the additional level of complexity was the addition of the factor of type of librarian into our design. In our study, the subjects were recruited from three different classes of librarians (academic, public, and special). Thus, subjects were nested in librarian type.

The design proposed by NIST was also too simple in several other regards. Several additional restrictions were imposed by the way the study was conducted. First, the study design was a modified crossover design. The subject was assigned to use one system on four consecutive topics and then crossed over to the other system on four different topics. The topics were allocated in two "blocks," designated in this analysis as B1 (topics T1 through T4) and B2 (topics T5 through T8). Subjects were also assigned a "sequence." That is, the subject was assigned to do the topics in either B1 or B2 first. The third factor was that of system (Boolean or natural language). The addition of the factors of block and sequence were required by the consensus design defined in the TREC-7 proposal.

There was also an additional restriction imposed by the TREC-7 design. Within each combination of block and sequence, the topics were always assessed in the same order. That is, topic 1 was always followed by topic 2, then 3, and then 4, while topic 5 was always followed by topic 6, then 7, and then 8. Thus there was a particular order of topics. We treated this order as a nested factor within the combination of block by sequence. However, it is important to note that topic was confounded with order. For example, the difficulty of the group of four topics may increase with successive topic number. Thus, the scores could decrease over time. On the other hand, the subjects could become more facile with the system over time and thus the scores might increase over time. Unfortunately, in the design used, topic order is confounded with topic, and thus there is no way to separate the effect of topic order from the effect of topic itself. While the term topic is used in the analyses described below, topic includes both topic itself and topic order.

Table 1 – Block assignment for users.

User	Block search 1	Block search 2
P1	B, Topic Set 1	N, Topic Set 2
P2	N, Topic Set 2	B, Topic Set 1
P3	B, Topic Set 2	N, Topic Set 1
P4	N, Topic Set 1	B, Topic Set 2

The original study design described in the TREC-6 interactive track specified three main effects: search system (SYSTEM), search topic (TOPIC), and study subject (ID) (Lagergren and Over 1998). The highest order model (M4) was defined as follows:

$$Y_{ijk} = \mu_{...} + a_i + b_j + g_k + (ab)_{ij} + (ag)_{ik} + e_{ijk}$$

where

Y_{ijk} = recall proportion for subject i, system j, topic k

$\mu_{...}$ = grand mean

a_i = subject effect, $i = 1,\ldots,8$

b_j = system effect, $j = 1,2$

g_k = topic effect, $k = 1,\ldots,8$

e_{ijk} = error

For the TREC-7 data, we decided to deviate from the models specified in the report previously cited. The model we developed is much more complex than the model above:

$$\begin{aligned} Y_{ijklmn} = \; & \mu_{.....} + a_{i(lm)} + b_j + g_{k(mn)} + l_l + h_m + q_n \\ & + (bl)_{jl} + (bh)_{jm} + (lh)_{lm} + (lq)_{ln} + (hq)_{mn} \\ & + e_{(ijklmn)} \end{aligned}$$

where

Y_{ijklmn} = recall proportion

$\mu_{.....}$ = grand mean

l_l = librarian type effect, $l = 1,2,3$

h_m = topic sequence effect, $m = 1,2$

$a_{i(lm)}$ = subject effect, nested, random, $i = 1,\ldots,8$

b_j = system effect, $j = 1,2$

q_n = topic block effect, $n = 1,2$

$g_{k(mn)}$ = topic effect, nested, random, $k = 1,\ldots,8$

$e_{(ijklmn)}$ = error

All effects are fixed except subject and topic, $a_{i(lm)}$ and $g_{k(mn)}$, respectively, which are random effects. Table 2 shows an example of the variable coding for four consecutive subjects.

One way to compare the two different models above is to consider the estimable functions for each ANOVA model. The estimable functions are linear combinations of the model parameters (the factor levels in an ANOVA model) which are invariant to the solution of the normal equations (Searle 1971). The model used in TREC-6 includes interactions for which the estimable function includes parameters other than the terms that form the interactions. Specifically, the system-by-topic interaction includes not only system and topic parameters but also subject parameters and the system-by-subject interaction also includes the topic parameters. This means that the test for these interaction terms are, to some extent, confounded with terms not included in the interactions. For example, the system-by-topic interaction is confounded by subject. The estimable functions for interaction terms in the OHSU ANOVA model include only parameters that are included in the interaction terms.

Table 2. Example of variable coding.

Librarian type	Block sequence	Searcher ID	System	Topic Block
Academic	B1-B2	P1-A	Boolean	B1
Academic	B1-B2	P1-A	Nat. Lang.	B2
Academic	B1-B2	P4-A	Nat. Lang.	B1
Academic	B1-B2	P4-A	Boolean	B2
Academic	B2-B1	P2-A	Nat. Lang.	B2
Academic	B2-B1	P2-A	Boolean	B1
Academic	B2-B1	P3-A	Boolean	B2
Academic	B2-B1	P3-A	Nat. Lang.	B1

The system comparison was obtained from collected data. No statistical analysis beyond the above was performed. The factors compared included:

1. Instance recall
2. Instance precision
3. Total number of search terms
4. Documents viewed (system showing title after search)
5. Documents seen (user displaying full document after selecting title)
6. Documents saved (representing an instance)
7. Post-system questionnaire rating of system being easy to learn
8. Post-system questionnaire rating of system being easy to use
9. Post-system questionnaire rating of system begin easy to understand
10. Post-system QUIS for user satisfaction
11. Exit questionnaire of which system was easier to learn
12. Exit questionnaire of which system was easier to use
13. Exit questionnaire of which system was liked better

Results

A total of 24 subjects participated in the study – eight each of special, professional, and academic librarians. All subjects were information professionals with a library degree. All completed the protocol as described above.

Searcher characteristics

The gender breakdown of the 24 subjects analyzed was 16 women and 8 men. The average age of all subjects was 41.1 years. Their average duration they had been doing on-line searching was 7.8 years. Table 3 shows their specific computer experience.

All subjects stated that they searched once or twice daily. All subjects also either agreed (41.7%) or strongly agreed (58.3%) with the statement, "I enjoy carrying out information searches."

Table 3 – Searchers' computer experience.

Experience with...	1 - No Experience	2	3 - Some experience	4	5 - A great deal of experience
Using a point-and-click interface (e.g., Macintosh, Windows)	0	0	1	2	21
Searching on computerized library catalogs either locally (e.g., local library) or remotely (e.g., Library of Congress)	0	0	0	7	17
Searching on CD-ROM systems (e.g., Encarta, Grolier, Infotrac)	0	1	5	12	6
Searching on commercial on-line systems (e.g., BRS Afterdark, Dialog, Lexis-Nexis)	1	5	6	5	7
Searching on World Wide Web search services (e.g., Alta Vista, Excite, Yahoo, HotBot)	0	1	1	8	14

Statistical analysis of instance recall

With 3 librarian types and 8 subjects per librarian type, there were 24 subjects. The 24 subjects each had 8 observations, one for each topic. Thus the design was balanced, and gave a total of 192 observations. Our model had 43 degrees of freedom, and the r^2 (amount of variance explained) for the model was 0.513.

Librarian type was a marginally significant effect. Pairwise comparisons revealed that special librarians were not significantly different from academic librarians (0.387 versus 0.344, respectively), and academic librarians were not significantly different from than public librarians (0.344 versus 0.302, respectively). However, there were difference between special and public librarians. Topic (nested within block and sequence) was also a significant effect, indicating variation in instance recall across different topics.

Across-system comparison

Table 4 shows comparisons across systems. Instance recall and precision were virtually identical for both systems. The total number of search terms used by subjects was also nearly identical. The number of documents viewed was much larger for the natural language system. The number documents both seen and saved was slightly higher for the Boolean system, though the difference was statistically significant. All of the post-system user satisfaction measures favored the Boolean system at or near statistically significant levels.

Table 5 shows responses from the exit survey. Users were asked their system preference in terms of ease of learning, ease of use, and overall preference. The Boolean system was clearly preferred on all three measures.

Table 4 – System effects ANOVA models.

Factor	Least squares mean		*p*-value	Model r^2
	Boolean	Nat. Lang.		
Instance Recall	0.346	0.342	0.8854	0.513
Instance Precision	0.688	0.698	0.7822	0.458
Total Search Terms	6.59	6.15	0.2815	0.617
Documents Viewed	141.1	241.4	**0.0004**	0.452
Documents Seen	14.68	13.58	**0.0641**	0.590
Documents Saved	5.32	4.65	**0.0543**	0.605
Post-System Easy to Learn (1–5) *	3.45	2.92	**0.0850**	0.808
Post-System Easy to Use (1–5) *	2.95	2.13	**0.0073**	0.868
Post-System Easy Understand (1–5) *	3.42	3.04	0.1310	0.864
Post-System QUIS average *	4.61	4.09	**0.0007**	0.969

* The nature of this varaible requires the exclusion of the Topic effect.

Table 5 - Exit survey responses, $n = 23$.

Factor	Boolean	Nat. Lang.	*p*-value
Users Said Easier to Learn	17	6	0.0347
Users Said Easier to Use	19	4	0.0026
Users Said Liked Better	19	4	0.0026

Discussion

This experiment assessed the ability of highly experienced information professionals to identify instances of topics in an on-line database. The presearch questionnaire showed they had a great deal of searching experience and computer experience in general. They performed online searching and carried it out on a daily basis. These subjects strongly preferred the Boolean searching interface, although they used about the same number of search terms and chose the same number of documents for seeing. Their instance recall and precision were virtually identical, indicating their searching performance with each was comparable.

There were some other observations of interest revealed by this study. First, subjects used slightly more than 6 unique search terms per query. The most notable aspect of this result is that it is about three times higher than the average number of terms used by general users of Web search engines (Jansen, Spink et al. 1998). Second, there were differences based upon the type of library position in which they worked. Those from special libraries did better as a group than those from academic libraries, while the latter outperformed those from public libraries.

Further analysis will change the orientation from a system-oriented to a user-oriented perspective. We will look at association of experience attributes, cognitive factors, personality types, and system operations with performance measures. We will also compare document-oriented recall and precision with the instance-oriented measures used in this study. In the future, we hope to compare other advanced retrieval systems and their features using the TREC-7 interactive track technique.

References

Chin, J., V. Diehl, et al. (1988). Development of an instrument measuring user satisfaction of the human-computer interface. *Proceedings of CHI '88 - Human Factors in Computing Systems*, New York, ACM Press, 213-218.

Hersh, W., C. Buckley, et al. (1994). OHSUMED: an interactive retrieval evaluation and new large test collection for research. *Proceedings of the 17th Annual International ACM SIGIR*, Dublin, Springer-Verlag, 192-201.

Hersh, W. and B. Day (1997). A comparison of Boolean and natural language searching for the TREC-6 interactive task. *Proceedings of the Sixth Text Retrieval Conference (TREC-6)*, Gaithersburg, MD, NIST, 585-595.

Hersh, W., D. Elliot, et al. (1995). Towards new measures of information retrieval evaluation. *Proceedings of the 18th Annual International ACM SIGIR*, Seattle, WA, ACM Press, 164-170.

Hersh, W. and D. Hickam (1995). An evaluation of interactive Boolean and natural language searching with an on-line medical textbook. *Journal of the American Society for Information Science* 46: 478-489.

Hersh, W., J. Pentecost, et al. (1996). A task-oriented approach to information retrieval evaluation. *Journal of the American Society for Information Science* 47: 50-56.

Jansen, B., A. Spink, et al. (1998). Real life information retrieval: a study of user queries on the Web. *SIGIR Forum* 32: 5-17.

Lagergren, E. and P. Over (1998). Comparing interactive information retrieval systems across sites: the TREC-6 interactive track matrix experiment. *Proceedings of the 21st Annual ACM SIGIR*, Melbroune, Australia, ACM Press, 162-172.

Robertson, S. and C. Thompson (1990). Weighted searching: the CIRT experiment. *Informatics 10: Prospects for Intelligent Retrieval*, York, ASLIB, 153-166.

Searle, S. (1971). Linear Models. New York, J. Wiley and Sons.

Turtle, H. (1994). Natural language vs. boolean query evaluation: a comparison of retrieval performance. *Proceedings of the 17th Annual International ACM SIGIR*, 212-220.

Witten, I., A. Moffat, et al. (1994). Managing Gigabytes - Compressing and Indexing Documents and Images. New York, Van Nostrand Reinhold.

A Two-Stage Retrieval Model for the TREC-7 Ad Hoc Task

Dong-Ho Shin
Artificial Intelligence Lab (SCAI)
Interdisciplinary Program in Cognitive Science
Seoul National University
Seoul 151-742, Korea
E-mail: dhshin@scai.snu.ac.kr

Byoung-Tak Zhang
Artificial Intelligence Lab (SCAI)
Dept. of Computer Engineering
Seoul National University
Seoul 151-742, Korea
E-mail: btzhang@scai.snu.ac.kr

ABSTRACT

A two-stage model for ad hoc text retrieval is proposed in which recall and precision are maximized sequentially. The first stage employs query expansion methods using WordNet and on a modified stemming algorithm. The second stage incorporates a term proximity-based scoring function and a prototype-based reranking method. The effectiveness of the two-stage retrieval model is tested on the TREC-7 ad hoc text data.

1 Introduction

Performance of text retrieval systems is usually measured on the basis of recall and precision. Recall is defined as the proportion of relevant documents retrieved, while precision is the proportion of retrieved documents that is relevant. We wish both high recall and high precision, but the one should usually be traded for the other. To increase recall, just retrieving many documents would be helpful. But then precision decreases and vice versa. In our early experiments on TREC collections we tried to improve retrieval performance by directly optimizing a combination of both factors. However, managing so large a size of documents in one homogeneous model was both inefficient and ineffective. In addition, many techniques were not applicable simply due to its computational efforts for optimizing both factors at the same time. These failures led us to use a two-stage model which deals with recall and precision separately.

In the two-stage retrieval model, we first attempt to maximize the recall performance and then try to improve precision subsequently. One advantage of this approach is that the effectiveness of several techniques can be analyzed separately since this separation reduces the interference effect of recall and precision. Another advantage is that this separate optimization can reduce computational overhead since the techniques for improving precision are applied to a small subset of documents which have been retrieved in the first stage. In this article, we describe the techniques for improving recall and precision separately, and report the experimental results obtained on the TREC-7 ad hoc task.

The paper is organized as follows. In Section 2, we describe the system architecture of the two-stage retrieval model. Section 3 describes the techniques we studied for improving recall and precision. Section 4 reports the experimental results on the TREC-7 ad hoc document set using combinations of the techniques in the framework of the two-stage retrieval model. Section 5 draws our conclusions from these experiments.

2 The Two-Stage Retrieval Model

Figure 1 illustrates the architecture of our information retrieval system for the TREC-7 ad hoc task. The system is based on the vector space model. Formally, a document is represented as a list of terms or term vectors. A document collection is represented as a term-document matrix which is normally very sparse. A query consists of a list of terms, too.

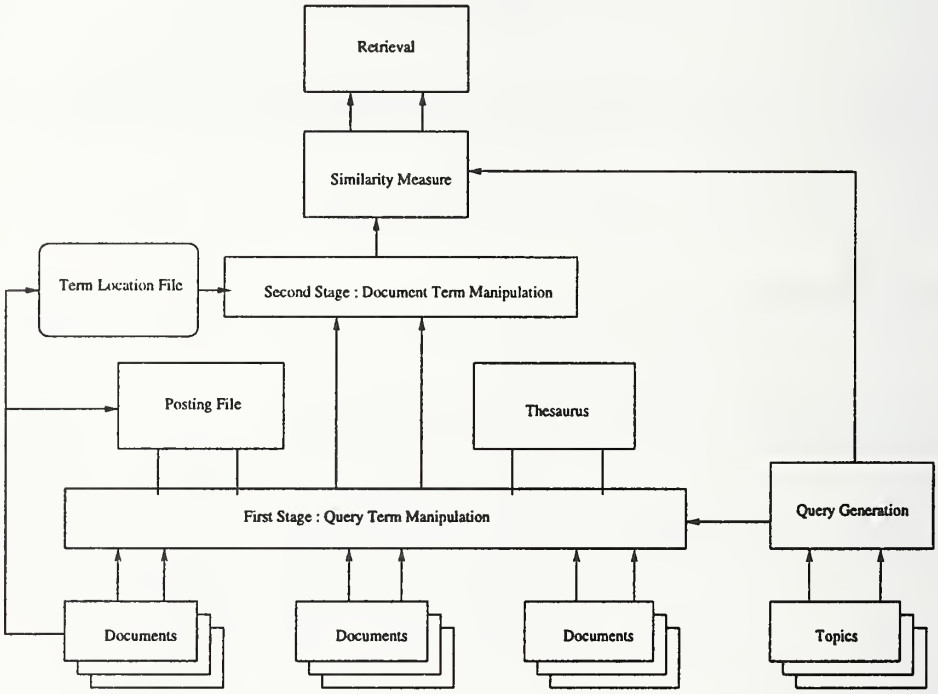


Figure 1: The two-stage model for text retrieval.

The documents are indexed by using the classical $tf \cdot idf$ weighting scheme [7]:

$$w_{ij} = tf_{ij} \cdot \log\left(\frac{N}{df_j}\right), \tag{1}$$

where w_{ij} is the weight of jth term in the ith document, tf_{ij} is the frequency of the jth term in the ith document, N is the total number of documents in the collection, and df_j is the number of documents in which the jth term occurs.

We have used various stemming methods for TREC-7 data, but could not achieve any significant performance improvement. For this reason, we used all words as indexing terms that appeared in

raw documents but not in the stop-list, and then tried to improve recall and precision in sequence. In the first stage, operations are carried out with respect to query terms, while in the second stage operations are performed with respect to document terms. The main objective of the first stage is to improve recall, while the second stage aims mainly at improving precision.

Techniques used in the first stage include query expansion methods based on a modified stemming algorithm and the WordNet. Techniques adopted in the second stage include methods using a term proximity-based scoring function and a prototype-based ranking method.

3 Retrieval Methods

3.1 Query Expansion

Two methods for query expansion have been studied. One is substring matching in query generation. The original query terms are generated from the topic text in the TREC-7 documents using the same method as for document indexing. Query j for topic j is then represented as a vector $\mathbf{v}_j = (v_{j1}, ..., v_{jk}, ..., v_{jn})$, where v_{jk} denotes the weight of term k in query j. In the substring matching method, the weight v_{jk} of term k is assigned proportional to the document frequency df_k of the corresponding term. The idea behind this weighting scheme is that people tend to use low-frequency terms in queries, and thus document frequency information is important. Note that this is a $tf \cdot idf$ method modified for query indexing.

We also studied a query expansion method using a thesaurus. The original query terms v_{jk} are expanded by their synonyms and hypernyms which are found using the WordNet [4].

3.2 Query Weighting

Once the queries are generated, they are matched against documents as follows. Let $\mathbf{w}_i = (w_{ik})$ and $\mathbf{v}_j = (v_{jk})$ denote the term vectors for document i and query j, where w_{ik} and v_{jk} are weight values for kth term in document and query, respectively. The relevance of document i with respect to query j is scored by the inner product of the document and query vectors:

$$S_j(i) = \mathbf{w}_i \cdot \mathbf{v}_j = \sum_{k=1}^{n} w_{ik} \cdot v_{jk}, \tag{2}$$

where k runs over the terms in the vocabulary of size n.

In a modified version, we use query weighting after query expansion. Here we regard the terms with low document frequency more important. To implement this, query terms are ranked and then the importance of term k is weighted by a power function

$$(m - rank_{jk})^p, \tag{3}$$

where m is the number of terms in query j and $rank_{jk}$ is the rank of the kth term in query j. Large p gives more weight to the ranking factor. In effect, the similarity score of document i to query j is defined as:

$$S_j(i) = \sum_{k=1}^{n} w_{ik} \cdot v_{jk} \cdot (m - rank_{jk})^p. \tag{4}$$

The lower the frequency of query terms is, the greater the score of the document. The score can be adjusted by the p value.

3.3 Term Proximity Information

The second stage aims at improving the retrieval precision. We experimented with two methods. One is using the word proximity information. Though appearing in the same document, two different terms may have no relationship with each other if one term occur far away from the other. To use proximity information, we apply an additional query operator, called NEAR, that take into account the distance between terms in the document. The proximity $prox(r, s)$ of rth and sth terms in document i is then defined as

$$prox(r, s) \propto \frac{1}{dist(w_{ir}, w_{is})} \tag{5}$$

where $dist(\cdot)$ is a distance measure. The proximity score $prox(i)$ of ith document is then defined as the sum of the proximity of term pairs in the document:

$$prox(i) = \sum_r \sum_s prox(r, s) \tag{6}$$

where r and s run over the terms in the ith document.

3.4 Prototype-Based Reranking

The second method for improving precision is to use the documents retrieved to rerank them. Among the retrieved documents, we select the top K documents which are then used to construct prototype documents. Let $\mathrm{p}_j = (p_{j1}, ..., p_{jn})$ denote the weight vector of the jth prototype, where n is the number of indexing terms. The similarity of document i to prototype j is then measured by cosine coefficient:

$$sim(i, j) = \frac{\sum_{k=1}^{n} w_{ik} \cdot p_{jk}}{\sqrt{\sum_{k=1}^{n} {w_{ik}}^2 \cdot \sum_{k=1}^{n} {p_{jk}}^2}} \tag{7}$$

where w_{ik} and p_{jk} are term weights for document i and prototype j, respectively.

4 Experimental Results

The methods described in the previous section have been used in various combinations for the ad hoc query on TREC-7 collections.

Table 1 summarizes the experimental results. The first row, i.e. experiment number 1, shows the results of the baseline retrieval method. This is the results we submitted to NIST in the summer of 1998. After this official submission, we extended the system by the techniques described in the previous section. Figure 2 shows the recall-precision curves for the methods tested.

Rows 2 and 3 in the table show the results obtained by using the query weighting method. In experiment 2, long queries were used, i.e. queries were generated from the title, description, and narrative fields of the topic text. Experiment 3 used short queries, i.e. the title field only. Term

Experiment No.	Run Type	Topic Length	Average Precision
1	baseline	T+D+N	0.0477
2	qwgt1	T+D+N	0.0601
3	qwgt2	T	0.0903
4	qexp1 (WordNet)	T	0.0967
5	qexp2 (range 1)	T	0.0778
6	qexp3 (range 3)	T	0.0640
7	proto	T	0.0652
8	proto + prox	T	0.1258
9	qwgt + prox	T	0.1468
10	qwgt + prox + proto	T	0.1277

Table 1: Comparison of average precisions for various combinations of methods. Symbols denote the names of various techniques: qwgt = query weighting, qexp = query expansion, prox = proximity infomation, proto = prototype-based ranking.

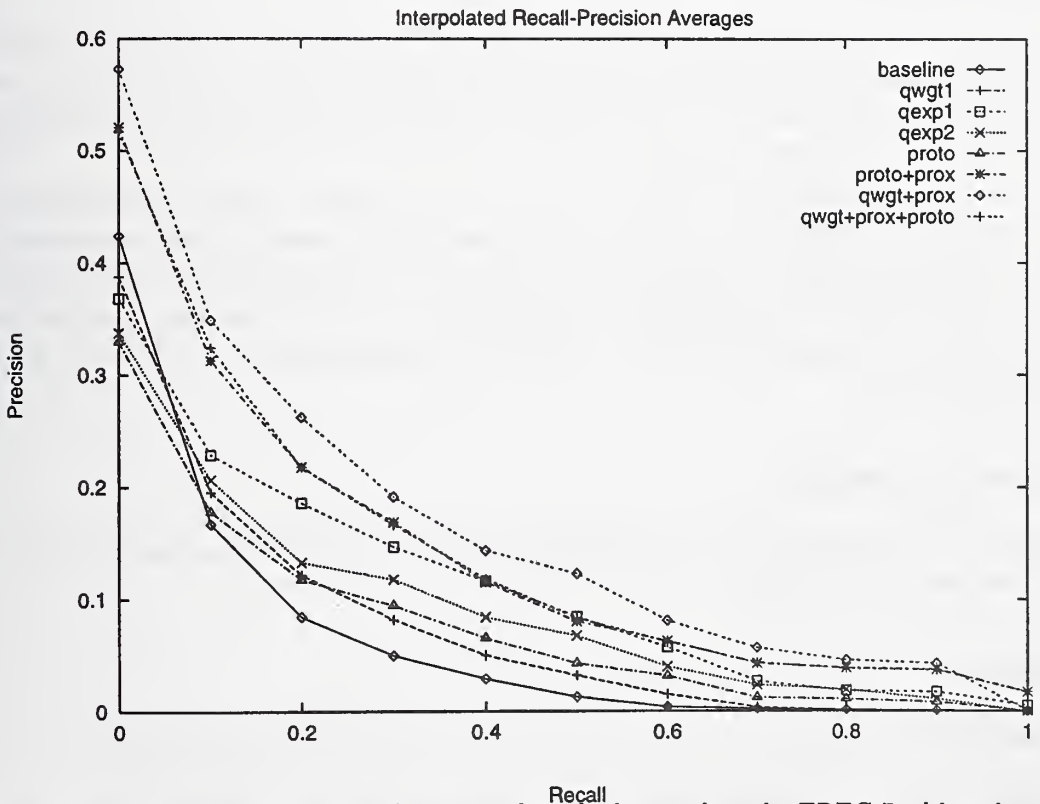


Figure 2: Recall-precision curves for the retrieval methods tested on the TREC-7 ad hoc document collection.

weighting improved the baseline method and, in terms of query length, short queries were better than long ones.

Rows 4 to 6 in the table summarize the performances of query expansion. Compared are three variants: using WordNet (experiment 4), substring matching with range of 1 (experiment 5), substring matching with range of 3 (experiment 6). Among these, the WordNet-based expansion method was the best. In substring matching, increasing the range of matching degraded the precision.

Experiments 7 to 10 are related with using proximity information and prototype-based reranking. In general, these methods and their combinations improved the precision. In particular, the effect of proximity information was more significant than the others. This is due to the fact that the term proximity measure extracts context information of terms, which is used as an additive term to the scoring function. In contrast, the reranking based on prototypes did not lead to significant improvements. This seems attributed to the fact that we used in these experiments one prototype constructed as the average of K document vectors rather than multiple prototypes; It is not very likely that the average pattern of top K (in our case 20) documents is representative of all the documents in the topic class.

5 Conclusions

We presented a two-stage model for the TREC-7 ad hoc retrieval task. By dividing the retrieval process into two stages, we could reduce the complicated interference effect of recall and precision on the whole performance. We proposed and experimented with various techniques that were designed to improve recall and precision, respectively.

Tested on the TREC-7 ad hoc text data, we improved the average precision performance from 0.0477 for the baseline method to 0.1468 for the two-stage method combining query manipulation and term proximity information. Though this performance is not among the best of the TREC-7 ad hoc entries in absolute value, we think it is a significant improvement as our first experiments in TREC. Refinements of proposed methods are in progress to further improve the performance of the current retrieval system.

Acknowledgements

This research was supported by the Korea Ministry of Information and Telecommunications under grant C1-98-0068-00 through IITA.

References

[1] Frakes, W.B and Ricardo, Baeza-Yates, *Information Retrieval*, Prentice-Hall, 1992.

[2] Korfhage, Robert R., *Information Storage And Retrieval*, John Wiley & Sons, 1997.

[3] Lee, J.H., Analyses of Multiple Evidence Combination, *SIGIR-97*, pp. 267-276, 1997.

[4] Miller, G.A., Five papers on WordNet, *International Journal of Lexicology*, vol. 3, no. 4, 1990.

[5] Robertson, S.E. and Sparck-Jones, K., Relevance weighting of search terms. *Journal of the American Society for Information Science* 27:-1976.

[6] Rocchio, J., Relevance feedback information retrieval, In G. Salton, editor, *The Smart Retrieval System - Experiments in Automatic Document Processing*, Prentice-Hall, pp. 313-323, 1971.

[7] Salton, G., *Automatic Text Processing*, Addison-Wesley, 1989.

[8] Yang, Y., Noise Reduction in a Statistical Approach to Text Categorization, *SIGIR-95*, pp. 256-263, 1995.

SPIDER Retrieval System at TREC7

Martin Braschler**, Bojidar Mateev*, Elke Mittendorf*,
Peter Schäuble*,Martin Wechsler**
* Swiss Federal Institute of Technology (ETH), CH-8092 Zürich
** Eurospider Information Technology, Schaffhauserstr. 18,
CH-8006 Zürich

Abstract

This year the Zurich team participated in two tracks: the automatic-adhoc track and the crosslingual track. For the adhoc task we focused on improving retrieval for short queries. We pursued two aims. First, we investigated weighting functions for short queries—explicitely without any kind of automatic query expansion. Second we developed rules that automatically decide for which queries automatic expansion works fine and for which it does not.

For the cross-language track, we approached the problem of retrieving documents from a multilingual document pool containing documents in all TREC CLIR languages. Our method uses individual runs for different language combinations, followed by merging their results into one final ranked list. We obtained good results without sophisticated machine translation or costly linguistic resources.

1 Introduction

For this year's adhoc runs we pursued ideas that were introduced by users with the hope to unite the weighting schemes based on probabilistic models with weighting schemes which take into account the user's subjective expectations. That is, we re-investigated the influence of feature-frequency weighting and the influence of proximity in particular for short queries.

Our runs for this year's cross-language track introduce a couple of new advancements compared to last year's submission: thesaurus filtering techniques and merging of results from multiple CLIR runs. The filtering techniques help us to automatically identify bad entries in the thesauri by comparing entries across different thesauri. The merging of results from individual crosslingual runs works by using document alignments computed for the TREC collections. The runs do not use any costly linguistic resources, and, in contrast to last year, we did not use any machine translation system.

2 General system description and reference method

Our indexing vocabulary Φ consists of single words $\varphi_i \in \Phi$ reduced by the Porter algorithm. A valid single word has a minimal length of three characters for the adhoc runs and two characters for the CLIR runs. Stopwords are removed.

The Retrieval Status Values (RSV) are obtained by

$$\mathrm{RSV}(q, d_j) = \sum_{\varphi_i \in \Phi(q,d_j)} a_{ij} \cdot b_i, \tag{1}$$

where $\Phi(q, d_j)$ denotes indexing features occurring in both the query q and the document d_j, a_{ij} denotes the weight of the document feature and b_i denotes the weight of the query feature.

As a **baseline** we used Lnu.ltn weighting as reported in [7] but applied minor changes. For adhoc experiments and final runs we used residual inverse document frequency (measure for the deviation from Poisson distribution) as suggested in [3]. The document length consisted of the number of (unique) stemmed and unstemmed features (this normalization has implementational reasons and unfortunately deteriorates the effectiveness).

Whe will refer in the next sections to **pseudo feedback** as choosing of features from the top ranked documents according to the Rocchio re-weighting function.

3 Adhoc Retrieval

Our focus in this year's adhoc track was on short queries. In last year's TREC we could see an impressive improvement on short queries based on the so-called "pseudo" feedback on top ranked documents [8]. Unfortunately users are often not satisfied with a "pseudo" feedback strategy. They are unhappy finding documents ranked on the top which do not contain the majority of the query features. This problem is even more severe when users are looking for a document containing proper names (person names, organization names etc.). Our approach is:

1. to investigate how good the retrieval functions can be **without pseudo feedback**, and to try to find a retrieval function which is closer to the user's usual expectations.

2. to **re-investigate the feature-frequency weighting** by looking for an alternative of the usual logarithmic feature-frequency weights, such as in the Lnu.ltn weighting scheme. The weighting should reward a document containing all query features.

We included **proximity** information and the coordination level match of features into the weighting scheme. In a second approach we designed rules for finding in which for which queries the proximity-based method works better than the baseline Lnu.ltn plus pseudo feedback. We tried to design the rules in such a way that the proximity method is chosen for queries where the user normally expects that all query features occur and for the rest of the queries the Lnu.ltn method with "pseudo" feedback is chosen.

3.1 Experiments and submitted adhoc runs

The baseline Lnu.ltn is described in Section 2. For the adhoc "pseudo" feedback run we chose the top 15 features (query expansion) from the seven top ranked documents. The parameters for the Rocchio formula were $\alpha = 7$ and $\beta = 3$. The slope for the Lnu.ltn was $s = 0.1$.

3.2 Method (ETHAR0)

The first method that we submitted uses the following strategy to rank the documents:

1. The documents, which have the maximal coordination-level match that is achieved by any document, are ranked before the documents with lower coordination-level match.

2. Documents with maximal coordination-level are ranked using a proximity weighting.

   (a) Determine the smallest window in the document that has the maximal coordination level. Documents with the narrowest window are ranked higher than documents with a larger window.

   (b) Documents ranked equally by (a) are further ranked according to how far is this window from the beginning of the document.

3. Documents with lower coordination-level are ranked according to the Lnu.ltn weighting scheme (disregarding the particular coordination level).

3.3 Method (ETHAC0)

We noticed that there are several queries for which pseudo feedback works significantly better than, e.g., the proximity and Lnu.ltn based method ETHAR0 and vice versa. Our intention is to find simple patterns which indicate when a short query weighted using a method based on proximity information outperforms a "pseudo" feedback method using a Lnu.ltn scheme. Our idea for finding such patterns is based on part-of-speech (POS) information. This information can be automatically obtained using a POS tagger (in our case the Brill tagger [2]). We compare the result list from method described in Section 3.2 with a result list obtained from baseline Lnu.ltn plus "pseudo" feedback. We focus on finding patterns for queries with noun phrases. The idea is that in such cases proximity might be a very useful information. Users are satisfied when—for a query consisting of noun phrases—they find documents which contain the same or only slightly different phrases.

The method decides for each query—based on a set of rules—which ranking method to use, either the ETHAR0 method or the Lnu.ltn plus "pseudo" feedback. The decision rules have been optimized on TREC6 queries (301–351). The rules are:

Choose method ETHAR0 if the query consists of

1. two rare nouns (the information whether or not a noun is rare is taken from the WordNet lexical database),

2. three nouns,

3. a nonstopword adjective followed by one or two nouns.

Choose method Lnu.ltn plus "pseudo" feedback otherwise.

According to these rules for 20 TREC6 short adhoc topics and for 25 TREC7 short adhoc topics the (coordination-level and proximity-based) method ETHAR0 is chosen.

On the one hand there is a large variety in the structure of noun phrases, on the other hand we have only a limited set of training queries (TREC6). It was obvious that under such conditions the rules for the choice method ETHAC0 might not be very robust. Moreover, the POS tagger does not perform as well on title-like short queries as it does on complete sentences. Despite of the possible lack of robustness we gave it a try.

3.4 Method ETHAB0

In this method the retrieval status values are determined in a first pass by Robertson–Sparck Jones (RSJ) [5] weighting, i.e., the ranking of documents is based primarily on inverse document frequency weights. After determining classes of documents that have the same RSJ retrieval status value. The documents within one RSJ-class are fine-ranked according to the Lnu.ltn weighting. Note that the RSJ-classes of documents are large because the queries are short.

3.5 Experiments

We report on experiments with the methods described above (Lnu.ltn baseline, Lnu.ltn plus pseudo-feedback, ETHAR0, ETHAC0, ETHAB0). A result list was produced for each of the methods for both TREC6 (301–350) and TREC7 (351–400).

Table 1 shows the average precision over all queries for the methods on both topic sets. In addition, Figure 1 and Figure 2 provide boxplots that visualize the distribution of the average precision per query.

Runs/Queries	TREC6 (301-350)	TREC7 (351-400)
Lnu.ltn	0.2202	0.1631
Lnu.ltn with pseudo feedback	0.2366	0.1853
ETHAR0	0.2251	0.1597
ETHAC0	0.2473	0.1645
ETHAB0	0.2306	0.1601

Table 1: Average precision for all queries

3.6 Discussion of Results

Unfortunately the results on TREC-6 and TREC-7 data are not consistent. The two methods that emphasise feature occurrence before feature frequencies, ETHAR0 and ETHAB0, yield a higher average precision on TREC-6 than standard Lnu.ltn, where high feature frequencies can overweigh pure occurrence of a feature. On TREC-7 the results are vice versa, standard Lnu.ltn outperforms our two new weighting schemes.

The tagger-based query classification method (used for ETHAC0), which is able to distinguish those TREC-6 queries for which pseudo feedback works better than the proximity-based method ETHAR0 and vice versa, fails for TREC-7 queries. The submitted method ETHAC0 is worse than the pseudo-feedback method, on which it is based.

In summary, in this year's TREC we have taken a chance and tried rather nonorthodox methods to emphasize occurrence over frequency, to emphasize proximity of features and to decide whether or not to use pseudo feedback. We know that the design of our methods has to be more robust.

4 Cross-Language Information Retrieval

Our participation this year consists of three runs, all using the German topics, and retrieving documents from the full pool of documents in all four languages (German, French, Italian

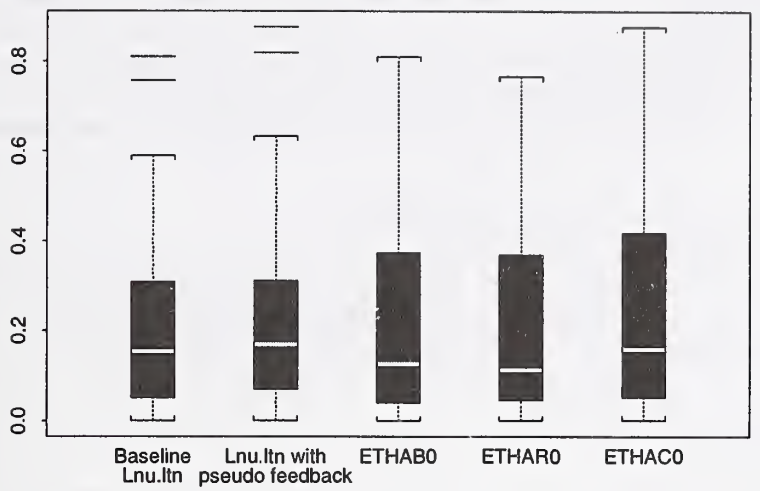


Figure 1: Experiments on TREC6 (topics 301–350), showing the distribution of average precision per query (y-axis).

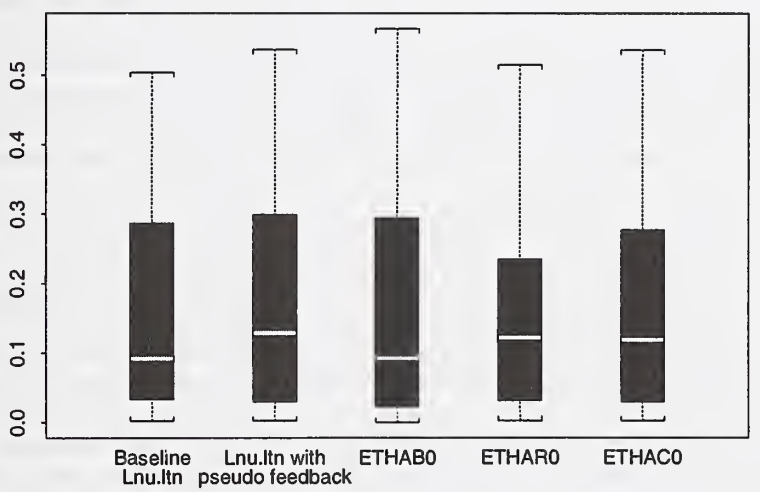


Figure 2: Experiments on TREC7 (topics 351–400), showing the distribution of average precision per query (y-axis).

and English). The runs were produced by doing individual runs for language pairs (e.g. German to French), and merging the results to form the final ranked list. Our focus this year was therefore both on improving such individual runs and on solving the problem of merging their results.

Merging individual ranked lists gives us the flexibility to use all the resources we had available for specific language pairs, instead of being restricted to a least common denominator across the languages. Consequently, not all the individual runs were produced in the same way.

4.1 Experiments and submitted runs

The weighting scheme for all runs was set to Lnu.ltn, as described in Section 2. The slope was set to $s = 0.24$. The parameters for the Rocchio formula were $\alpha = 1$ and $\beta = 0.81$, with the 20 best terms taken from the top 10 documents for pseudo-feedback.

4.2 Monolingual run

- German→German
  The German monolingual run was produced by doing an initial retrieval run using all configurations as described above. We then used pseudo-feedback [8] to expand the query by terms coming from the top ranked documents. For stemming, the German stemmer that is included with NIST ZPRISE system was adapted.

4.3 Crosslingual runs

For our crosslingual runs, we used similarity thesauri (for French, Italian) and a wordlist (a simplistic bilingual dictionary, for English). A similarity thesaurus is a data structure that is automatically derived from appropriate training data. While originally developed for monolingual query expansion, it has been successfully applied to the problem of crosslingual retrieval by using multilingual documents as training data. These documents are obtained through a document alignment process [1] that finds pairs of similar documents in different languages and creates a single multilingual document by joining them. We applied this alignment process to the TREC collections for all three language combinations described below.

The thesaurus itself is built by using co-occurrence statistics from the training data to determine the similarity of every pair of terms in the collection. The most similar pairs are stored to disk. The similarity can be calculated by exchanging the role of documents and terms in conventional weighting schemes. This way, the similarity of terms is determined through the sets of documents that indexes them. A similarity thesaurus can cover very large vocabularies; it usually however also contains various low quality entries. More details can be found in [6].

We also used a German→English wordlist we assembled from various free sources on the Internet as a simplistic form of a bilingual dictionary. While the resulting list is rather large (141,240 entries, or 85,931 unique "head" entries), it contains many questionable or even wrong entries, since we were not concerned with any clean-up of the data. We believe that through coupling the wordlist with our corpus-based alignment techniques the resulting retrieval process gets robust with respect to such "noise".

- German→French
  The German→French crosslingual run uses a similarity thesaurus built on the SDA data. Compared to the experiments described in [4], we had more data available from SDA than last year, also covering time periods not present in the SDA texts used for this year's track. Our complete document pool consisted of all German and French SDA news texts from 1988 to March of 1998 (1988-90 of this pool comes from the official track data). We exploited this by using thesaurus merging techniques to filter the entries in individual thesauri built over different time periods. While this allows us to generalize the resulting thesaurus, this also means that the fact that for the CLIR track the thesaurus can be built on the collection itself is not exploited as much as for last year's experiments. We think, however, that using a generalized thesaurus gives a more realistic scenario.

  The similarity thesauri are employed in much the same way as outlined in [1]. A pseudo-translation is produced using the similarity thesaurus, and is then combined with terms coming from a selection process on the top aligned documents, using the same document alignments as computed for the creation of the thesaurus. The initial monolingual run used to determine the aligned documents was produced as outlined above.

- German→Italian
  The German→Italian run is very similar to the German→French run. Again, we had additional SDA data available (the complete pool consisting of all Italian SDA news texts from end of 1989 to March of 1998), which allowed us to build several thesauri and employ our thesaurus filtering techniques. The pseudo-translation coming from the thesaurus is also combined with terms from the top aligned documents.

- German→English
  Because we did not think that a German→English similarity thesaurus with satisfactory quality can be derived from the differently focused international AP news and the national SDA news, we used the Internet wordlist for German→English query translation. This also nicely demonstrates the ability to use different resources for different language pairs in our approach. Query translation itself was done using a simplistic word-by-word dictionary lookup procedure, but was again complemented with terms coming from the top aligned documents. This combination helps offset many of the sense ambiguity problems associated with simple dictionary lookup.

4.4 Merging

The ranked lists of the four individual runs were then merged using the technique described in [1]. By again making use of the document alignments from the individual collections of the track data, we produced tables giving the relations between scores of the individual runs. It is then possible to map these scores to a common range using linear regression. After rescaling the scores, merging is done by simply sorting a joined version of both ranked lists. Repeating this for every language, we ultimately produced the final ranked lists that were submitted.

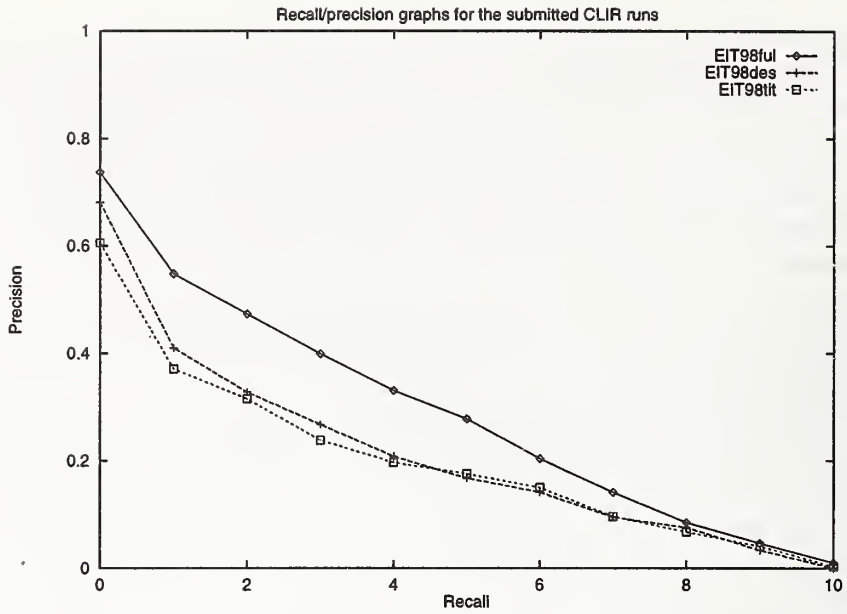


Figure 3: Comparison of the submitted cross-language runs.

4.5 Results

We submitted three runs for TREC-7, all using the German topics and the full document pool. The difference in the runs was in the topic fields used: EIT98ful used all topic fields ("full topics"), EIT98des used the title and description fields, and EIT98tit used the title field only.

We feel the results (Table 2) are very encouraging given the fact that we used no costly linguistic resources to produce the runs. The similarity thesauri were derived completely automatically from training data, whereas the wordlist was taken from free Internet sources. As mentioned, however, the presented approach is flexible towards incorporating further resources should they be available.

Clearly, the run using the full topics (EIT98ful) is outperforming the other two submissions (see Figure 3. We believe that long queries are beneficial to corpus-based techniques like the similarity thesaurus that have a broad vocabulary coverage, but also contain bad entries that may have an adverse effect if little input is available for the translation process. The fact that the full queries perform so much better than the other two query sets also shows that our corpus-based techniques suffer less from a word ambiguity problem than purely dictionary-based approaches. The latter approaches will produce very long output if the query length increases; this due to every word potentially having more than one translation. Such very long queries are not likely to perform well. The similarity thesaurus however allows to translate queries by using terms similar to the overall query concept instead of individual words, thus even allowing to perform query reduction.

Work is clearly needed for the case of shorter queries, which, as mentioned in the section about adhoc retrieval, is the usual case for a lot of applications. The future direction of work in this area will likely be the incorporation of more linguistic resources.

Merging seems to have done well. Problems we have encountered include when there are only a few score pairs available for the linear regression because only few documents of the result lists have been aligned. This can lead to an instability of the process. The method also seems to tend to prefer one run over the other for top ranked documents, giving a somewhat unbalanced mix at the top of the merged result list. We intend to look further into this effect.

run	above median	on median	below median	avg. prec
EIT98ful	17	0	11	0.2767
EIT98des	8	2	18	0.1962
EIT98tit	7	2	19	0.1841

Table 2: Results of the cross-language runs.

References

[1] M. Braschler and P. Schäuble. Multilingual Information Retrieval Based on Document Alignment Techniques. In *Second European Conference on Research and Advanced Technology for Digital Libraries*, pages 183–197, 1998.

[2] E. Brill. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging. *Computational Linguistics*, 1995.

[3] K. W. Church. One term or two? In *ACM SIGIR Conference on R&D in Information Retrieval*, pages 310–318, 1995.

[4] B. Mateev, E. Munteanu, P. Sheridan, M. Wechsler, and P. Schäuble. ETH TREC-6: Routing, Chinese, Cross-Language and Spoken Document Retrieval. In *Proceedings of the sixth Text REtrieval Conference (TREC-6)*, 1997.

[5] S. E. Robertson. The Probability Ranking Principle in IR. *Journal of Documentation*, 33(4):294–304, 1977.

[6] P. Sheridan, M. Braschler, and P. Schäuble. Cross-Language Information Retrieval in a Multilingual Legal Domain. In *The First European Conference on Research and Advanced Technology for Digital Libraries*, pages 253–268, 1997.

[7] A. Singhal, C. Buckley, and M. Mitra. Pivoted Document Length Normalization . In *ACM SIGIR Conference on R&D in Information Retrieval*, pages 21–29, 1996.

[8] E. M. Voorhees and D. Harman. Overview of the Sixth Text REtrieval Conference (TREC-6). In *Proceedings of the sixth Text REtrieval Conference (TREC-6)*, 1997.

TNO TREC7 site report: SDR and filtering

Rudie Ekkelenkamp, Wessel Kraaij* and David van Leeuwen***

Contact Information

*) TNO-TPD (Instute for Applied Physics, department of Multimedia Technology)
Stieltjesweg 1, 2600 AD Delft
The Netherlands
**) TNO-HFRI (Human Factors Research Institute)
Kampweg 5, 2769 DE Soesterberg
The Netherlands
email: ekkelen@tpd.tno.nl
kraaij@tpd.tno.nl
vanleeuwen@tm.tno.nl
WWW: http://www.tpd.tno.nl/TPD/smartsite304.html (TNO-TPD MMT)

1. Introduction

This paper reports about experiments in the CLIR and filtering track, carried out at TNO-TPD and TNO-TM. TNO-TPD is also a member of the TwentyOne consortium and as such participated in the AdHoc task and the CLIR track. These experiments are discussed in a separate paper (cf. [Hiemstra and Kraaij98]) elsewhere in this volume.

2. SDR track

The TNO spoken document retrieval system is based on the ABBOT Large Vocabulary Continuous Speech Recognition (LVCSR) system [Renals 1998] developed by Cambridge University, Sheffield University and SoftSound, and uses word spotting, the TNO Vector Space Engine and fuzzy matching based on phoneme trigrams for indexing and retrieval. We participated in full SDR mode and experimented with several approaches

1. Fuzzy matching on a phoneme representation of the database.
2. Phone lattice based word spotting.
3. A hybrid approach were the fuzzy matching method acts as a first step to constrain the selection of input documents for the wordspotter. The idea is that there is a very efficient but rather imprecise first step and a relatively inefficient but precise second step.

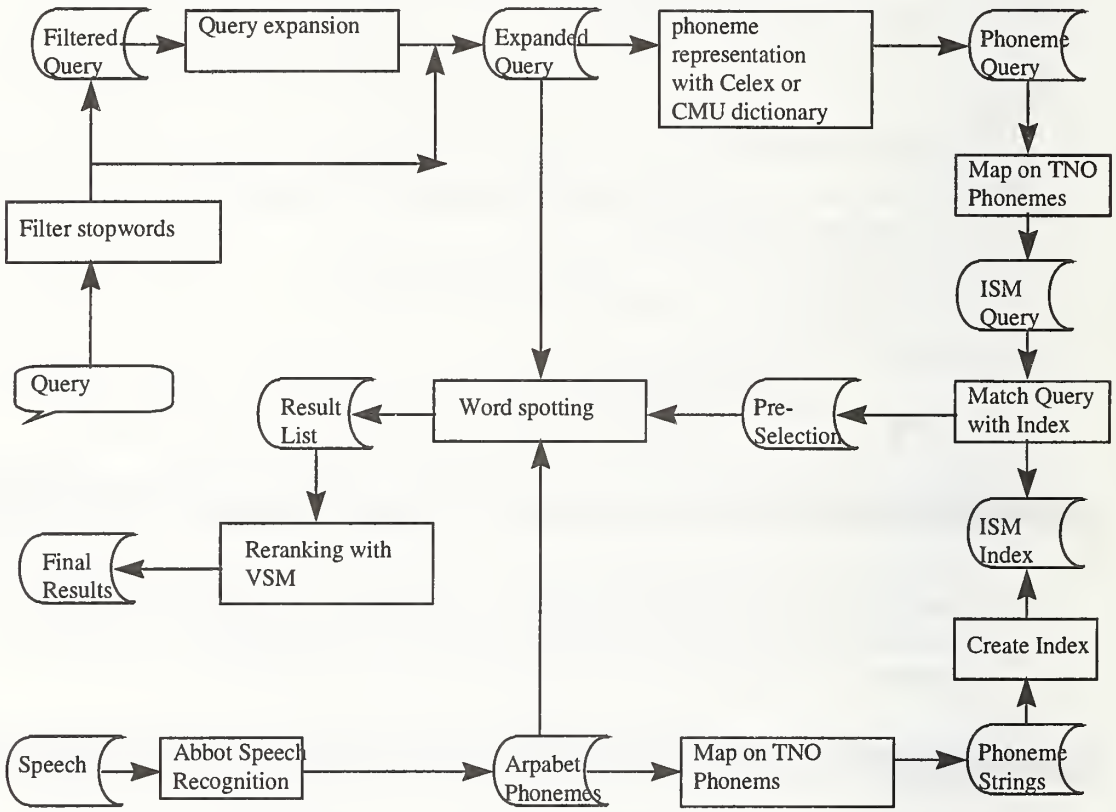
The advantage of phoneme based approaches is that they do not restrict vocabulary. This is quite important for non English languages with a rich morphology and productive compounding. Also in a News domain, proper nouns are quite important. In this paper we only

discuss the results of an approach based on methods 1 and 2, results of approaches 3 will be discussed elsewhere. Similar phone based experiments have been carried out at ETH [Wechsler1998] , University of Cambridge [James1995][Jones1996] and Dublin City University[Smeaton 1998].

The acoustical models needed to carry out phone recognition and word spotting were kindly provided by Tony Robinson from the University of Cambridge.

2.1 System

The following figure shows the architecture of the TNO SDR with the different approaches as described in the following sections



2.1.1 Fuzzy matching on phoneme transcripts

Abbot is configured as a phone recogniser (instead of a continuous word recogniser), in order to generate phone [1] transcripts of the speech documents. These are in turn converted to phoneme strings by segmenting the phones on pause symbols and mapping the phone symbols onto the characters a-z and A-Z. The phoneme strings are input for a fuzzy index based on phoneme trigrams (ISM index). For retrieval a fuzzy match is carried out between a phoneme representation of the query and the phoneme tirgram index resulting in the top N documents which contain phrases similar to the query. The phonetic representation of the topic is determined by using the Carnegie Mellon Pronouncing Dictionary [CMU, 1995]. Out Of Vocabulary (OOV) words have been ignored.

[1] A *phone* is an acoustical realisation of a sound. A *phoneme* is a conceptual representation of a sound. There can be several phone realizations for a single phoneme in a language, for instance the 't' in 'top' is aspirated, while the one in 'stop' is not.

2.1.2 Word spotting with *tf.idf* term weighting

Off-line processing

First, Abbot generates phone lattices, by reducing the acoustical input to posterior probability vectors of all phones in the phoneset, of each 16 ms time frame. These lattices can be used to do both phone recognition (see 2.1.1) and on-line word spotting.

On-line processing

For word spotting, a phonemic representation of all query words is made. The words are mixed with simple phones in a finite state grammar, and the query words are spotted in the phone lattices using the finite state grammar decoder of Abbot. This is effectively a linear search.

After word spotting all documents will be matched with a vector space model and ranked by similarities of the match. This approach has been used for submitting the S1 task.

2.2 Results

2.2.1 Official runs

We used a single strategy for the R1, B1 and B2 tasks. A vector space index has been built on the documents and the topics have been matched with index. The weighting scheme used is okapi9, as used in the PRISE engine from NIST.

okapi9 defines the *tf* component as: $tf/(tf + {}^2log(1.0 + (doclen / avg\_doclen)))$

This resulted in the following average precision values for the tasks R1, B1 and B2.

Run Type	*AVP*
R1: Reference Retrieval using human-generated "perfect" transcripts	0.3970
B1: Baseline Retrieval using medium error (35% WER) recognizer transcripts	0.3533
B2: Baseline Retrieval using high error (50%WER) recognizer transcripts	0.2833

For the S1 run we submitted a run based on the method described in 2.1.2.

S1: Full SDR based on wordspotting	0.0436

2.2.2 Unofficial run

After receiving the relevance judgements some unofficial runs have been done for the S1 task. It turned out that there were some major errors in the system. Some of these errors have been solved now (cf. 2.3) and the best run for the S1 task using the word spotting approach has an average precision of 0.1219. This run is based on the new Twente term weighting scheme (cf. [Hiemstra and Kraaij98] this volume and [Hiemstra98]).

2.3 Discussion

The baseline runs show that the average precision decreases steadily with increased word error rate. Still with a 50% WER the performance is still quite reasonable. The S1 results were quite disappointing we have identified a series of possible causes. First of all, due to lack of time no phoneme or phone lattice transcript of the training set was available for the S1 task. To be able to evaluate the runs the results for the R1 run were used as relevance judgements. It turned out

to be very hard to tune the system with these judgements. Two other problems were related to document length normalisation and inactive term weighting.

Post-hoc analysis of our S1 run revealed some problems:

Document length of word spotted document

The ranking of the documents was suboptimal because we didn't know the document length of the spoken documents. In the official S1 run we used the number of spotted words as document length. This turned out a bad measure for the document length. In our unofficial run we used the length of the phonetic representation of the document. This dramatically improved the performance.

False alarms for small words

Another big problem is the word spotting for small words since many false alarms have been generated. For example: the word "gun" has been spotted 14.000 times while in the transcriptions it only occurs about 100 times. This degraded the performance dramatically. In the future the confidence value of the spotted word should be taken into account to be able to tune for small and large words.

OOV query terms

In the Dutch version of the word spotter a rule base text to phoneme converter is used to transform queries into their phonetic representation. Unfortunately no text to phoneme converter was available for English, so the CMU dictionary (0.4 version) has been used. Unfortunately some topic words haven't been found in the dictionary among which some very relevant words like: paparazzi, Montserrat and US. Since these words haven't been spotted they will never be retrieved very well.

There is an important difference in the consequences of OOV's in the conventional word recognition based retrieval and the word spotting based retrieval. For word spotting, only phone representations of OOV query words need to be generated on-line after the query has been made. A fast word spotting search can then be performed.

A more elaborate analysis of the SDR experimental result can be found in [Kraaij 1998].

3. Filtering track: Adaptive Filtering

Because we did not participate in the filtering task in previous editions of TREC , we decided to use proven techniques. We chose the Adaptive Filtering subtask because we considered it a realistic task, close to real-world applications. Because the literature on these kind of applications is very scarce, the task to build a system based on Rocchio with a dynamic training procedure turned out to be very challenging .

We built an adaptive filtering system which is initially based on Rocchio relevance feedback, we intend to migrate to rule based classifiers at a later stage. For every topic a profile (binary classifier) is built consisting of a weighted term vector, threshold function and similarities of the last N relevant and non-relevant documents that have been positively classified by the profile. Initially a profile is filled with a term frequency vector that will be weighted using a *tfidf* scheme. Collection frequencies have been intialized by taking statistics from the LA Times corpus. During the filtering process every incoming document is transformed into a weighted term frequency document vector. Every profile vector is matched with the document vector and will result in a similarity using the cosine measure. If the similarity is

larger than the threshold of the profile, the document will be assigned to the profile. If the document is relevant, the profile vector is adapted using Rocchio relevance feedback [Rocchio 1971]. For all documents that have been assigned to a profile, the similarities of the match are stored in the profile. If a relevant document is assigned to a profile, the threshold of the profile is adapted using the midpoint of the averages of non-relevant similarities and relevant similarities.

3.1 Overview of the system

3.1.1 Initializing the adaptive filtering system

To determine statistics about document frequencies the LA Times 1988 Corpus has been used. The frequencies are used in weighing the profile vectors and document vectors. All terms from the corpus have been stemmed and stop words have been removed.

3.1.2 Creating the profiles

For every topic a profile is created as follows:

Stop words are removed from the topic text and the terms from the topic are stemmed using Porter. The resulting text is converted into a term frequency vector that is weighed using the following *tfidf* variant:

$^{2}\log(tf\_ij+1.0) * idf / {}^{2}\log(doclength)$ (1)

The threshold of the profile is initially set at 0.4 and the Rocchio parameters are initialized at 2 and 4.

3.1.3 Creating a document vector

When a document arrives for filtering, it is first converted into a term frequency document vector. Then the weights are determined based according to weighting function (1). The statistics about term frequencies and document frequencies are updated with the new document. The resulting weighted document vector will be matched with the profile vector.

3.1.4 Matching the document with the profiles

After a weighted document vector has been created, every filter profile is matched with the document to determine the similarity of the document to the profile.

For the profile document similarity we took the cosine measure. If the similarity is larger than the threshold of the filter profile, the document is assigned to the associated topic.

3.1.5 Updating the profile

Once a document is assigned to a topic, the relevance judgements can be used to update the profile of the topic.

First will be determined whether the assigned document was relevant to the profile. If so, the similarity will be stored in a history list of the last N similarities of documents that have been assigned to the document. For a relevant document the profile vector will also be updated using Rocchio relevance feedback:

Updating a filter profile vector V with a document vector D that is relevant using Rocchio can be done as follows:

$V_{new} = \alpha * V_{old} + \beta * D$, where α and β are to be defined.

If the weighted elements of the filter profile vector V get too large after updating they are normalized by dividing them by a constant value. This is needed to be able to use a constant threshold during the filtering process.

Finally the profile threshold is adapted using the similarities of the N previously assigned relevant and non-relevant documents. The average of the N relevant similarities and the average of the N non-relevant similarities will be calculated and the new threshold in the middle of these averages.

3.2 Results

Using the previously described system the following results have been obtained. Two runs have been submitted:

TNOAF102 and TNOAF103 (preferred run).

For all runs the Rocchio parameters have been set to 2 (for α) and 4 (for β); no negative relevance feedback has been used. For TNOAF102 the initial threshold has been set to 0.35 and for TNOAF103 the threshold has been set to 0.40. There has been no tuning for F1 or F3 so these runs have been submitted for both utilities.

After training with the AP 1988 and AP 1989 corpora it turned out that the TNOAF103 run performed best. The following table shows a summary of the evaluation, each cell lists first the number of profiles that were 'silent' (i.e. did not retrieve any document), then the number of topics above and below median respectively.

TNOAF103	AP88	AP89	AP90
F1	26/12/12	26/9/15	25/10/15
F3	26/10/14	26/6/18	25/9/16

Tabel 1: Topicset breakdown figures for F1 and F3: 'silent'/above median/below median

3.3 Discussion

Compared to the track medians the TNOAF103 run scores slightly below median, which is promising taking into account that this is our first filtering application. Biggest problem are the 'silent' profiles, most of which should have found relevant documents. Apart from these topics, there is a considerable number of profiles where the run scores better than median performance. Preliminary conclusion: the approach does work, but it has to be tuned. Tuning will involve a careful examination of a selection of characteristic topics. Because averaging the F1 utility over the topicset is pointless we mention a few observations based on a first glance at the track results:

- About 10-15 topics perform better than median (cf. table)
- Most (25) topics stay 'silent'. The starting threshold is probably too high for these topics.
- Other topics score extremely bad, the starting threshold is probably too low for these topics.
- Experiments with other threshold setting did not yield global improvement.

We can conclude that a uniform starting threshold is ineffective. We intend to do a more detailed investigation to solve this problem and to find other dominant factors.

4. Conclusions

We have succeeded in building laboratory versions of an application for Spoken Document Retrieval based on phone recognition and a system for adaptive filtering. The initial results revealed a number of errors, some of which have already been corrected, resulting in big improvements. As such, the TREC6 evaluation testbed will be used to test and validate improved version of our applications. The unofficial corrected SDR runs have already shown that phone based retrieval is a feasible and scaleable approach. For filtering we intend to extend our work in two directions. We expect that selection of better input features, e.g. phrases, semantic labels etc. will improve the results of the system. Secondly we have planned experiments with rule based classifiers which are less sensitive to the threshold problem.

5. Acknowledgements

We thank Tony Robinson from the University of Cambridge for providing us with the acoustical models for American English. We also thank Djoerd Hiemstra from the University of Twente and Joop van Gent (TNO-TPD) for fruitful discussions on thresholding and filtering in general.

6. References

[CMU, 1995] Carnegie Mellon Pronouncing Dictionary (cmudict.0.4, 1995). Http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

[Hiemstra 1998a] Hiemstra, D , *A Linguistically Motivated Probabilistic Model of Information Retrieval*, Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries (ECDL2),Crete,1998.

[Hiemstra 1998b] Hiemstra, D. and W. Kraaij, *TREC working notes: Twenty-One in ad-hoc and CLIR,* this volume, 1998.

[James 1995] David James, *The application of Classical Information Retrieval Techniques to Spoken Documents,* Thesis, University of Cambridge, 1995.

[Jones 1996] Jones, Gareth J.F., J.T.Foote, K. Sparck-Jones and S. Young, *Retrieving Spoken Documents by Combining Multiple Index Sources,* Proceedings of ACM-SIGIR 1996, Zürich.

[Kraaij 1998] Kraaij, W., van Gent, J., Ekkelenkamp, R., van Leeuwen, D. *Phoneme based spoken document retrieval* In: D. Hiemstra, F.M.G. de Jong, K. Netter (eds.), Language Technology in Multimedia Information Retrieval. Proceedings Twente workshop on Language Technology (TWLT14), pp. 141-153, Enschede, 1998.

[NIST 1998] The ZPRISE 1.0 Home page: www-nlpir.nist.gov/~over/zp2.

[Rocchio 1971] Rocchio, J. J. *Relevance Feedback in Information Retrieval* , In G. Salton (ed.), *The SMART Retrieval System*, Englewood Cliffs, N.J, Prentice Hall, 1971.

[Renals 1998] A. J. Robinson, M. Hochberg and S. Renals, *"The use of recurrent networks in continuous speech recognition".* In C. H. Lee, K. K. Paliwal, and F. K. Soong, eds.,

Automatic Speech and Speaker recognition---Advanced Topics, chapter 10, pages 233--258. Kluwer Academic Publishers, 1996.

[Smeaton 1998] Smeaton, A. F., M. Morony, G. Quinn and R. Scaife, *Taiscéalái: Information Retrieval from an Archive of Spoken Radio News*, Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries (ECDL2),Crete,1998.

[Wechsler 1998] Wechsler. M., E. Munteanu and P. Schäuble, *New Techniques for Open-Vocabulary Spoken Document Retrieval,* Proceedings of ACM-SIGIR 1998, Melbourne.

Manual Queries and Machine Translation in Cross-language Retrieval and Interactive Retrieval with Cheshire II at TREC-7

Fredric C. Gey, Hailing Jiang and Aitao Chen
UC Data Archive & Technical Assistance (UC DATA)
gey@ucdata.berkeley.edu, hjiang1@sims.berkeley.edu, aitao@sims.berkeley.edu

Ray R. Larson
School of Information Management and Systems
ray@sherlock.berkeley.edu
University of California at Berkeley, CA 94720

Abstract

For TREC-7, the Berkeley ad-hoc experiments explored more phrase discovery in topics and documents. We utilized Boolean retrieval combined with probabilistic ranking for 17 topics in ad-hoc manual entry. Our cross-language experiments tested 3 different widely available machine translation software packages. For language pairs (e.g. German to French) for which no direct machine translation was available we made use of English as a universal intermediate language. For CLIR we also manually reformulated the English topics before doing machine translation, and this elicited a significant performance increase for both quad language retrieval and for English against English and French documents. In our Interactive Track entry eight searchers conducted eight searches each, half on the Cheshire II system and the other half on the Zprise system, for a total of 64 searches. Questionnaires were administered to gather information about basic demographic and searching experience, about each search, about each of the systems, and finally, about the user's perceptions of the systems.

1 Introduction

Berkeley's participation in the TREC conferences has been used as a testing ground for algorithms for probabilistic document retrieval. Probabilistic document retrieval attempts to place the ranking of documents in response to a user's information need (generally expressed as a textual description in natural language) on a sound theoretical basis. Bayesian inference is applied to develop predictive equations for probability relevance where training data is available from past queries and document collections. Berkeley's particular approach has been to use the technique of logistic regression. Logistic regression has by now become a standard technique in the discipline of epidemiology for discovering the degree to which causal factors result in disease incidence [9]. In document retrieval the problem is turned around, and one wishes to predict the incidence of a rare disease called 'relevance' given the evidence of occurrence of query words and their statistical attributes in documents.

In TREC-2 [3] Berkeley introduced a formula for ad-hoc retrieval which has produced consistently good retrieval results in TREC-2 and subsequent TREC conferences TREC-4 through

TREC-6. The logodds of relevance of document D to query Q is given by

$$\log O(R|D,Q) = -3.51 + \frac{1}{\sqrt{N}+1}\Phi + 0.0929 * N \tag{1}$$

$$\Phi = 37.4 \sum_{i=1}^{N} \frac{qtf_i}{ql+35} + 0.330 \sum_{i=1}^{N} \log \frac{dtf_i}{dl+80} - 0.1937 \sum_{i=1}^{N} \log \frac{ctf_i}{cf} \tag{2}$$

where

N	is the number of terms common to query and document,
qtf_i	is the occurrence frequency within a query of the ith match term,
dtf_i	is the occurrence frequency within a document of the ith match term,
ctf_i	is the occurrence frequency in a collection of the ith match term,
ql	is query length (number of terms in a query),
dl	is document length (number of terms in a document), and
cf	is collection length, i.e. the number of occurrences of all terms in a test collection.

The summation in equation (2) is carried out over all the terms common to query and document. This formula has also been used, with success, in document retrieval with Chinese and Spanish queries and document collections of the past few TREC conferences. In TREC-6, we utilized this identical formula for German queries against German documents in the cross-language track for TREC-6. In TREC-7 this was the formula also used for all cross-language runs.

2 Ad-hoc retrieval

In TREC-6 [7], Berkeley introduced a variation of the formula which explicitly separated the evidence supported by phrases from the evidence supported by single terms. Phrases were chosen using a technique from computational linguistics, computation of the Mutual Information (MI) Measure which showed whether two words occurred together more than randomly. However, in the Berkeley TREC-6 experiments and subsequent experiments showed no discernible advantage to separability of phrases.

For TREC-7 Berkeley experimented with different stemming and phrase discovery approaches which fall short of full NLP tagging of phrases, including modification of the MI measure to be used after stemming and use of the WordNet stemmer. For example with TREC-7 topic 379 "mainstreaming" the Lovins-style stemmer of SMART-11 truncates to the fairly common term "mainstream," while the WordNet stemmer leaves the term as a whole. This term was used in our manual submission and the resulting precision for that query moved from 0.0244 for the fully automatic run (using the SMART-11 stemmer) to 0.3658 using a Boolean query (described below) and the WordNet stemmer. In the large, however, our experiments showed no significant advantage of one stemmer over another.

One failure of phrase discovery directly derived from our decision to abandon phrase discovery before stop word processing (which we had done in TREC-6). Phrase discovery before stop word processing required us to maintain a file of a large number of bigrams (word pairs) which was too big for our system to maintain efficiently. However, for the topic 368 "in-vitro fertilization" the phrase "in-vitro" can't be found because "in" is a stop word. Another failure of phrase discovery occurred in topic 394 "home schooling" – the words 'home' and 'school' are very common and hence an MI measure does not discover this term, whereas Natural Language Processing of this topic would surely uncover this crucial term. Our best performance on this query had overall precision 0.0538.

2.1 Boolean queries

Past research by Hearst [8] and Cormack et al. at Waterloo [6] has indicated that a carefully constructed Boolean query can be used to weed out irrelevant documents and thereby increase the precision of other selected documents. Berkeley decided to experiment with this approach in a limited way. Seventeen topics were given a Boolean formulation (actually many more were experimented with, but for these seventeen it seemed that an improvement might be obtained over automatic full text and manual queries.

In almost all cases (14 of the 17 topics), an improvement resulted, but a spectacular improvement occurred for three topics. First, for topic 351 "Falkland petroleum exploration" our title and narrative runs (Brkly24, Brkly25) had precision 0.2982 and 0.3137, whereas the Boolean query (in prefix form)

(AND (OR falklands falkland_islands) (OR Britain UK Argentina) (OR oil petroleum))

achieved a precision of 0.8784, best TREC-7 overall run for that query. For topic 352 "British Chunnel Impact" the Brkly 24 and Brkly25 runs were an abysmal 0.0379 and 0.0097 respectively while the Boolean query

(AND (OR British Britain English) (OR chunnel (AND channel tunnel)))

obtained a precision of 0.3112. Finally for the above mentioned topic 379 "Mainstreaming" we formulated the following Boolean query:

(OR (AND mainstreaming education) (AND mainstream schools) (AND handicapped schools))

to obtain the precision of 0.3658.

Boolean queries, of course, return an unranked set of documents, almost always fewer than the 1,000 documents required by TREC for ranking systems. So how should one rank the set of documents retrieved by a Boolean query, and how should one augment a retrieved set size less than 1,000 documents? Berkeley's approach was to use its standard logistic regression ranking algorithm for the Boolean query's document set, and to make a separate run of all manually reformulated queries ranked using logistic regression, and then to merge the two by adding the value of 1 to all documents in Boolean set retrieved. Then we have the problem of the same document appearing twice in the ranked set with different estimated probability of relevance. This was resolved by removing duplicates from the Boolean retrieved set before ranking it. An alternative would be to remove the lower ranked duplicate, but we choose not to do this.

3 Cross-language Retrieval Experiments

We created one index file from TREC-7 CLIR collections consisting of documents in English, French, German and Italian. The English words are stemmed but not the French, German and Italian words. For English, we used the SMART stemmer and a list of some 600 stopwords. We constructed a French stopword list by combining the French translation of the English stopwords using SYSTRAN [13] and the top 200 French words that most frequently occur in the French document collection. The German stopword list and the Italian stopword list were constructed in the same manner.

We submitted four cross-language retrieval runs using queries in English, French, German, and Italian against documents in all four languages. Our approach to the CLIR task is to translate the queries in the source language to other languages that are present in the collection using machine

translation software. The copy of the Globalink [2] we used is capable of translating English to French, German and Italian and vice versa; however, the translation among French, German adn Italian is not supported. For the English queries, we directly translated them into French, German and Italian using the Gloablink machine translation software. But for the French, German and Italian queries, we had to use Enlgish as a universal intermediate language. For example, the French queries were translated into English using Globalink; then the English translations of the original French queries were translated into German and Italian using Globalink again. The process of translating French queries into English, German and Italian is illustrated in Figure 2. For each set of queries in a source language, we have generated three set of queries in the other three query languages. For each set of queries, the translations and the source queries were combined to produce a set of multilingual queries. The pooled multilingual queries were run against the document collection consisting of documents in four languages. The final results for each run consists of the top-ranked 1000 documents for each pooled query. The translation and retrieval process of using English queries as the source queries is illustrated in Figure 1. The results of our four official runs are presented in Table 1. For the BKYCL7ME run, the English queries were

Run ID	Category	Query Language	Document Languages	Average Precision	Relevant Retrieved	No. >= Median	No. < Median
BKYCL7ME	Manual	English	E,F,G,I	0.3390	2648	23	5
BKYCL7AF	Automatic	French	E,F,G,I	0.2369	2405	12	16
BKYCL7AG	Automatic	German	E,F,G,I	0.2406	2482	12	16
BKYCL7AI	Automatic	Italian	E,F,G,I	0.2184	2344	12	16

Table 1: Results of four official runs.

manually reformulated before they were translated into other languages.

After the relevance judgments for the cross-language retrieval were made available, we performed two additional runs using English queries against French and English documents. The results for those two runs are shown in Table 2. Our manual run of English queries against English and

Run ID	Category	Query Language	Document Languages	Average Precision	Relevant Retrieved	No. >= Median	No. < Median
BKYCL7MEF	Manual	English	E,F	0.4185	2106	27	1
BKYCL7AEF	Automatic	English	E,F	0.3261	2007	23	5

Table 2: Results of English queries against English and French collections.

French documents performed substantially better than the automatic run. We also evaluated three machine translation software packages—SYSTRAN, Globalink, and EasyTranslator [1]—on the TREC-7 CLIR test collection. The average precision values over a set of 28 queries are shown in Table 3

The Globalink translations show that Globalink leaves new words (i.e., words unknown to the translation system) in the source text unchanged. The term mismatch problem arises when the spellings of the equivalents of a word are different and the word is left untranslated. For example, in topic 26, the German equivalent of the proper name *Létschberg* in English is *Lötschberg* and the Italian equivalent is *Lütschberg*; the French equivalent is the same as the English one. Because the same proper name has different spellings in English, German and Italian, we believe that the failure of properly translating the proper name in one language into its equivalents in other

	English (Manual)	English	French	German	Italian
SYSTRAN	0.3316	0.2615	0.2318	0.2102	0.1924
Globalink	0.3390	0.2602	0.2369	0.2406	0.2184
EasyTranslator	0.3072	0.2302	0.1795	0.1961	

Table 3: Comparison of three machine translation systems in cross-language retrieval.

languages would result in missing many of the relevant documents in the multilingual collection.

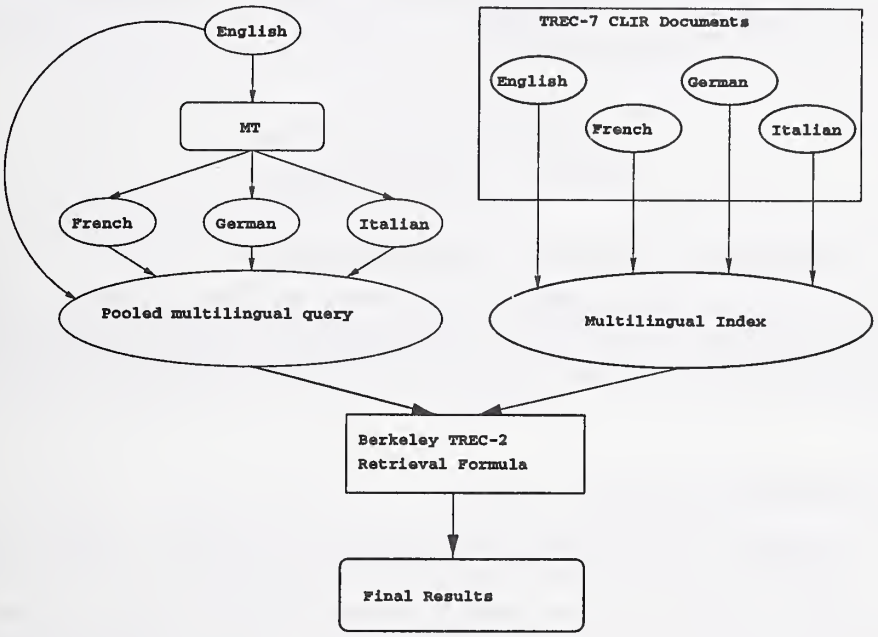


Figure 1: TREC-7 CLIR.

4 Interactive Probabilistic Retrieval: Cheshire II at TREC 7

This section briefly discusses the UC Berkeley entry in the TREC7 Interactive Track. In this year's study eight searchers conducted eight searches each, half on the Cheshire II system[11] and the other half on the Zprise system, for a total of 64 searches. Questionnaires were administered to gather information about basic demographic and searching experience, about each search, about each of the systems, and finally, about the user's perceptions of the systems. This section will briefly describe the systems used in the study and how they differ in design goals and implementation. The results of the interactive track evaluations and the information derived from the questionnaires are then discussed and future improvements to the Cheshire II system are considered. A more detailed version of the discussion in this section is available as http://sherlock.berkeley.edu/cheshire_trec7.pdf.

The primary goals of UC Berkeley entry in the TREC-7 Interactive track were to 1) attempt to replicate our entry in the TREC-6 Interactive track[10] with a larger number of participants (searchers), and to see if there were substantial differences in the ranking of the systems between last year and this year, and 2) to follow the complete TREC-7 Interactive track protocol to obtain further information than obtained in TREC-7 via the standard questionnaires filled in by all searchers on all systems. We are hoping to develop a baseline that can be used to evaluate changes and additions

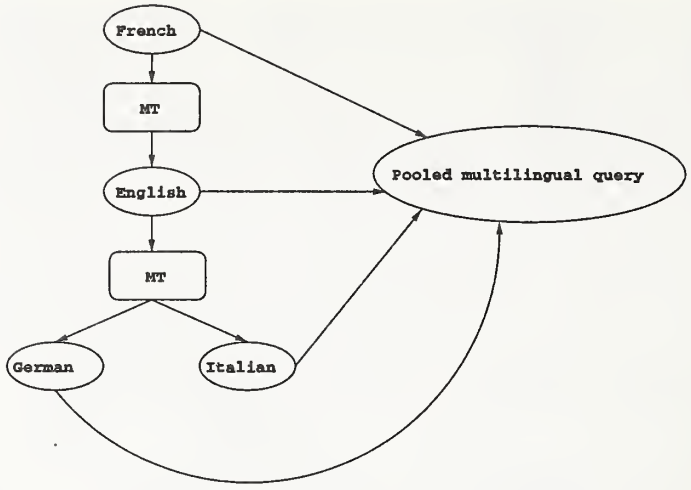


Figure 2: Query Generation.

to the systems (primarily to the Cheshire II system) in the future.

In TREC-7 we used virtually identical implementations of the Cheshire II system and the ZPRISE system as those used in TREC-6. The database and indexing for each system were also the same as for TREC-6. The characteristics of the Cheshire II system and ZPRISE systems are discussed below.

4.1 The Cheshire II System

The original design goals of the Cheshire II system were to develop a "next-generation" online library catalog system that would incorporate ranked retrieval based on probabilistic retrieval methods along with the Boolean retrieval expected in "second generation" online catalog systems. Much has changed since these initial goals were formulated. The Cheshire II system now finds its primary usage in full text or structured metadata collections based on SGML and XML, often as the search engine behind a variety of WWW-based "search pages" or as a Z39.50 server for particular applications.

4.1.1 The Cheshire II Search Engine

The Cheshire II search engine supports both probabilistic and Boolean searching. The design rationale and features of the Cheshire II search engine have been discussed in the TREC-6 paper [10] and will only be briefly repeated here.

The search engine functions as a Z39.50 information retrieval protocol server providing access to a set of databases. In the TREC-7 experiments the TREC Financial Times (FT) database was the only database used by participants. The system supports various methods for translating a searcher's query into the terms used in indexing the database. These methods include elimination of unused words using field-specific stopword lists, particular field-specific query-to-key conversion or "normalization" functions, standard stemming algorithms (Porter stemmer) and support for mapping database and query text words to single forms based on the WordNet dictionary and thesaurus using a adaption of the WordNet "Morphing" algorithm and exception dictionary..

The Cheshire II search engine supports both Boolean and probabilistic searching on any indexed element of the database. In probabilistic searching, a natural language query can be used to retrieve the records that are estimated to have the highest probability of being relevant given the user's query. The search engine supports a simple form of relevance feedback, where any items found

in an initial search (Boolean or probabilistic) can be selected and used as queries in a relevance feedback search.

The probabilistic retrieval algorithm used in the Cheshire II search engine is based on the *logistical regression* algorithms developed by Berkeley researchers and shown to provide excellent full-text retrieval performance in previous TREC evaluations [5, 3, 4]. Formally, the probability of relevance given a particular query and a particular record in the database $P(R \mid Q, D)$ is calculated and the records are presented to the user ranked in order of decreasing values of that probability. In the Cheshire II system $P(R \mid Q, D)$ is calculated as the "log odds" of relevance $\log O(R \mid Q, D)$, where for any events A and B the odds $O(A \mid B)$ is a simple transformation of the probabilities $\frac{P(A|B)}{P(\overline{A}|B)}$. The Logistic Regression method provides estimates for a set of coefficients, c_i, associated with a set of S statistics, X_i, derived from the query and database, such that

$$\log O(R \mid Q, D) \approx c_0 \sum_{i=1}^{S} c_i X_i \tag{3}$$

where c_0 is the intercept term of the regression.

For the set of M *terms* (i.e., words, stems or phrases) that occur in both a particular query and a given document.

The regression equation and coefficients used in the TREC-7 Interactive Track are the same as used in our TREC-6 entry. These are based on the TREC-3 Adhoc entry from Berkeley [4] where the coefficients were estimated using relevance judgements from the TIPSTER test collection:

The basic elements are:

$X_1 = \frac{1}{M} \sum_{j=1}^{M} logQAF_{t_j}$. This is the log of the absolute frequency of occurrence for term t_j in the query averaged over the M terms in common between the query and the document. The coefficient c_1 used in the current version of the Cheshire II system is 1.269.

$X_2 = \sqrt{QL}$. This is square root of the query length (i.e., the number of terms in the query disregarding stopwords). The c_2 coefficient used is -0.310.

$X_3 = \frac{1}{M} \sum_{j=1}^{M} logDAF_{t_j}$. This is is the log of the absolute frequency of occurrence for term t_j in the document averaged over the M common terms. The c_3 coefficient used is 0.679.

$X_4 = \sqrt{DL}$. This is square root of the document length. In Cheshire II the raw size of the document in bytes is used for the document length. The c_4 coefficient used is -0.0674.

$X_5 = \frac{1}{M} \sum_{j=1}^{M} logIDF_{t_j}$. This is is the log of the *inverse document frequency*(IDF) for term t_j in the document averaged over the M common terms. IDF is calculated as the total number of documents in the database, divided by the number of documents that contain term t_j The c_5 coefficient used is 0.223.

$X_6 = logM$. This is the log of the number of common terms. The c_6 coefficient used in Cheshire II is 2.01.

These coefficients and elements of the ranking algorithm have proven to be quite robust and useful across a broad range of document types.

Probabilistic searching, as noted above, requires only a natural language statement of the searcher's topic, and thus no formal query language or Boolean logic is needed for such searches. However, the Cheshire II search engine also supports complete Boolean operations on indexed elements in the database, and supports searches that combine probabilistic and Boolean elements. At present, combined probabilistic and Boolean search results are evaluated using the assumption

that the Boolean retrieved set has an estimated $P(R \mid Q_{bool}, D) = 1.0$ for each document in the set, and 0 for the rest of the collection. The final estimate for the probability of relevance used for ranking the results of a search combining Boolean and probabilistic strategies is simply:

$$P(R \mid Q, D) = P(R \mid Q_{bool}, D)P(R \mid Q_{prob}, D) \quad (4)$$

where $P(R \mid Q_{prob}, D)$ is the probability estimate from the probabilistic portion of the search, and $P(R \mid Q_{bool}, D)$ the estimate from the Boolean. This has the effect of restricting the results to those items that match the Boolean portion, with ordering based on the probabilistic portion.

Relevance feedback is implemented quite simply, as probabilistic retrieval based on extraction of content- bearing elements (such as titles, subject headings, etc.) from any items that have already been seen and selected by a user. Similarly, multiple records may be selected and submitted for feedback searching. In this case the contents of all those records are merged into a single query and submitted for searching. At the present time we do not use any methods for eliminating poor search terms from the selected records, nor special enhancements for terms common between multiple selected records [12], but we plan to experiment further with various enhancements to our relevance feedback method.

4.1.2 The Cheshire II Client Interface

The design of the Cheshire II client interface (shown with the TREC FT database in Figure 3) was driven by a number of goals:

1. to support a consistent interface to a wide variety of Z39.50 servers., and to dynamically adapt to the particular server.

2. to reduce the cognitive load on the users wishing to interact with multiple distributed information retrieval systems by providing a single interface for them all.

3. to minimize use of additional windows during users' interactions with the client in order to allow them to concentrate on formulating queries and evaluating the results, and not expend additional mental effort and time switching their focus of attention from the search interface to display clients;

4. to provide functions not immediately related to searching, such as print and e-mail facilities, to facilitate users' ability to 'take the results home'; and

5. to design a help system within the interface that would assist users not only in the mechanics of operating the Cheshire II client, but also in the more general tasks of selecting appropriate resources for searching, formulating appropriate queries, and employing various search tactics.

However, the initial design goals for the client interface made the assumption that most of the information that would be viewed in the search interface would be brief metadata records for documents, and not full text documents themselves. The ability to view full-text documents such as the FT articles used in the TREC-7 Interactive track experiments was added to the existing interface easily, but as the comments and questionnaire responses discussed below show, this was probably not an optimal implementation for the tasks posed by the experiments.

Additional functionality beyond searching and browsing has been relatively easy to implement. Functions for printing, e-mailing and saving records are all available when records are displayed, and the user has the option of acting on either the entirety of the current record display or a subset

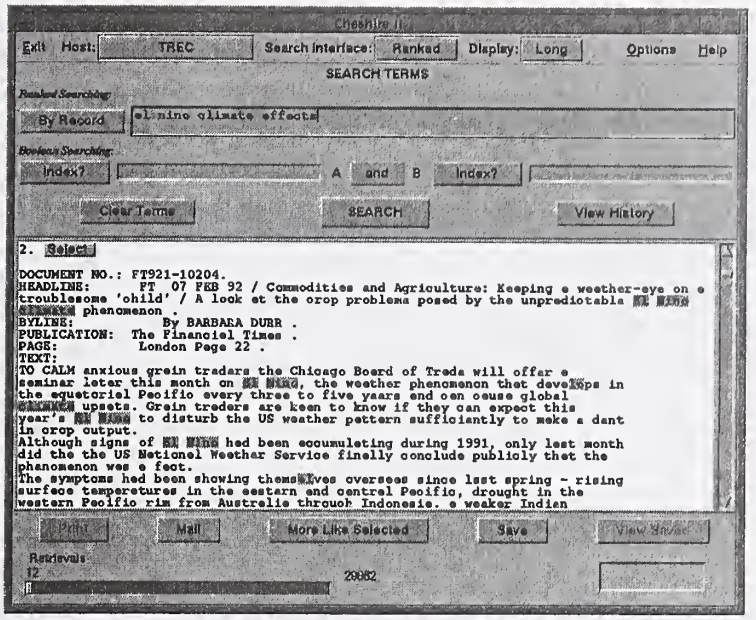


Figure 3: Cheshire II Long Display.

thereof by selecting individual records using the "select" buttons on each record (visible in Figure 3 next to the record number).

Among the changes made to the client interface for TREC was the inclusion of display formats for the FT records (as shown in Figure 3). A routine was also added to highlight query terms in the text of the document to aid searchers in scanning for relevant passages. Note that the highlighting feature doesn't necessarily catch all of the terms that contributed to the selection of the document, because only the original query terms, and not stemmed terms, are used in the highlighting. Since the highlighting is using simple string matching on the text, partial words are sometimes highlighted erroneously.

4.2 The Zprise System

The second (control) system used in the TREC-7 Interactive track at Berkeley was the Zprise system from NIST. This system was used in the same configuration and with the same database indexing setup as used for the global control system in the TREC-6 Interactive Track. Zprise, as configured for this test was limited to a total of 24 retrieved items and relevance feedback was disabled. However, the interface was set up so that it provided a very good fit for the tasks involved in the interactive track. For example, documents were viewed in full text form in a separate window from the short display (consisting primarily of title and date as well as control elements for indicating relevant documents and for moving around in the brief display(see Figure 4.

Most of our users found the ZPRISE displays simple to learn and to operate, in fact most found that the operations required to carry out the Interactive Track tasks were easier to do on the ZPRISE interface than they were on the Cheshire II interface. This was not entirely surprising, since the ZPRISE interface is designed to support TREC-like databases containing full text, while the Cheshire II interface, as noted above, was designed for brief metadata records and not with the idea of providing support for the sort of reading and selection activities that make up the user tasks in the TREC Interactive Track.

In some ways the comparison between the interfaces comes down to how well a generic interface,

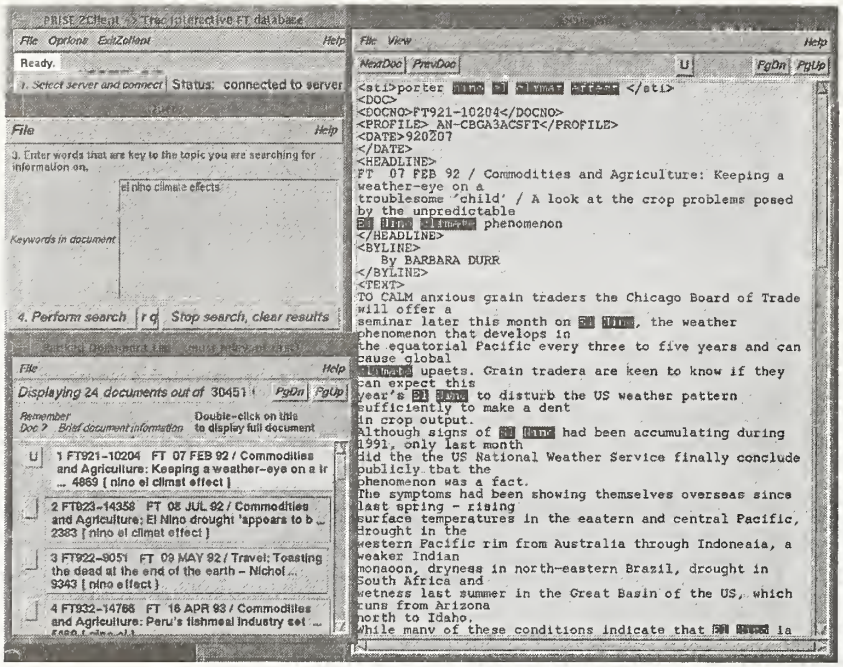


Figure 4: ZPrise Interface.

not particularly adapted to the specific task, compares to an interface tailored to that task. The underlying PRISE search engine in ZPRISE uses (apparently) a fairly standard Vector Space model search algorithm, which performs quite well given the usually brief and simple query statements that characterized most searches by the searchers participating in this year's interactive track. In the following section I will describe the results obtained from the Aspectual Precision and Recall evaluations at NIST and the results of the demographic, search and system related questionnaires filled out by the participating searchers.

4.3 TREC Interactive Track

The administration of the interactive track followed the protocols set down in the track guidelines. This mandated a minimum group of 8 participant searchers, each of whom conduct 8 searches, half on the control system (ZPRISE, identified as "Z") and half on the experimental system (Cheshire II, identified as "C").

Each searcher was asked to use the features of the respective interfaces to select as relevant those documents that they considered to relevant to one or more aspects of the specific topic. Because of some delays in obtaining the license and materials for the FA-1 Controlled Associations Test, this test was administered to the subjects independently from their actual search sessions.

The pooled results for all systems were evaluated at NIST by the TREC evaluators and "Aspectual Precision" and "Aspectual Recall" for each searcher was calculated. Table 4 shows the values for Aspectual Precision by TREC topic for all systems in the TREC Interactive Track. Table 5 shows the values for Aspectual Recall for all of the participating systems. Note that these two tables were derived from the per- search Recall and Precision figures reported by NIST. Note also that in these tables all of the system usages were combined in the calculations, therefore "ok_noRF" which was used in two separate experiments has the results from both experiments combined. The two Berkeley systems ("C" and "Z", the Cheshire II system and ZPRISE systems respectively) are shown in boldface in Tables 4 and 5. The control system "Z" performed marginally better than the

experimental system in terms of the Aspectual Precision. It is also interesting to note that virtually identical performance was achieved by the "ZP" system from NMSU and the "zp_noRF" from the Okapi Group, I believe that both of these systems, like "Z", are unmodified ZPRISE systems.

The Cheshire II system also did not perform as well the control ZPRISE system in these experiments. This fact can largely be attributed to the more complex interactions required to perform the search tasks on the generic Cheshire II interface than on the ZPRISE system. In addition, there were some specific search failures due to misspelling (one searcher had 0 Precision and Recall for one search due to this).

Analysis of the mean and standard deviation of precision and recall over all searches for each searcher and system showed a considerable range of performance within the searchers at Berkeley. In the following section we will examine the characteristics of the searchers as reported in the questionnaires administered during the experiments. Figure 4 summarizes the average aspectual precision and recall for each of the systems participating in the TREC-7 Interactive Track.

System	T							opic Number	Total
	352i	353i	357i	362i	365i	366i	387i	392i	
a	0.4418	0.3860	0.4870	0.5913	0.9168	0.9168	0.8125	0.6110	0.6454
b	0.7448	0.3483	0.4125	0.7500	0.9018	0.8875	0.7375	0.8310	0.7017
C	**0.7993**	**0.2145**	**0.6368**	**0.7058**	**0.9793**	**0.9375**	**0.8458**	**0.8068**	**0.7407**
clus	0.4355	0.3594	0.5514	0.6773	0.8750	0.4291	0.5938	0.6276	0.5686
irisa	0.7448	0.4333	0.5715	0.7500	0.7223	1.0000	0.9063	0.8088	0.7421
irisp	0.7333	0.1750	0.5533	0.6368	1.0000	0.9168	0.7500	0.7085	0.6842
iriss	0.6745	0.6021	0.5803	0.6194	0.8875	0.8611	0.7884	0.7303	0.7179
J24	0.6250	0.5178	0.5083	0.5568	0.9375	0.8750	0.6615	0.7315	0.6767
list	0.2811	0.1916	0.4200	0.5209	0.9584	0.3750	0.8084	0.7540	0.5387
MB	0.8831	0.4876	0.4164	0.7249	0.7524	0.9679	0.6494	0.6020	0.6855
MR	0.6342	0.4309	0.2857	0.7056	1.0000	1.0000	0.7944	0.7190	0.6962
ok_noRF	0.9018	0.4984	0.3033	0.4981	0.8840	0.8750	0.7710	0.4874	0.6524
ok_withRF	0.8578	0.5865	0.3818	0.3558	0.7520	1.0000	0.8335	0.7475	0.6893
RUINQ-G	0.5844	0.5109	0.3618	0.4628	0.8814	0.7918	0.8854	0.6228	0.6357
RUINQ-R	0.5558	0.3160	0.4391	0.6055	0.8674	0.7889	0.7689	0.6819	0.6286
Z	**0.9500**	**0.4405**	**0.5030**	**0.6043**	**1.0000**	**0.8333**	**0.9333**	**0.7093**	**0.7467**
ZP	0.5863	0.4045	0.4285	0.8928	0.9688	0.9500	0.8873	0.8493	0.7459
zp_noRF	0.8875	0.2323	0.4303	0.7640	0.8750	0.7918	0.9015	0.8250	0.7134
Grand Total	0.6733	0.4090	0.4367	0.6324	0.8928	0.8354	0.7776	0.6916	0.6689

Table 4: Aspectual Precision by System and Query

System							T opic Number		Total
	352i	353i	357i	362i	365i	366i	387i	392i	
a	0.1250	0.2728	0.3658	0.3540	0.8023	0.5355	0.4723	0.3403	0.4085
b	0.2413	0.1593	0.3080	0.2918	0.8543	0.5000	0.3888	0.2433	0.3733
C	**0.2768**	**0.1820**	**0.3080**	**0.3123**	**0.8543**	**0.5358**	**0.4443**	**0.4373**	**0.4188**
clus	0.0893	0.1024	0.2310	0.1459	0.6874	0.2144	0.1110	0.1840	0.2207
irisa	0.2858	0.1593	0.2888	0.2708	0.5208	0.2860	0.1943	0.2430	0.2811
irisp	0.2413	0.1138	0.3655	0.1873	0.7918	0.5358	0.2220	0.3403	0.3497
iriss	0.1653	0.2161	0.2791	0.2605	0.7186	0.3571	0.3193	0.2985	0.3268
J24	0.1875	0.2275	0.3080	0.1875	0.8750	0.2503	0.4445	0.5280	0.3760
list	0.0534	0.0683	0.2791	0.1354	0.6770	0.0715	0.2499	0.2846	0.2274
MB	0.3368	0.1950	0.2090	0.3274	0.6161	0.4490	0.4522	0.2401	0.3532
MR	0.3061	0.1949	0.1155	0.2737	0.7260	0.3776	0.3171	0.4603	0.3464
ok_noRF	0.3349	0.2161	0.2406	0.3020	0.8489	0.4288	0.4305	0.2675	0.3837
ok_withRF	0.4105	0.2275	0.3655	0.3123	0.8230	0.3930	0.3610	0.2848	0.3972
RUINQ-G	0.2619	0.2340	0.2909	0.2499	0.8288	0.3930	0.4723	0.2918	0.3817
RUINQ-R	0.2365	0.2427	0.2214	0.3333	0.7449	0.3651	0.4073	0.2560	0.3489
Z	**0.2233**	**0.2955**	**0.2695**	**0.2918**	**0.8230**	**0.4285**	**0.6113**	**0.4723**	**0.4269**
ZP	0.2768	0.1593	0.3465	0.1668	0.8543	0.4643	0.5555	0.4308	0.4068
zp_noRF	0.3838	0.1138	0.3080	0.1878	0.8333	0.3930	0.5000	0.4723	0.3990
Grand Total	0.2478	0.1875	0.2573	0.2592	0.7503	0.3749	0.3750	0.3238	0.3472

Table 5: Aspectual Recall by Systems and Query

4.3.1 User Characteristics

The administration of the interactive track followed the track guidelines with a single group of 8 participants. While none of the participants had used either the experimental (Cheshire II) or control (ZPRISE) systems in searching tasks, many had seen demonstrations of the experimental

system. The searchers who participated in the study were volunteers drawn from the School of Information Management and Systems at UC Berkeley (a call for participation was sent to all students and faculty at SIMS and the first 8 volunteers were scheduled for search sessions. A pre-search questionnaire asked each participant for basic demographic information and educational background, as well as their experience with various types of search systems.

All of the participants, except one undergraduate, held college degrees (One held a PhD, two others were PhD students with previous undergraduate and graduate degrees, and the remaining 4 were Masters students in the SIMS program). Two of the participants (P1 and P3) had considerable experience in online searching on other systems, the other two had very limited experience with online systems. The range of search experience with various systems varied from over 20 years to less than one year. By far, the most frequently used search systems were the Web search services and the next most frequent were online catalogs. Exploratory correlation analyses were performed on all of the variables from the presearch questionnaires, combined with the matching search and system questionnaires with the per-search and search precision and recall information from the NIST evaluators. Not surprisingly, there were very few significant correlations found in the analysis and many of those were trivial (years of search experience is positively correlated with age). Somewhat more interesting was that search experience with online systems like Dialog and BRS was also significantly correlated with search experience. It appears that most recent searchers will be gaining their experience from the WWW and possibly from online library catalogs, and will probably not have experience (or as much experience) with traditional Boolean systems such as Dialog.

4.3.2 Per Search Results

Following each search the participants were given a questionnaire asking about familiarity with the search topic, how easy it was to start and conduct the search and whether the user was satisfied with the results.

The responses from each participant are included in the WWW version of this section noted above. All searchers found the search easier to do with the ZPRISE system than with the Cheshire II system. Similarly, analysis of the average responses to the "Are you satisfied with the results" question showed that the ZPRISE system is given higher marks than the Cheshire system. Analysis was also conducted of the average responses to the question "Are you familiar with this topic?" Here the responses show that the searchers where generally less familiar with the topics searched on the Cheshire system versus those on the ZPRISE system. Correlation analysis showed, however, no significant correlation between familiarity with a topic and either the ease of searching or the satifaction with search results. Satisfaction was however fairly strongly correlated with how easy it was to do the search task (Pearson's R=0.646, prob=0.0001). Interestingly, there was no significant correlation between any of the post-search questions and either Aspectual Precision or Aspectual Recall, but the signs of these correlations indicated some interesting items for further research. For example, a slight negative correlation was indicated between Precision and Recall and the user's confidence that they had identified all of the different instances for a topic.

4.3.3 Post-System Questions

The searches were conducted in blocks of 4 questions on each system. Following the searcher's interaction with a system, a post-system questionnaire was administered. This post-system questionnaire asked each searcher questions about how easy the system was to learn, use, and understand, and permitted comments on the features of the system.

Overall, the searchers found both system very easy to learn. The Cheshire system was marked down again on the "easy to use" question. From the comments, this appeared to be related to some missing features (e.g. Boolean AND but no NOT), and several searchers mentioned the need to scroll back to the beginning of a record to select it as relevant. Others (who used ZPRISE first) mentioned a preference for having the full- text document in a separate window. With responses on a scale from 1 (difficult) to 5 (extremely easy), the average "ease of use" for Cheshire II was 3.38 and the average for ZPRISE was 4.25.

4.3.4 Exit Questionnaire

After the completion of all searches an exit questionnaire was administered to the searchers. This questionnaire asked how well the searchers understood the task, whether it was similar to other seach task, how they would rank the systems in relation to each other, and what they liked and disliked about each system.

The searchers claimed to have a very good understanding of the search task (mean was 4.25), and they found the task similar to other searching tasks (mean of 3.63). They also found the systems somewhat different (mean of 3.37). In ranking the systems, 5 out of 8 ranked ZPRISE as easier to learn to use, while 7 out of 8 chose ranked it as easier to use. Curiously the searchers were evenly split (four each) on which they liked the best. One search commented that she would prefer Cheshire "for serious research" but found ZPRISE better suited to the TREC search tasks.

5 Conclusions and Acknowledgments

In our TREC-7 experiments for the ad-hoc task and cross-language track, Berkeley utilized our probabilistic document retrieval methods for all retrieval. In the ad-hoc task we experimented with discovery of the "best" Boolean query and merged Boolean and probabilistic retrieval. This provided some spectacular successes and seemed to provided some overall improvement. In the interactive track, it was impossible to draw any firm general conclusions from our small sample of searchers and searches. But it is obvious that an interface that is well adapted to the specific search task will tend to be preferred by searchers even if the underlying system produces better overall performance in terms of Recall, and comparable performance in terms of precision. As observed at TREC-6, the overall performance of the Cheshire II system was quite good, although it was not dramatically better than the control system on average. These results, as has often been noted in previous TREC interactive evaluations, tend to be highly influenced by individual behavior and search techniques (this is apparent in the differences between the searchers on the same questions and in the same systems). What seems apparent from the results of the questionnaires coupled with the Precision and Recall measures is that a generic interface can perform quite acceptably in the TREC tasks, even if it isn't particularly liked by the users, compared to another system that is better suited to the TREC tasks, but may not be as useful in other situations.

This research was supported by the Information and Data Management Program of the National Science Foundation under grant IRI-9630765 from the Database and Expert Systems program of the Computer and Information Science and Engineering Directorate. The original development of the Cheshire II system was sponsored by a College Library Technology and Cooperation Grants Program, HEA-IIA, Research and Demonstration Grant #R197D30040 from the U.S. Department of Education. Further development work on the Cheshire II project and system was supported as part of Berkeley's NSF/NASA/ARPA Digital Library Initiative Grant #IRI-9411334.

Both projects are currently being supported in part by DARPA (Department of Defense Advanced Research Projects Agency) under research contract N66001-97-C-8541, AO-F477.

References

[1] EasyTranslator Version 2.0. Transparent Language Inc.

[2] Globalink Power Translator Version 6.02. Globalink Inc.

[3] W. S. Cooper, A. Chen, and F. C. Gey. Full Text Retrieval based on Probabilistic Equations with Coeffici ents fitted by Logistic Regression. In D. K. Harman, editor, *The Second Text REtrieval Conference (TREC-2)*, pages 57–66, March 1994.

[4] William S. Cooper, Aitao Chen, and Fredric C. Gey. Experiments in the probabilistic retrieval of full text documents. In *Text Retrieval Conference (TREC-3) Draft Conference Papers*, Gaithersburg, MD, 1994. National Institute of Standards and Technology.

[5] William S. Cooper, Fredric C. Gey, and Daniel P. Dabney. Probabilistic retrieval based on staged logistic regression. In *15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Copenhagen, Denmark, June 21-24*, pages 198–210, New York, 1992. ACM.

[6] C.R. Palmer G. V. Cormack, C.L.A. Clarke and S.S.L. To. Passage-Based Refinement (Multi-Text Experiments for TREC-6). In D. K. Harman and Ellen Voorhees, editors, *The Sixth Text REtrieval Conference (TREC-6), NIST Special Publication 500-240*, pages 303–319, August 1998.

[7] F. C. Gey and A. Chen. Phrase Discovery for English and Cross-language Retrieval at TREC-6. In D. K. Harman and Ellen Voorhees, editors, *The Sixth Text REtrieval Conference (TREC-6), NIST Special Publication 500-240*, pages 637–647, August 1998.

[8] Marti Hearst. Improving Full-Text Precision on Short Queries using Simple Constraints. In *The Fifth Annual Symposium on Document Analysis and Information Retrieval SDAIR), Las Vegas, Nevada*, April 1996.

[9] David W. Hosmer and Stanley Lemeshow. *Applied Logistic Regression.* John Wiley & Sons, New York, 1989.

[10] Ray R. Larson and Jerome McDonough. Cheshire II at TREC 6: Interactive probabilistic retrieval. In Donna Harman and Ellen Voorhees, editors, *TREC 6 Proceedings (Notebook)*, pages 405–415, Gaithersburg, MD, 1997. National Institute of Standards and Technology.

[11] Ray R. Larson, Jerome McDonough, Paul O'Leary, Lucy Kuntz, and Ralph Moon. Cheshire II: Designing a next-generation online catalog. *Journal of the American Society for Information Science*, 47(7):555–567, July 1996.

[12] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41:288–297, 1990.

[13] SYSTRAN:. *http://babelfish.altavista.digital.com/.*

TREC-7 Experiments at the University of Maryland

Douglas W. Oard
Digital Library Research Group
College of Library and Information Services
University of Maryland, College Park, MD 20742
oard@glue.umd.edu

Abstract

The University of Maryland participated in three TREC-7 tasks: ad hoc retrieval, cross-language retrieval, and spoken document retrieval. The principal focus of the work was evaluation of merging techniques for cross-language text retrieval from mixed language collections. The results show that biasing the merging strategy in favor of documents in the query language can be helpful. Ad hoc and spoken document retrieval results are also presented.

1 Introduction

The principal goal of the University of Maryland's participation in the Seventh Text REtrieval Conference (TREC-7) was to evaluate the performance of alternative merging strategies for Cross- Language Information Retrieval (CLIR) from mixed language collections. The Logos machine translation system[1] was used in a fully automatic mode for query translation, and PRISE from the National Institutes of Standards and Technology was used for all runs. We participated in the Ad Hoc task as well in order to gain experience with PRISE, and we also used PRISE for Spoken Document Retrieval (SDR) track runs. No manual processing was done, and all of our runs were submitted in the automatic category.

2 Cross-Language Information Retrieval

As typically formulated, interactive information retrieval involves at least three stages: query formulation, searching the document collection using the query to identify a set of possibly relevant documents, and selection of desirable documents by the user [1]. CLIR potentially adds complexity to each stage. The focus of our work in the CLIR track at TREC has been on fully automatic techniques that are appropriate for the middle stage, finding possibly relevant documents when the query and document may not be in the same language. At TREC-6 we compared query translation and document translation approaches, finding little difference in overall retrieval effectiveness [2]. Query translation is the more efficient of the two approaches, and that advantage is magnified when documents in several languages are present in the collection as is the case in the TREC-7 CLIR track. We have thus chosen query translation as the basis for our experiments this year.

In TREC-6 we learned that language-specific processing such as stemming can have a substantial effect on retrieval effectiveness, a lesson that others have learned before [3]. In those experiments we used Inquery version 3.1, which was capable of stemming English but not German. With long queries, we observed that indexing English translations of German documents (with stemming) gave better results than indexing the documents in German (without stemming or compound splitting). We initially believed that this gave evidence favoring document translation. After seeing the same effect on English (AP) documents, however, we now believe that the differences resulted from a failure to perform stemming or compound splitting in German.

[1] Logos Corporation, 111 Howard Boulevard, Suite 214, Mount Arlington, NJ 07856 USA

2.1 Experiment Design

The TREC-7 CLIR track requires that documents in German, French, Italian, and English be processed. Since we had reliable *a priori* knowledge of the language contained in each portion of the collection, we used that knowledge to select appropriate language-specific processing. Documents in the AP collection were treated as English, documents in the "French SDA" collection were treated as French, documents in the "Italian SDA" collection were treated as Italian, and documents in both the "German SDA" and the "NZZ" collections were treated as German.

The Logos machine translation system can translate from English to French, German, Italian and Spanish. Our queries were thus based on the English topics. We began by translating the queries from English into each other language, using the Logos system in a fully automatic mode with no application-specific additions to the lexicon or semantic rules. We then formed title queries from the words in the title field, and long queries from every topic word except SGML markup, the contents of the query number field, and the terms "Description:" and "Narrative:" that appear in every query.

PRISE includes the Porter stemmer for English, a German stemmer implemented by Martin Braschler, and a French stemmer implemented by Jacques Savoy. We did not have an Italian stemmer, and no compound splitting was performed in any language. The stopword list from Inquery version 3.1 was used in English, and degenerate stopword lists were used in the other languages ("le" in French, "die" and "dir" in German, and "du" in Italian — PRISE choked if the stopword list was empty). No stop-structure removal was performed. Separate PRISE indexes were built for each language, with the German index covering both the "German SDA" and the "NZZ" collections. Index construction required between two and four hours on a dedicated Sparc 20, depending on the number of documents in each language, and retrieval results for all 25 queries were typically computed in a few minutes (varying slightly with query length and whether stopwords were used). In our official runs we inadvertently omitted the 1989 and 1990 AP documents from the English index, and this adversely affected our results. That has been corrected in the results reported here.

Vector space text retrieval systems such as PRISE typically produce retrieval status values that lack comparability across collections, so rank-based merging generally outperforms strategies based on retrieval status values. Voorhees demonstrated that giving more weight to collections that are historically more productive can yield better results that a uniform rank-based merging strategy [5]. In TREC-6 we observed that machine translation of German queries into English achieved 56% of the average precision that was observed when English queries were used for monolingual retrieval, and we expected that a strategy which selected more documents from the English collection than from the other three collections would perform well. We thus implemented a uniform weighted merge in which the top N documents were selected (without replacement) from English every time the top document was selected (without replacement) from each of the other languages.

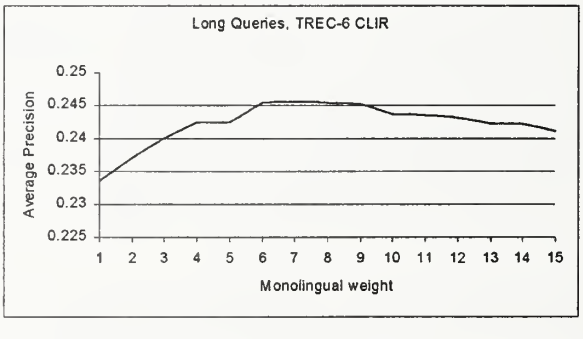


Figure 1: The effect of varying N on TREC-6 CLIR long queries.

2.2 Parameter Selection

In order to get some idea of a reasonable range for N, we tried our strategy on the TREC-6 CLIR collection. The TREC-6 document collection is a substantial subset of the TREC-7 document collection, lacking only

the Italian SDA documents. The limited pool of participating systems may, however, have limited the completeness of the TREC-6 relevance judgments in some languages, and there were some differences in the way queries were formulated in the two evaluations. In our official runs the omitted 1990 and 1991 portions of the AP collection reduced the performance of the English collection. Not surprisingly, $N = 1$ outperformed higher values of N under those conditions, so our two official TREC-7 CLIR submissions were produced with an even merging strategy ($N = 1$) on title (run umdxeot) and long (run umdxeof) queries. When we reran our experiments on the complete TREC-6 collection we found that weighted merging outperformed an even merging strategy by about 5% on long queries (at $N = 6$), but that no more than a 0.3% advantage could be achieved on title queries (at $N = 1.4$). Figure 1 illustrates the long-query results.

2.3 Results

When the TREC-7 CLIR relevance judgments became available we observed a similar advantage for strongly weighted merging, achieving an 9% improvement on long queries at the $N = 6$ parameter learned on the TREC-6 data and an 11% improvement on long queries at the *post hoc* optimum parameter value ($N = 9$). Weighted merging again produced only a modest improvement (2% at $N = 5$) on title queries. Figure 2 illustrates these results.

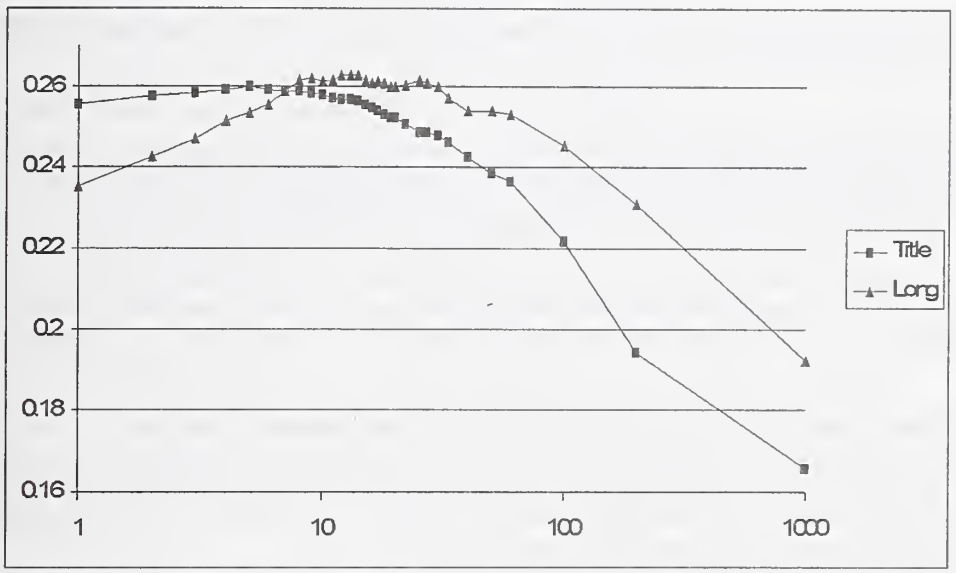


Figure 2: The effect of varying N on TREC-7 CLIR title and long queries.

We were surprised by how large the large values of N were that produced the best results for long queries and by the consistent difference between the effectiveness of weighted merging for title and long queries, so we decided to examine the monolingual performance of our system for each language pair using the TREC-7 data. Table 1 shows the uninterpolated average precision obtained when English queries were used to retrieve documents in a single language. In this case, only relevance judgments for documents in that language were considered. Some topics lack known relevant documents in some languages, so the number of queries over which the averages are calculated are shown for each language. There is some variation evident between the two query lengths in German, but no systematic differences are evident.

Table 2 shows some collection statistics. On average, nearly twice as many relevant documents are known for English as for any other language, and there are even fewer known relevant documents in the Italian collection. The average density of relevant documents is somewhat more consistent, however.

Our results suggest two factors that might be useful when selecting collection weights if a uniform merge strategy is used. The most obvious is the expected performance of each system - a monolingual system would be expected to outperform a cross-language one, for example. The second possibly useful factor is collection size, which should predict the number of relevant documents well if the collections and queries

Doc Lang	Title Queries	% of English	Long	% of English	Num of Queries
English	0.4357		0.5290		26
French	0.2827	65%	0.3420	65%	28
German	0.2265	52%	0.2311	44%	27
Italian	0.2453	56%	0.2874	54%	25

Table 1: Non-interpolated average precision with English queries for documents each language.

Doc Lang	Documents	Average Relevant	Average Density
English	242,917	60	2.5
French	141,656	35	2.5
German	251,850	33	1.3
Italian	62,359	18	2.9

Table 2: Density of known relevant documents per 10,000 documents, averaged over 28 topics.

are chosen in a way that produces similar densities of relevant documents across the collections. It is not yet clear whether the number of relevant documents is actually more important than their density, but our results suggest that a focused investigation of that issue could prove useful in this context.

A note of caution should be sounded regarding our use of the average precision measure. Our monolingual English run achieved higher precision at 5, 10, 15, 20, and 30 documents than the best merged run on both title and long queries.[2] The advantage of the merging strategy is only evident at 100, 200, 500 and 1000 documents. The average precision measure is useful because it balances precision and recall, but other measures may me more appropriate for specific applications.

3 Ad Hoc and Spoken Document Retrieval Tasks

We used our participation in the ad hoc retrieval task to become familiar with PRISE. The official run was submitted using the default term weighting strategy in PRISE, which does not do as well as the "okapi1" weights that we used for our CLIR and SDR experiments.

We are working on user interface design for information retrieval systems that provide access to large collections of recorded speech [4], and the SDR track offers an opportunity to gain additional experience with content-based retrieval using speech recognition output. Our speech recognition system was not ready in time for these runs, so we submitted results only for the baseline recognizer output. We used a modified version of PRISE for these experiments in which some changes had been made to the numerical details of retrieval status value computation, but a comparison with the original system revealed no significant differences in the ranked output. The Porter stemmer, okapi1 weights, and the Inquery stopword list were the only deviations from the default settings in the indexer. Indexing took approximately 15 minutes for each of the three runs, and batch processing of the queries was completed in under a minute per collection. The queries used were identical for each of the three runs.

4 Conclusion

We have demonstrated one useful strategy for merging retrieval results from collections in different languages. As the richness of the TREC CLIR corpus grows, we plan to exploit it to investigate more sophisticated

[2] In this case, precision values for the monolingual English runs were computed using all relevance judgments rather than those for English alone in order to produce comparable results.

strategies. We are also interested in integrating automatic language identification in order to investigate a whether applications in which the document languages cannot be reliably determined from *a priori* information will pose substantially greater challenges. The TREC CLIR corpus also provides an excellent resource for evaluating other approaches to CLIR, and we hope to use it to explore both cognate matching and corpus-based techniques.

Acknowledgments

The author is grateful to Paul Hackett and Skip Warnick for their assistance with the experiments, to NIST for their extremely helpful support as we learned to use PRISE, and to the Logos Corporation for the use of their machine translation system. This work has been supported in part by DARPA contract N6600197C8540.

References

[1] Douglas W. Oard. Serving users in many languages: Cross-language information retrieval for digital libraries. *D-Lib Magazine*, December 1997. http://www.dlib.org.

[2] Douglas W. Oard and Paul G. Hackett. Document translation for cross-language text retrieval at the University of Maryland. In *The Sixth Text REtrieval Conference (TREC-6)*. National Institutes of Standards and Technology, November 1997. http://trec.nist.gov/.

[3] Páraic Sheridan and Jean Paul Ballerini. Experiments in multilingual information retrieval using the SPIDER system. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, August 1996. http://www-ir.inf.ethz.ch/Public-Web/sheridan/papers/SIGIR96.ps.

[4] Laura Slaughter, Douglas W. Oard, Vernon L. Warnick, Julie L. Harding, and Galen J. Wilkerson. A graphical interface for speech-based retrieval. In Ian Witten, Rob Akscyn, and Frank M. Shipman, III, editors, *The Third ACM Conference on Digital Libraries*, pages 305–306, June 1998.

[5] Ellen M. Voorhees, Narendra K. Gupta, and Ben Johnson-Laird. Learning collection fusion strategies. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 172–179. ACM Press, July 1995.

TREC-7 CLIR using a Probabilistic Translation Model

Jian-Yun Nie
Laboratoire RALI,
Département d'Informatique et Recherche opérationnelle,
Université de Montréal
C.P. 6128, succursale Centre-ville
Montréal, Québec, H3C 3J7 Canada
nie@iro.umontreal.ca

In this report, we describe the approach we used in TREC-7 Cross-Language IR (CLIR) track. The approach is based on a probabilistic translation model estimated from a parallel training corpus (Canadian HANSARD). The problem of translating a query from a language to another (between French and English) becomes the problem of determining the most probable words that may appear in the translation of the query. In this paper, we will describe the principle of building the probabilistic model, and the runs we submitted using the model as a translation tool.

1. Introduction

For Cross-Language IR (CLIR) the solution that immediately comes to one's mind is to translate the information query using a machine translation (MT) system, and to submit the resulting translation to a classical monolingual IR system. In [Nie98], we compared this approach with the two following ones:

- using a bilingual dictionary;
- using a probabilistic translation model.

Our results on TREC-6 data showed that using a bilingual dictionary alone lead to poor performances; but using a probabilistic translation model, we obtained a performance close to those with commercial MT systems (LOGOS and SYSTRAN).

In TREC7, we used the same strategy. A probabilistic translation model is used to translate queries from a language to another (between English and French). The translation result is a list of words, together with a probability value. It is then submitted to a modified SMART system for retrieval.

Let us first give a brief description on how the probabilistic model is built, then we will describe our tests in Trec7.

2. A Probabilistic Translation Model

By translation model, we mean a mechanism which associates to each source language sentence (or query) **e** a probability distribution $p(\mathbf{f}|\mathbf{e})$ on the sentences (or queries) **f** of the target language. A precise description of a family of such models can be found in Brown & al. [Brown93]. The model we will be using for the experiments reported here is basically their "Model 1". In this model, a source **e** and its translation **f** are connected through an alignment **a**, that is a mapping of the words of **e** onto those of **f**. If $\mathbf{e} = e_1, e_2, \ldots, e_l$ and $\mathbf{f} = f_1, f_2, \ldots, f_m$ then $\mathbf{a}_j$ will be used to refer to the particular position in **e** that is connected with position j in **f** (for example, $\mathbf{a}_2 = 4$ expresses the fact that f_2 is connected with e_4) and e_{aj} will be used to refer to the word in **e** at position $\mathbf{a}_j$.

The probability $p(\mathbf{f}|\mathbf{e})$ is decomposed as a sum over all possible alignments:

$$p(\mathbf{f}|\mathbf{e}) = \Sigma_{\mathbf{a}\in \mathbf{A}} p(\mathbf{f}, \mathbf{a}|\mathbf{e})$$

The conditional probability of **f** under alignment **a** given **e** can be analyzed as follows:

$$p(\mathbf{f},\mathbf{a}|\mathbf{e}) = p(\mathbf{f}|\mathbf{a},\mathbf{e})\, p(\mathbf{a}|\mathbf{e}) = K_{\mathbf{e},\mathbf{f}}\, p(\mathbf{f}|\mathbf{a},\mathbf{e})$$

The latter equality stems from the fact that in model 1, all alignments are considered equiprobable. Consequently $p(\mathbf{a}|\mathbf{e})$ is a constant $K_{\mathbf{e},\mathbf{f}}$ equal to 1 over the total number of alignments.

The core of the model is $t(f|e)$, the lexical probability that some word e is translated as word f. The value of $p(\mathbf{f}|\mathbf{a},\mathbf{e})$ depends mostly on the product of the lexical probabilities of each word pair connected by the alignment:

$$p(\mathbf{f}|\mathbf{a},\mathbf{e}) = C_{\mathbf{f},\mathbf{e}} \prod_{j=1,m} t(f_j|e_{aj})$$

where $C_{\mathbf{f},\mathbf{e}}$ is a constant that accounts for certain dependencies between the respective lengths of sentences **e** and **f** (mostly irrelevant here).

The probability of observing word f_j in **f** under a particular alignment **a** is:

$$p(f_j|\mathbf{a},\mathbf{e}) = t(f_j|e_{aj})$$

And the probability of observing word f_j in **f** under any alignment is:

$$p(f_j|\mathbf{e}) = \Sigma_{i=1,l}\, t(f_j|e_i)$$

Since all alignments are considered equiprobable, we can simply sum up the values obtained by connecting f_j to each word $e_1, e_2, \ldots, e_l$ of **e**. In other words, the probability of observing a particular word in a given position in **f** is established as the total of the lexical contributions of each word of **e**.

The parameters of our translation model are estimated from a bilingual parallel corpus in which each sentence has been aligned with the corresponding sentence(s) of the other language. Such alignments can be produced using algorithms such as the one described in [Simard92]. Given such alignments we can estimate reasonable values for the parameters $t(f|e)$ using the Expectation Maximization algorithm, as described in [Brown93]. The model used in the experiments reported here has been trained using 8 years of the Canadian Hansard (parliamentary debates), that is, approximately 50 million words in English and in French.

We noticed in [Nie98] that the probabilistic model cannot distinguish true translation words from those only statistically associated, in particular, when the source words have low occurrence frequency in the training corpus. In order to solve this problem, we enforced, in a query translation, the "probability" of the words that are recognized as translations of some query words in a bilingual dictionary. This leads to a combined approach. Our experiments with TREC-6 data showed that this combination is very effective. In general, we obtained about 5% increase in average precision over the approach using the probabilistic model alone. On TREC-6 data, we used a small bilingual dictionary with less than 8000 words. It is showed that when the rate of enforcement was set at 0.02 we obtained the best performance. For TREC-7 experiments, we used a larger bilingual dictionary (a terminology database) with

about 1.2 million entries (most of them are compound terms). The enforcement rate has been set at 0.01 because we have now much more translations added in.

3. Experiments

We used a modified version of SMART system [Buckley85] for monolingual document indexing and retrieval. The *ltc* weighting scheme is used for documents. For queries, we used the probabilities provided by the probabilistic model, multiplied by the *idf* factor. From the translation words obtained, we retained the 50 most probable words. This limit in number allows us to eliminate many noisy words in the translation that are simply statistically related to query words. The setting of 50 has been shown to be reasonable on Trec6.

Before indexing, a text is first stemmed as follows: According to a probabilistic tagging, each word is first associated with a (or several) grammatical category. It is then transformed into a canonical, citation form. For example, nouns and (French) adjectives are transformed into their masculine singular form, and verbs are transformed into their infinitive forms.

Our initial goal of participating in TREC-7 is to re-evaluate how effective the cross-language IR based on the probabilistic translation model is. So, we first submitted the following 4 runs:

- RaliAPf2e: Using French queries to retrieve AP English documents. This run only uses the probabilistic model;
- RaliDicAPf2e: The same as above, but it combines the probabilistic model and the bilingual dictionary.
- RaliSDAe2f: Using English queries to retrieve SDA French documents. This run only uses the probabilistic translation model.
- RaliDicSDAef: The same as above, but using the combined approach.

Later on, we also submitted two other runs in which SDA French documents and AP English documents are merged.

- RaliDicE2EF: Using English queries to retrieve English and French.
- RaliDicF2EF: Using French queries to retrieve English and French documents.

In these two runs both the probabilistic model and the bilingual dictionary are used.

Simple CLIR

The monolingual runs are performed using *ltc-ltc* weighting with SMART. The CLIR runs used *mtc-ltc* weighting. Table 1 shows the performances obtained in comparison with the monolingual runs on the same collection.

As we can see, the CLIR effectiveness is comparable to the monolingual runs. This is quite surprising because on TREC-6 data, the same approach led to performances of about 80% of the monolingual runs. What is even more surprising is the better performances obtained in French to English CLIR, than the English to English monolingual run. A possible explanation, in addition to some slight differences between the original English and French queries, is that the probabilistic translation allows us to include some very useful related words or synonyms. This phenomenon has been observed in a number of queries.

	Mono (F-F)	RaliDic	Rali
Rel.	991		
Rel.Ret.	806	792	784
0.0	0.6559	0.5983	0.5653
0.1	0.4829	0.4481	0.4598
0.2	0.4456	0.3973	0.3980
0.3	0.3808	0.3310	0.3118
0.4	0.3168	0.3003	0.2906
0.5	0.2623	0.2623	0.2563
0.6	0.1930	0.2010	0.1946
0.7	0.1544	0.1684	0.1597
0.8	0.1156	0.1390	0.1330
0.9	0.0726	0.0723	0.0763
1.0	0.0100	0.0159	0.0217
Avg.prec.	0.2658	0.2551	0.2491

SDA English to French retrieval (E-F)

	Mono (E-E)	RaliDic	Rali
Rel.	1689		
Rel.Ret.	1231	1381	1416
0.0	0.6128	0.6441	0.6543
0.1	0.4552	0.5042	0.5343
0.2	0.3987	0.4395	0.4549
0.3	0.3696	0.4155	0.4209
0.4	0.3418	0.3949	0.3943
0.5	0.3060	0.3362	0.3399
0.6	0.2487	0.2709	0.2796
0.7	0.2242	0.2365	0.2434
0.8	0.1938	0.1970	0.2046
0.9	0.1277	0.1495	0.1448
1.0	0.0699	0.0741	0.0727
Avg.prec.	0.2864	0.3186	0.3229

AP French to English retrieval (F-E)

Table 1. English to French and French to English runs.

Let us illustrate this by the following examples.

Query 30: Famine in Sudan
famine=0.154774
soudan=0.129183
étude=0.075273
étudier=0.023295
sévir=0.010796
pouvoir=0.010070
victime=0.007599
présenter=0.007366
port-soudan=0.006182
soudanais=0.006059
effectuer=0.005955
pressant=0.005752
trois=0.005726
secours=0.005652
seulement=0.004535
publier=0.004400
lutter=0.004091
signaler=0.003660

query 40: Concorde Supersonic Jet
français=0.042530
développement=0.037197
avion=0.033672
supersonique=0.030150
concorde=0.029113
réaction=0.027248
colombie-britannique=0.026678
pouvoir=0.014754
coopératif=0.013521
opération=0.012910
utiliser=0.010497
identifier=0.010412
activité=0.009597
question=0.009530
jet=0.009523
venu=0.009260
concorder=0.009003
britannique=0.008239

We observe that the top-ranked French words found for these queries are highly relevant to the original English queries. Some related words are also found. For example, "victime", "port-soudan" and "soudanais" in query 40, and "avion", "réaction" in addition of the true translation "supersonique" and "jet".

However, we can notice several translation problems.

- Some non-significant common words such as "pouvoir" (can) have been included in a number of translations. These words, however, cannot be put in the stop list because they are meaningful in some cases ("pouvoir" may also mean "power"). This problem can be partly solved by including *idf* factor in the final weighting. In the final vector obtained with *mtc* weighting, the word "pouvoir" appears at 26th rank.
- Due to the particularities of the training parallel corpus, the word "British" in query 40 (in the description field) is translated first by "colombie-britannique" with a much higher probability than "britannique". This phenomenon caused more problems than the previous one because *idf* cannot decrease their importance in the final vector. In the final vector, "colombie-britannique" is the 5th most important term.
- Many unrelated words appear in the translation because they occur often in a sentence that is aligned with one containing a word of the original query. For example, we can notice "effectuer" (carry out) in the translation of query 30, and "activité" (activity) in that of query 40.

In order to compare with an MT system, we translated the queries with the Systran system. The translated queries processed as in the monolingual runs. The following table shows the performances obtained.

	E-F	F-E
Trec7	0.2206	0.3185

Table 2. Average precision using MT

We can see that the probabilistic translation model performed slightly better than the Systran system under the same conditions. This confirms the same conclusion we drawn in [Nie98] using the Trec6 data.

Merging runs

Our emphasis in this Trec CLIR track has been put on simple CLIR without merging. The merging run has been submitted at the last minute. We did not spend much time to define a reasonable merging strategy. We used a very simple approach: The original queries (English or French) are used to retrieve documents in the same language from one of the two collections (AP or SDA), and the translated queries are used to retrieve documents in the other collection. Retrieved documents from the two collections are re-ranked according to their similarities to the queries.

The problem we were facing with is that the similarities obtained in monolingual IR and CLIR are not comparable. Words in vectors are weighted in very different ways. In monolingual runs, the SMART's *ltc* scheme is used, whereas in the CLIR runs, the weight is a combination of translation probability and *idf*. The direct merging of the two document lists resulted in a very unbalanced ranking of AP and SDA documents: Either we have many AP documents at the top level, or the SDA documents at the top level.

In order to solve partly the incompatibility of similarities, we chose to use *mtc* for queries and *ltc* for documents in both cases. The documents from the two runs seem to be more

balanced in the merged result, although not completely. Typically, we still observed that the similarities in the monolingual answer set are more distanced between the top and bottom than in the CLIR answer set.

Table 2 shows a comparison of these two merging runs with other runs in this category.

		Rel. Ret. @ 1000					Avg. Prec.				
Topic	Rel.	Best	Med.	Worst	E-EF	F-EF	Best	Median	Worst	E-EF	F-EF
26	12	11	9	0	9	7	0.1200	0.0342	0.0000	0.1019 (>)	0.0947
27	84	80	42	10	49	51	0.3200	0.0911	0.0133	0.1506 (>)	0.1428
28	157	147	118	18	118	126	0.6815	0.3748	0.0148	0.3748 (=)	0.3863
29	19	19	12	2	12	11	0.9060	0.5824	0.0005	0.5335 (<)	0.4357
30	133	133	130	12	133	133	0.5784	0.4053	0.0111	0.5784 (B)	0.6521
31	227	202	165	62	194	190	0.4660	0.3548	0.1029	0.3526 (<)	0.3543
32	57	57	56	12	56	56	0.8428	0.7565	0.0117	0.7469 (<)	0.6779
33	87	83	66	10	40	48	0.6516	0.2657	0.0158	0.0501 (<)	0.0918
34	11	11	11	2	11	6	0.1218	0.0341	0.0013	0.0341 (=)	0.0157
35	74	60	46	13	46	32	0.1520	0.0975	0.0078	0.0763 (<)	0.0337
36	114	108	84	15	84	79	0.6559	0.3349	0.0091	0.2560 (<)	0.1497
37	44	37	14	0	14	19	0.3675	0.0247	0.0000	0.0115 (<)	0.0179
38	147	144	135	16	135	138	0.6794	0.4239	0.0027	0.5330 (>)	0.5413
39	35	32	16	2	32	30	0.1223	0.0609	0.0012	0.0500 (<)	0.1432
40	43	43	38	1	42	41	0.7890	0.5626	0.0000	0.7890 (B)	0.5999
41	290	277	239	4	239	222	0.7337	0.4104	0.0001	0.4104 (=)	0.4165
42	55	50	31	6	41	39	0.3246	0.0622	0.0288	0.0604 (<)	0.1078
43	242	142	96	6	7	137	0.2112	0.0549	0.0005	0.0005 (W)	0.1976
44	6	6	4	1	4	4	0.4095	0.2565	0.0003	0.2711 (>)	0.1826
45	47	47	45	6	23	22	0.7028	0.3158	0.0010	0.2114 (<)	0.2653
46	2	2	1	0	2	1	0.0083	0.0026	0.0000	0.0026 (=)	0.0006
47	140	139	136	25	136	137	0.6186	0.3568	0.0331	0.3568 (=)	0.5304
48	101	93	48	15	69	77	0.6608	0.1277	0.0232	0.1277 (=)	0.3266
49	130	109	100	8	102	94	0.4673	0.1905	0.0004	0.1905 (=)	0.1506
50	216	158	116	10	119	135	0.3943	0.1529	0.0042	0.1529 (=)	0.1736
51	43	42	39	14	41	42	0.7903	0.5803	0.0366	0.6307 (>)	0.5110
52	51	49	33	6	33	33	0.5317	0.1429	0.0012	0.0952 (<)	0.0838
53	113	36	17	2	7	50	0.0575	0.0060	0.0001	0.0007 (<)	0.0569
Avg.	92.4	79.9	63.69	9.59	62	67.59	0.4609	0.2435	0.0111	0.2465	0.2622

Runs : E-EF = RaliDicE2EF,
F-EF = RaliDicF2EF

Table 3. Merging runs with English and French documents

For the E2EF run, the comparison with other runs is shown in the following table. The average precision for all the queries is about the same as the median.

Best	> median	= median	< median	Worst
2	5	8	12	1

Table 4. Comparison with other participants

Although the merge run on English and French documents using French queries is not an official category, we also provide this run in the above table in order to compare with the CLIR run using English queries. In the French to English/French run, we obtained slightly better average precision on all the queries.

The difference between the two runs is the sharpest for query 43. After analyzing the query, we found that the poor performance in E-EF run was due to a mistake in manipulating the original query. The original query has been wrongly altered, so that the monolingual retrieval did not find any relevant document for this query. After correcting the situation, we obtained an average precision of 0.1636 for this query in monolingual run, and 0.1472 in the merge run. This is above the median level.

The medium performance of the merge run is not surprising to us. In choosing *mtc* weighting for monolingual run, we knew that the effectiveness will drop (this has been tested on Trec6 data). This, in addition to the still unbalanced ranking of SDA and AP documents in the final list, greatly affected the merge run.

4. Final remarks

Our participation to the TREC-7 CLIR track is to verify the effectiveness of our approach using a probabilistic translation model. Our previous experiments with TREC-6 data [Nie98] showed that CLIR using this approach may match and even surpass that using commercial MT systems. The tests in Trec7 confirmed this once more. However, there are several problems in the translation model used. We will try to improve the model and its application to CLIR in the future.

In comparison with the best performances of CLIR, our results are still low. The main reason lies in the global setting of the system. The weighting schemes we used are not the most effective. In the future, we will try to use better weighting scheme such as *ltu* or *Okapi* formula. Despite this, our comparison with the monolingual runs and the runs using Systran still hold. They are carried out under the same condition. So we expect to have the same comparison with new weighting schemes or other system setting.

References

[Brown93] P. F. Brown, S. A. D. Pietra, V. D. J. Pietra, and R. L. Mercer, The mathematics of machine translation: Parameter estimation. *Computational Linguistics*, vol. 19, pp. 263-312 (1993).

[Buckley85] C. Buckley, *Implementation of the SMART information retrieval system*. Cornell University, Technical report 85-686, (1985).

[Nie98] J.Y. Nie, P. Isabelle, P. Plamondon, G. Foster, Using a probabilistic translation model for cross-language information retrieval, *Sixth workshop on Very Large Corpora*, Montreal, pp. 18-27, (1998)

[Simard92] M. Simard, G. Foster, P. Isabelle,Using Cognates to Align Sentences in Parallel Corpora, *Proceedings of the 4th International Conference on Theoretical and Methodological Issues in Machine Translation*, Montreal (1992).

IRIS at TREC-7

Kiduk Yang, Kelly Maglaughlin, Lokman Meho, and Robert G. Sumner, Jr.
School of Information and Library Science
University of North Carolina
Chapel Hill, NC 27599-3360 USA
{yangk, maglk, mehol, sumnr}@ils.unc.edu

0 Submitted Runs

unc7aal1, unc7aal2 – Category A, automatic ad-hoc task runs (long query)
unc7ias, unc7iap – interactive track runs

1 Introduction

In our TREC-5 ad-hoc experiment, we tested two relevance feedback models, an adaptive linear model and a probabilistic model, using massive feedback query expansion (Sumner & Shaw, 1997). For our TREC-6 interactive experiment, we developed an interactive retrieval system called IRIS (Information Retrieval Interactive System[1]), which implemented modified versions of the feedback models with a three-valued scale of relevance and reduced feedback query expansion (Sumner, Yang, Akers & Shaw, 1998). The goal of the IRIS design was to provide users with ample opportunities to interact with the system throughout the search process. For example, users could supplement the initial query by choosing from a list of statistically significant, two-word collocations, or add and delete query terms as well as change their weights at each search iteration. Unfortunately, it was difficult to tell how much effect each IRIS feature had on the retrieval outcome due to such factors as strong searcher effect and major differences between the experimental and control systems.

In our TREC-7 interactive experiment, we attempted to isolate the effect of a given system feature by making the experimental and control systems identical, save for the feature we were studying. In one interactive experiment, the difference between the experimental and control systems was the display and modification capability of term weights. In another experiment, the difference was relevance feedback by passage versus document.

For the TREC-7 ad-hoc task, we wanted to examine the effectiveness of relevance feedback using a subcollection in order to lay the groundwork for future participation in the Very Large Corpus experiment. Though the pre-test results showed the retrieval effectiveness of a subcollection approach to be competitive with a whole collection approach, we were not able to execute the subcollection retrieval in the actual ad-hoc experiment due to hardware problems. Instead, our ad-hoc experiment consisted of a simple initial retrieval run and a pseudo-relevance feedback run using the top 5 documents as relevant and the 100th document as non-relevant.

Though the precision was high in the top few documents, the ad-hoc results were below average by TREC measures as expected. In the interactive experiment, the passage feedback results were better than the document feedback results, and the results of the simple interface system that did not display query term weights were better than that of the more complex interface system that displayed query term weights and allowed users to change these weights. Overall interactive results were about average among participants.

2 Key Components of IRIS

2.1 Text Processing

IRIS processes the text first by removing punctuation, and then excluding the 390 high-frequency terms listed in the WAIS default stopwords list as well as "IRIS stopwords," which are defined as all numeric words, words that start with a special character, words consisting of more than 25 non-special characters, and words with embedded special characters other than a period, apostrophe, hyphen, underline, or forward or backward slash. The IRIS stopwords definition was arrived at by examining the inverted index and identifying low frequency terms that appeared meaningless. The removal of IRIS stopwords reduced the number of unique terms by over 25% (401,423 to 295,257 in the Financial Times collection), which can effect considerable savings in machine resources. Such savings can be a significant factor when dealing with massive collections.

After the initial processing step described above, IRIS conflates each word by applying one of the four stemmers implemented in the IRIS Nice Stemmer module,[2] which consists of a simple plural remover (Frakes & Baeza-Yates, 1992, chap. 8), the Porter stemmer (Porter, 1980), the modified Krovetz inflectional stemmer, and the Combo stemmer. The modified Krovetz inflectional stemmer implements a modified version of Krovetz's inflectional stemmer algorithm (Krovetz, 1993) and restores the root form of plural ("-s," "-es," "-ies"), past tense ("-ed"), and present participle ("-ing") words, provided this root form is in our online dictionary. Though this

[1] A prior version of IRIS was developed by Kiduk Yang, Kristin Chaffin, Sean Semone, and Lisa Wilcox at the School of Information and Library Science (SILS) at the University of North Carolina. They worked under the supervision of William Shaw and Robert Losee.

[2] Nice stemmer was implemented by Kiduk Yang, Danqi Song, Woo-Seob Jeong, and Rong Tang at SILS at UNC.

stemmer's conservative conflation approach can be advantageous over suffix-removal stemmers that can adversely affect precision by overstemming, it can also cause lower recall by understemming, since the morphological variations targeted for conflation are few. The Combo stemmer attempts to minimize the disadvantages of both understemming and overstemming by taking as the final result the shortest whole word (i.e., word that appears in a dictionary) returned by the three stemmers. For example, the Krovetz stemmer does not conflate "disappointment" and "goodness," and the Porter Stemmer overconflates "ponies," "agreed" and "troubling" to "poni," "agre," and "troubl," but the Combo stemmer correctly stems these words to "disappoint," "good," "pony," "agree," and "trouble."[3] Unfortunately, the Combo stemmer's computational cost is very high due to its multiple dictionary lookup per word. Given the resource limitations at SILS relative to the size of the TREC-7 collection and the fact that the effectiveness of the Combo stemmer has not yet been fully tested, we opted for the modified Krovetz stemmer as the default stemmer for the TREC-7 experiments.

2.2 Phrase Construction

In our TREC-6 experiments, we constructed a statistically significant, two-word collocation index by extracting co-occurring word pairs within a window of 4 words (Haas & Losee, 1994; Losee, 1994) and selecting those that co-occur with statistically significant frequency (Berry-Rogghe, 1974). Though this collocation index worked very well in some cases, its overall effect on retrieval effectiveness did not appear to be significant (Sumner et. al., 1998). Furthermore, the computational cost of constructing the collocation index was quite high.

Consequently, we tried another approach to constructing a phrase index in TREC-7. Using the online dictionary and the clause recognition algorithm built into the Nice Stemmer, we constructed a two-word noun-noun phrase index by first extracting adjacent word pairs of noun and proper noun combinations within a clause,[4] and then discarding the phrases occurring 20 or less times in the collection to reduce indexing time and to conserve computer resources. The phrase occurrence frequency threshold of 20 was arrived at by selecting the number that produced the phrase index whose size was most comparable to that of the collocation index. To augment the proper nouns in the online dictionary, all capitalized words not occurring at the beginning of a sentence were considered to be proper nouns. Since the Krovetz stemmer does not conflate hyphenated words, hyphenated words were broken up and stemmed by the simple plural remover before the noun-noun phrase construction module was applied. Hyphenated words in their raw form (i.e. as they appear in documents sans punctuation) were added to the index as well.

2.3 Ranking Function and Term Weights

IRIS ranks the retrieved documents in decreasing order of the inner product of document and query vectors,

$$\mathbf{q}^{\mathrm{T}}\mathbf{d}_i = \sum_{k=1}^{t} q_k d_{ik}\,, \qquad (1)$$

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. We used SMART *Lnu* weights for document terms (Buckley, Singhal, Mitra, & Salton, 1996; Buckley, Singhal, & Mitra, 1997), and SMART *ltc* weights (Buckley, C., Salton, G., Allan, J., & Singhal, A., 1995) 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, Buckley, & Mitra, 1996). Our implementation of *Lnu* weights was the same as that of Buckley et al. (1996, 1997) except for the value of the *slope* in the formula, which is an adjustable parameter whose optimal value may depend, in part, on the properties of the document collection.

According to the pre-test experiments, an *Lnu* slope of 0.5 performed best with feedback, especially when using both single term and phrase indexes. In initial retrieval without any feedback, however, a slope of 0.2 or 0.3 showed best results. Based on these findings, we used a slope of 0.3 in the ad-hoc experiment to optimize the initial retrieval results, but used a slope of 0.5 in the interactive experiment to optimize performance with feedback.

2.4 Feedback Models

2.4.1 Adaptive Linear Model

Currently, the default relevance feedback model of IRIS is the adaptive linear model (Wong & Yao, 1990; Wong, Yao, Salton, & Buckley, 1991). The basic approach of the adaptive linear model, which is based on the concept of the preference relation from decision theory (Fishburn, 1970), is to find a *solution vector* that, given any two documents in the collection, will rank a more-preferred document before a less-preferred one (Wong et al., 1988).

The goal of the adaptive linear model, in essence, is to construct a query vector that ranks the entire document collection according to the user's preferences. Since the user's preferences are not usually known for the whole collection, however, we can only create a solution vector for the *training set T*, which is the cumulative set of documents retrieved and evaluated by the user. As knowledge of the user's preferences accumulates with relevance feedback iterations, one can expect the solution vector for T to more accurately rank the entire collection (Wong & Yao, 1990).

---

[3] For interactive comparison of these stemmers, please visit http://ils.unc.edu/iris/nstem.htm.

[4] IRIS identifies a clause boundary by the presence of appropriate punctuation marks such as a comma, period, semicolon, question mark, or exclamation mark.

The *error-correction procedure* we used to find a solution vector for T in our TREC experiments is based on the procedure used by Wong et al. (1991). The error-correction procedure begins with a *starting vector* $\mathbf{q}_{(0)}$ and repeats the cycle of "error-correction" until a solution vector is found. The error-correction cycle i is defined by

$$\mathbf{q}_{(i+1)} = \mathbf{q}_{(i)} + \alpha\mathbf{b}\,, \tag{2}$$

where α is a constant, and $\mathbf{b}$ is the *difference vector* resulting from subtracting a less-preferred document vector from a more preferred one. (For details about how this difference vector is chosen, see Sumner et al., 1998.) The choices for the constant α and the *starting vector* $\mathbf{q}_{(0)}$ are very important since they can influence not only the composition of the solution vector but also the number of error-correction cycles needed to arrive at it Different choices have been made for α and $\mathbf{q}_{(0)}$ in our TREC-5, TREC-6, and TREC-7 experiments (Sumner & Shaw, 1997, Sumner et al., 1998).

In the relevance feedback interface of IRIS, users can evaluate documents as "relevant," "marginally relevant," or "nonrelevant." By adapting the concept of the user preference relation to extend the relevance scale from a binary to a three-valued scale, we constructed the following formula for the starting vector. Note that this formula can be adjusted for any multivalued relevance scale:

$$\mathbf{q}_{(0)} = c_0\mathbf{q}_{rk} + \frac{c_1}{N_{new\,rel}}\sum_{new\,rel}\mathbf{d} + \frac{c_2}{N_{new\,mrel}}\sum_{new\,mrel}\mathbf{d} - \frac{c_3}{N_{new\,nonrel}}\sum_{new\,nonrel}\mathbf{d}\,, \tag{3}$$

where $\mathbf{q}_{rk}$ is the query vector that produced the current ranking of documents; c_0, c_1, c_2, and c_3 are constants; $N_{new\,rel}$, $N_{new\,mrel}$, and $N_{new\,nonrel}$ are the number of new relevant, new marginally relevant, and new nonrelevant documents respectively in the current iteration; and the summations are over the appropriate new documents. This formula is similar to the relevance feedback formulas used by Rocchio (1971) and Salton and Buckley (1990). A "new" document during a given search iteration is one that was not retrieved and evaluated during a previous iteration. Alternatively, it may also be a document that was retrieved and evaluated in a previous iteration, but whose relevance judgement was changed in the current iteration.

Because every new document vector already contributes to the starting vector (Equation 3), we used a value of $\alpha = 0.5$ (Equation 2) to reduce the influence of any one new document. The value of $c_2 = 0.6$ was chosen so that a marginally relevant document could still contribute to the final query vector even after being subtracted in the error correction procedure ($c_2 - \alpha = 0.1$). We set $c_1 = 1.2$ so that the influence of relevant documents would be twice that of marginally relevant ones and set $c_3 = 0.6$ for internal consistency. Though we used $c_0 = 1.0$ in TREC-6 experiments, we adjusted it to the value of 0.2 in the TREC-7 interactive experiment to reduce the influence of the initial query. We noticed in our post-TREC6 experiments that the influence of the initial query tended to overshadow the user feedback, and consequently set $c_0 = 0.2$, which seemed to make the system more responsive to the user feedback. In the ad-hoc experiment, however, we used the value of $c_0 = 1.0$ since the importance of pseudo-feedback for that task was viewed as minimal.

2.4.2 Probabilistic Model

In addition to the adaptive linear model, a variation of the binary probabilistic feedback model that accommodates three levels of relevance judgments is implemented in IRIS. As is the case with the adaptive linear model, this probabilistic model with the graded relevance formula (Yang & Yang, 1997) can be adjusted for any multivalued relevance scale including the binary relevance scale.

According to the TREC-7 pre-tests as well as our past TREC results, our implementation of the adaptive linear model performed consistently better than that of the probabilistic model when using binary relevance feedback. The findings of the TREC-6 interactive experiment regarding the comparative performances of the two feedback models using the three-valued relevance scale is inconclusive due to other factors such as searcher effect. Given these considerations and our resource limitations, we decided to exclude the probabilistic model from the actual TREC-7 experiments. The detailed description of the probabilistic model can be found in Sumner et. al. (1998).

2.4.3 Passage Feedback Model

The conventional relevance feedback models assume the user's relevance judgement, whether binary or multi-valued, to be about an entire document. The unit of a document, however, can sometimes be arbitrary, as in the case of web documents whose boundaries are often determined for reasons of convenience or efficiency rather than content, or can contain subsections of various information content as in Congressional Record and Federal Register documents. The findings from passage feedback research (Melucci, 1998) as well as from comments made by IRIS users at large indicate that determination of relevance is sometimes based on certain portions of a document rather than the entirety of it.

To test out this theory in TREC-7 experiments, we implemented in IRIS a third relevance feedback model called the "passage feedback model". The formula for feedback vector creation in the passage feedback model looks almost identical to the "Ide regular" formula (Ide, 1971; Salton & Buckley, 1990), except where the document vector $\mathbf{d}$ is replaced by $\mathbf{p}$, the passage vector.

$$\mathbf{q}_{new} = \mathbf{q}_{old} + \sum_{rel}\mathbf{p} - \sum_{nonrel}\mathbf{p} \tag{4}$$

Since the normalization factor of the *Lnu* weight is based on document length, an inverse document frequency weight was used for the passage vector $\mathbf{p}$.

In the interactive setting of the IRIS passage feedback interface, the unit of passage is determined by the user, who can simply highlight the relevant and nonrelevant portions of documents with a mouse. Passage feedback can also be implemented by

automatically selecting passages with high frequencies of matching query terms, though the automatic determination of nonrelevant passages is not possible with this approach. Such automatic passage feedback approach may be useful if activated after the initial feedback so as to expand the initial query with related terms.

The passage feedback approach differs fundamentally from the philosophy of the adaptive linear model and the probabilistic model. Regardless of whether a document or passage is used as the unit of feedback, the passage feedback model does not attempt, in principle, to rank a document collection in the preference or relevance order defined by a training set. Instead, the passage feedback model, similar to conventional vector space models, simply expands the query vector to make it more "similar" to relevant passages and "dissimilar" to nonrelevant passages.

3 Pre-test Experiments

In our TREC-7 pre-test experiments, we chose to examine the effects of 4 main system components of representation (single term vs. phrases), term weighting (normalization slope), feedback model (adaptive linear vs. probabilistic), and feedback query expansion size (full-expansion, 300 terms, 30 terms). As a preparation for potential future efforts to scale up to massive document collections, we also examined the effectivenss of relevance feedback using a subcollection. The FT collection with TREC-6 queries and relevance judgements was used in these experiments, and many system design decisions in both ad-hoc and interactive experiments were based on the findings from the pre-test results.

3.1. System Component Tests

3.1.1 Experiment Design

Prior experiments, both in and outside of TREC, have shown the use of syntactic phrases to be only marginally effective (Salton, 1968; Lewis, Croft & Bhandaru, 1989). However, most of the findings were based on the performance of initial retrieval only and did not investigate the effect of automatically expanding the feedback query with phrase index terms.[5]

Though *Lnu* weights with a slope of 0.2 proved effective in both TREC-4 and TREC-5 (Buckley et. al., 1996; Buckley, Singhal, & Mitra, 1997), we found a slope of 0.3 to be more effective with respect to initial retrieval in our TREC-6 experiments (Sumner et. al., 1998). As is the case with phrases, *Lnu* weight experiments did not investigate its effects on retrieval beyond the first feedback iteration.

In our past TREC experiments, we compared the performances of the adaptive linear model (ALM) and probabilistic model (PM) in relevance feedback, and though ALM generally outperformed PM, we noticed distinctly different retrieval patterns between the two models, which warranted further investigation (Sumner & Shaw, 1997; Sumner et. al., 1998).

In TREC-6, we also compared the performance of a fully expanded feedback vector with that of a shorter feedback vector, namely one with the top 250 positive-weighted terms and the lowest 50 negative-weighted terms. Previous routing and adhoc pseudo-feedback experiments in TREC have shown that effectiveness improves linearly with the log of the number of added terms, with the point of diminishing improvement at 300 terms (Buckley et. al., 1995). This was in direct contrast to our results in TREC-6, which, though somewhat suspect due to a system bug, indicated superior performance of the fully expanded feedback vector over the 300 term feedback vector. However, the advantage gained by full expansion of the feedback vector was marginal and the shorter feedback vector performed reasonably well given its size, which was about one tenth that of the full feedback vector. In addition to reconfirming our previous findings regarding feedback query size, we wanted to investigate the retrieval performance level of an even shorter feedback query that consisted of the top 25 positive-weighted terms and the lowest 5 negative-weighted terms—just to see how much gain in efficiency can be achieved without sacrificing too much in effectiveness.

In order to identify the optimum retrieval component combinations of representation, normalization weight, feedback model, and feedback query size, one of us (Yang) conducted an experiment using 5 retrieval iterations (4 feedback iterations with a feedback window of 20 documents) with 48 retrieval model combinations as outlined below. A feedback window of 20 documents means that the top 20 *previously unretrieved* documents of the current ranking are added to the training set. A feedback window of 20 documents and 5 retrieval iterations were chosen to simulate the capacity of a human searcher based on the data from TREC-6 experiments.

Representation	*Lnu* slope	Feedback model	Feedback expansion
Single term	0.1	Adaptive Linear	Full expansion
Single & phrase term	0.2	Probabilistic	250p + 50n
	0.3		25p + 5n
	0.5		

2 * 4 * 2 * 3 = 48 Retrieval Model combinations

The retrieval results of all the model combinations were then compared using optimum effectiveness (F) in the top 20 documents retrieved as well as using standard TREC evaluation metrics. We chose these evaluation measures because optimum F, which represents the optimum performance level of the top 20 retrieved documents, and TREC metrics, which signify the overall performance level of the top 1000 retrieved documents, tend to complement each other.

[5] Here, routing and filtering experiments are not considered due to the different nature of those tasks and the ad hoc task.

TREC evaluation measures used were average precision across all relevant documents, R-precision, and the total number of relevant documents retrieved in the top 1000 documents. Optimum F is the highest F value in all retrieval iterations, where F is computed from recall and precision (Shaw, 1986) by the formula,

$$F = \frac{2}{\frac{1}{R} + \frac{1}{P}}. \quad (5)$$

3.1.2 Results

The analysis of retrieval results by all evaluation measures used showed a consistent pattern of improved retrieval performance with the larger feedback query. The difference in performance between the 30 term feedback vector and the 300 term feedback vector, however, was much greater than that between the 300 term feedback vector and the full feedback vector. As a matter of fact, the reduction in performance by limiting the feedback vector to 300 terms was almost negligible, whereas significant loss of performance occurred by reducing the feedback vector to 30 terms.

Though there were slight variations across evaluation methods, an *Lnu* slope of 0.5 seemed to be most advantageous for ALM and an *Lnu* slope of 0.2 seemed to perform best with PM. Using phrases in feedback as well as single terms resulted in slightly improved retrieval performances by both feedback models, which suggested that using phrases in feedback provides some utility though less than one might hope for.

As was the case in our previous TREC experiments, ALM consistently outperformed PM across all evaluation measures. The difference between the two models was most prominent in the number of relevant documents retrieved in the top 1000 documents, where ALM retrieved hundreds more relevant documents than PM. Upon closer inspection of ALM and PM, we discovered a pattern of "failure" by PM, where PM's feedback query formulation strategy of selecting terms from relevant documents would stagnate the performance of feedback when no more relevant documents could be found. ALM, on the other hand, would continue expanding the feedback vector in its attempt to find the solution vector (Wong et al., 1988; Wong et. al. 1991).

At this point, we devised the "Adaptive Probabilistic Model" (APM), which will keep adding terms from top-ranked non-relevant documents until finding the solution vector that will rank a more-preferred document before a less-preferred one (Wong et al., 1988). Due to time constraints, however, we did not engage in full-scale retrieval experiments with APM. Instead we tested APM in a limited fashion, which resulted in only a marginal improvement of retrieval performance.

3.2. Subcollection Tests

3.2.1 Experiment Design

One of the immediate challenges in the field of Information Retrieval is effective and efficient handling of massive document collections. When dealing with massive document collections, the conventional IR approach of ranking the entire document collection by document-query similarity scores becomes extremely resource-intensive, especially with relevance feedback, where retrieval cycles have to be repeated with expanded query vectors.

One way to deal with massive data may be to create a subcollection, which is small enough to be efficient and yet large enough to contain most of the relevant documents. Once such a subcollection has been created, we can not only refine the search with relevance feedback at relatively small cost, but can also continue to refine and/or update the subcollection by periodically resubmitting to the entire collection the reformulated query created using the subcollection. The main question in subcollection IR is twofold; First, how do we create a subcollection with high enough recall and small enough size? Second, is the retrieval performance of an optimum subcollection competitive to that of the whole collection? In order to investigate these questions, we explored various subcollection creation methods to identify the optimum subcollection creation method, after which we compared its retrieval performance with that of using the whole collection.

The first objective of subcollection creation is to maximize recall at some optimum document rank *N*, so that the subcollection is small enough to be efficient while containing enough relevant data to be effective. In addition to applying the optimum retrieval component combinations identified in the system component tests, we implemented combinations of document-reranking methods to retrieve relevant documents that may not necessarily contain any initial query terms. Initial retrieval, being essentially a Boolean OR retrieval, will only retrieve documents that contain at least one query term. Thus, a poorly formulated initial query will tend to not retrieve many relevant documents that may include only synonyms or related concept terms.

One way to overcome this problem is to expand the query vector with synonyms or related concept terms as well as using word-stems to conflate the morphological variations. Relevance feedback also expands the query vector indirectly with synonyms and related concept terms often contained in the body of relevant documents, though the effect may not be as precise as using a thesaurus or other such natural language processing methods. Consequently, we experimented with expanding the initial query with noun-noun phrases as well as expanding it by applying the "pseudo-relevance feedback" method of assuming that the top 5 documents are relevant and the 100th document is non-relevant. Variations on this method of expanding the initial query by pseudo-relevance feedback (using terms from the top *n* documents) have been used by top performing participants in past TREC ad-hoc experiments (Buckley et. al., 1995; Voorhees & Harman, 1997).

In addition to query expansion by phrases and automatic feedback, we also tested query expansion methods by using passages[6] with matching initial query terms. Three variations of query expansion by passage feedback were tested by selecting terms from only

[6] IRIS identifies a passage boundary by SGML tags or a clause break followed by a carriage return.

the "relevant" passages, terms from relevant passages and top-ranked "non-relevant" passages, and terms from all passages (both relevant and non-relevant) in the top 100 documents retrieved. The "relevance" of a passage is determined by an arbitrary threshold of matching query term numbers in a given passage. These subcollection creation methods along with the baseline method of initial retrieval with the unexpanded original query were then investigated by comparing recall at various document ranks up to the rank of 21,000 (10 % of FT collection).

After identifying the optimum subcollection creation method and cutoff, we created 47 subcollections, one for each query[7], and recomputed their collection statistics, namely *Lnu* weights for documents, and *ltc* weights for queries. We then performed 5 retrieval iterations with a feedback window of 20 documents using selected retrieval models from the system component tests, and compared the performance of subcollection retrieval with that of whole collection retrieval. The same evaluation metrics used in the analysis of system component test results were applied to evaluate the performance of subcollection retrieval.

3.2.2 Results

According to recall values at fixed document ranks, the top-performing subcollection creation method was pseudo-relevance feedback by ALM, though average recall (recall averaged over queries) at document rank 5000 was the same for the top 4 methods. Interestingly enough, the baseline method was one of the top 4 methods, performing only slightly below the methods of initial query expansion by phrases and feedback by ALM. The results of passage feedback methods (PFM) were disappointing. However, poor performance of PFM could be due to an improper threshold of "relevant" passage identification (e.g. passages with n or more query terms). The hypothesis that a relevant passage would include related terms and concepts is critically dependent on the correct identification of relevant passages. Since the top 4 subcollection creation methods all achieved average recall of 0.87 at 5000 documents, which is only 2.4% of the total FT collection, we chose the optimum cutoff at 5000 and decided on the simplest method (i.e. baseline initial retrieval with single term queries) to create the subcollections.

The comparison of the subcollection retrieval results with the whole collection retrieval results showed an interesting difference between ALM and PM. The performance of PM using a subcollection was better than that of PM using the whole collection, whereas ALM's performance deteriorated slightly with subcollection retrieval. Overall performance of ALM, however, was again superior to that of PM, though the gap in performance between ALM and PM was much narrower in subcollection retrieval than whole collection retrieval.

Different behaviors of ALM and PM may be attributed to fundamental differences in feedback query formulation between the two models. ALM starts out with the initial query vector and keeps adding and subtracting terms to find the solution vector, which is a radically different approach from PM's strategy of estimating the probability of term occurrence in all relevant/nonrelevant documents from its occurrence characteristics in a training set. ALM's feedback vector is firmly anchored with the initial query terms and is more resilient to improper and/or insufficient feedback evaluations, whereas PM's feedback vector can be affected severely by bad relevance judgements and small training sets. It is therefore reasonable to assume that PM will perform better as the ratio of the training set size to the document collection size increases, as in the case of the subcollection retrieval.

The overall results of relevance feedback using subcollections has shown it to be almost as effective as using the whole collection while being much more efficient. However, by virtue of the fact that only the top 100 documents of various TREC runs are evaluated for a given set of topics, the TREC test collection may be inherently put together to show high recall for the top N documents, given N is sufficiently large enough. Accordingly, it is difficult to tell how much of these good results using subcollections are due to TREC bias, and whether the optimum subcollection creation method of using simple initial retrieval or ALM pseudo-relevance feedback at 2 or 3% of total collection cutoff will still be applicable in other instances. Though high recall value at such a low rank (under 3% of the total document collection) is somewhat suspect due to the potential bias introduced by the TREC pooling method of relevant document identification (Voorhees & Harman, 1997), it is still reasonable to think that subcollection IR can be an effective as well as efficient way to deal with the problem of massive document collections.

4 Ad-hoc Experiment

4.1 Research Question

As a natural consequence of our belief that the user is an integral component of a truly effective information retrieval system, our approach to information retrieval centers on various ways to involve the user and then to effectively incorporate the user contribution into the search process. Thus, our main goal of the ad-hoc experiment was to explore methods of preparing the system for such an eventuality. More specifically, we wanted to examine a strategy for creating a subset of a document collection to be used in relevance feedback.

One obvious advantage of using a subcollection is the reduced computational cost. If it can be shown that the retrieval effectiveness of using a subcollection is competitive to that of using a whole collection, then subcollection retrieval may be a desirable strategy when iteratively querying (e.g. relevance feedback, query refinement) a document collection (Sumner & Shaw, 1997) that is massive or composed of distributed collections, where collection statistics for the whole collection are not known. A less obvious advantage of a subcollection might be its increased homogeneity. Being more densely populated with relevant documents that are likely to be topically similar, a subcollection may be more responsive to a refined query than a whole collection with diverse subject matter. For example, "court rulings on the use of peyote" queried against the entire Web document collection may retrieve documents

[7] Three TREC-6 queries that did not have any relevant FT documents were dropped from pre-test experiments.

about *either* courts *or* peyote, whereas the same query submitted to a subcollection of legal documents may boost those documents specifically about court rulings on the use of peyote to the top of the document ranking (Sumner, Yang & Dempsey, 1998).

For these potential advantages to be realized, a subcollection has to be small enough to incur savings in computational cost while at the same time contain enough relevant document to be effective. Thus, we are interested in finding answers to the following questions regarding subcollection retrieval strategy.

- What is the best way to create a subcollection?
- How effective is subcollection retrieval compared to whole collection retrieval?

The focus of our ad-hoc experiment, therefore, was on achieving high recall at some reasonable document rank in order to create an effective and efficient subcollection for relevance feedback.

4.2 Research Design

The constitution of the ad-hoc collection compounds the subcollection question. Since the ad-hoc collection is made up of four document collections, subcollection creation methods can be applied to the document collections separately or as a whole. If subcollection creation methods are applied to individual collections, then the question of how the results should be merged must be addressed. In previous research on this "collection fusion" problem, various strategies were employed to compensate for the potential incomparability of query-document similarity scores across collections (Voorhees, Gupta, & Johnson-Laird, 1995; Savoy, Calve, & Vrajitoru, 1997).

Though the "raw score" merging method can be problematic when collection-dependent term weights (i.e. *idf* weight) cause the retrieval scores of similar documents to vary in different collections (Dumais, 1993; Voorhees et. al., 1995), we thought longer queries and massive retrieval window used for subcollection creation might mute its adverse effects. Any advantages gained by more complex retrieval strategies are likely to have less impact on subcollection creation, whose goal is to retrieve the bulk of relevant documents at an acceptable document rank. Ideally, these assumptions should be empirically tested by experimenting with exhaustive combinations of subcollection creation, collection fusion, and various ad-hoc retrieval strategies, but we decided to test only a few subcollection creation methods for the reasons of practicality and simplicity. One of the overriding factors that influenced our research design in TREC-7 was the machine resource limitations that restricted a large scale experimentation. Besides, we figured if a simple method could create an effective enough subcollection, we could forgo complex methods in favor of a simple one.

Therefore, we chose the two most simple and yet effective subcollection creation methods from the pre-test and applied them to individual collections separately. Subcollection creation methods tested were:

- *unc7aal1*: Collection fusion by raw score merging of the simple initial retrieval results without any feedback.
- *unc7aal2*: Collection fusion by raw score merging of the pseudo-relevance feedback results with the adaptive linear model using the top 5 retrieved documents as relevant and the 100th document as non-relevant.

Both *unc7aal1* and *unc7aal2* were produced by first retrieving 10% of documents in each collection and merging the results by their raw query-document similarity scores.

The second phase of our ad-hoc experiment, which we planned to do in a post-study, involved performing relevance feedback on the subcollections by using the official TREC relevance judgments for the top 20 retrieved documents. The results of relevance feedback on a subcollection would then be compared with that on the whole collection to determine the relative effectiveness of subcollection retrieval.

The system construct for the ad-hoc experiment was based on the system component pre-test results. We used the document term weight of *Lnu* 0.3 to optimize the initial retrieval results and allowed for the full feedback query expansion to maximize the feedback effect. We also heavily weighted the initial query in the starting vector formulation of the adaptive linear model (i.e. $c_0 = 1.0$ in Equation 3) to reduce the adverse effect of the pseudo-relevance feedback. We did not create a phrase index for the ad-hoc experiment since we thought its creation cost in time and machine resources far outweighed any potential benefit gained by using it.

4.3 Results

The TREC-7 ad-hoc collection consists of 130,471 FBIS, 19,842[8] Federal Register, 210,158 Financial Times, and 131,896 LA Times documents. Each document collection was first processed individually to generate single-word indexes of 243,778 terms for FBIS, 117,743 terms for Federal Register, 295,257 terms for Financial Times, and 222,155 terms for LA Times collection.

Unfortunately, we experienced a hard disk problem that corrupted the entirety of TREC-7 data and disabled our main research computer soon after we completed the first phase of our ad-hoc experiment. We turned in the top 1000 retrieved documents produced by subcollection creation runs of initial retrieval (*unc7aal1*) and pseudo-feedback with ALM(*unc7aal2*). We are still in the process of restoring the data and consequently, we were not able to conduct the second phase of our experiment to test the effectiveness of relevance feedback using a subcollection.

According to TREC evaluation measures, which indicate the retrieval performance of the top 1000 documents only, the pseudo-relevance feedback with the adaptive linear model did slightly better than the initial retrieval without feedback, though both runs performed slightly below the median level of all the ad-hoc participants (Table 1). As for the subcollection creation results, the smaller and homogeneous FT collection results still held true for the larger and heterogeneous ad-hoc collection. As can be seen in Table 2,

[8] Using the corrected document tag <PARENT> instead of <DOCNO> reduced the number of Federal Register documents from 55,630 to 19,842.

there was very little difference in average recall between the two runs. In both runs, the subcollections consisting of only two percent (document rank 10,000) of the whole document collection contained over 80% of the relevant documents on the average.

Table 1. Performance Statistics of top 1000 documents

	unc7aal1	*unc7aal2*	*Best**	*Median**	*Worst**
Average Precision	0.1506	0.1618	0.4334	0.1822	0.0009
Number of Relevant Documents Retrieved	2188	2264	3890	2481	79

* Best, Median, Worst of all TREC7 ad-hoc participants

Table 2. Recall at Document Ranks averaged over 50 Queries

Document Rank	Average Recall by *unc7aal1*	Average Recall by *unc7aal2*
1,000	0.57	0.59
5,000	0.75	0.76
10,000	0.80	0.81
15,000	0.84	0.85
20,000	0.85	0.86
40,000	0.90	0.90

Closer examination of individual query results revealed some outlier queries with many relevant documents (e.g. queries 370, 389) or with possibly dissimilar relevant documents (e.g. query 373) that produced recall much below the average. Though it is very conceivable that more complex methods of collection fusion and/or ad-hoc retrieval methods may produce a subcollection with higher recall at a smaller size, whether those methods can push the results of outlier queries beyond the "recall block" remains to be seen.

5 Interactive Experiment

5.1 Research Question

Feedback from IRIS users at large as well as those in TREC-6 include mixed response regarding the complexity of its user interface. Some like the interactive nature of its interface throughout the search process, while others are taken back by the complexity of it. One of the most often mentioned IRIS interface components is its ability to display and modify term weights. Most novice searchers are confused by it, though some like "seeing how the system works" and the opportunity to intervene in the system process.

In addition to the users' ambivalence, there is also the question of how the users' term weight modifications will affect the retrieval result. Certainly, if the user does not understand the significance of term weights and modifies them inappropriately, the search result will be adversely affected. If the system's search direction is amiss and needs to adjusted, however, user intervention by term weight modification might be beneficial.

Another often discussed IRIS component is the relevance feedback interface, especially its three levels of relevance (i.e. "yes," "maybe," and "no," representing relevant, marginally relevant, and nonrelevant). Users like the option of judging a document beyond the dichotomous "relevant' or "nonrelevant," but they are not quite sure what a "marginally relevant" document should be. The question of what makes a document relevant is a fertile ground for research (Schamber, 1991; Barry, 1994). In a prior research, we investigated the relationship between the proportionality and the degree of relevance and found that the number of relevant passages in a document corresponded directly with the degree of relevance awarded to the document by the user (Maglaughlin, Meho, Yang, & Tang, 1998). If the proportionality of relevance is an important factor in determining the relevance of a document, then the relevance levels used by the system should be finely graded to better reflect the user's evaluation of relevance. One method of addressing this aspect of relevance may be to use "passage feedback," where passages instead of documents are used as the unit of relevance feedback.

Based on these observations, we asked the following questions in our TREC-7 interactive experiment.

- Does the display and modification option of term weights in an interactive retrieval system help or hinder the retrieval result?
- Is passage feedback an effective alternative to conventional "document" feedback?

5.2 Methodology

We learned from our TREC-6 interactive experience that it is difficult enough to gauge the effects of various contributing factors in an interactive retrieval experiment without compounding the analysis by introducing numerous system features. Consequently, we attempted to isolate the effect of system features by keeping the experimental and the control system identical except in one aspect.

In one interactive experiment (*unc7iap*), the only difference between the experimental and the control system was the display and modification capability of term weights. In another experiment (*unc7ias*), the difference was relevance feedback by passage versus document. Both experiments used the same control system called "*iriss*" which did not have the term weight display. The experimental system in *unc7iap*, "*irisa*", is essentially the standard IRIS with term weight display used since TREC-6, whereas the

experimental system in *unc7ias*, "*irisp*", implements the passage feedback based on the "simple" interface (i.e. without term weight display) [9].

All three systems use the same initial interface, but the initial query modification interface differs in the display of query terms. Though all three systems offer the "suggested phrases" with which the user can supplement the initial query, *iriss* and *irisp* (Figure 1.1) do not have the term weight display where the user can change the term weights as in *irisa* (Figure 1.2). Instead, the term display of the simple interface offers check boxes users can click to include or exclude terms. The feedback query modification interface is structured in the same fashion (Figures 3.1 and 3.2). In addition to the modification of existing terms, the feedback query modification interface allows the user to add terms with emphasis, indicated by the plus or minus symbol (simple interface, Figure 3.1) or by term weights (advanced interface, Figure 3.2).

Another major difference between systems occurs in the relevance feedback interface. Both *iriss* and *irisa* employ the conventional feedback mechanism of judging the relevance of a document as a whole (Figure 2.1), but this document feedback mechanism is replaced by the passage feedback in *irisp* (Figures 2.2). Instead of checking each document as yes, maybe, or no for relevance, the user can simply copy and paste relevant and nonrelevant portions of documents into the appropriate passage feedback box in *irisp*.

The system construct for all three systems was essentially the same except for the mechanism of passage feedback. This is described in the section 2.4.3. Based on the system component pre-test results, we used document term weights of *Lnu* 0.5 to maximize the relevance feedback influence and restricted the feedback query expansion to the 250 terms with highest positive weights and the 50 terms with lowest negative weights in order to optimize the system for efficiency. We also reduced the contribution of the initial query in the starting vector formulation of the adaptive linear model (i.e. $c_0 = 0.2$ in Equation 3) to allow the actions taken during the feedback process to have a stronger influence on the direction of the search. We also created a phrase index of adjacent noun-noun pairs to use in suggesting potentially useful phrases for the initial query as well as in expanding the feedback vectors.

5.3 Results

The performance of IRIS measured by the mean instance recall (MIR) measure was slightly below the median of all interactive track runs (Table 3). Though the term weight display system of *irisa* had the highest MIR of all three IRIS systems tested, the passage feedback system (*irisp*) showed more improvement when performance is compared pairwise within each experimental run. The superior performance of both *irisa* and *irisp* over *iriss* seems to indicate that searcher interventions help rather than hinder the retrieval process. Also, the high relative MIR of *irisp* suggests passage feedback is an effective alternative to conventional document feedback.

Table 3. Interactive Experiment Result Statistics

	iriss vs. irisp		*iriss vs. irisa*		*Best*	*Median*	*Worst*
Mean Instance Recall	0.281	0.314	0.340	0.350	0.420	0.363	0.221

Tables 4.1 and 4.2 show the information about each searcher's background and search experience gathered by pre-study questionnaires. All searchers had received a bachelor's degree and were enrolled in the School of Information and Library Science. Three searchers had previous graduate degrees. The searchers had been searching between 1 and 15 years, with 5 being the average. Four of the 16 searchers were male.

In addition to the pre-search questionnaire, the searchers also completed a psychometric evaluation in an attempt to assess their query formulation skills. The psychometric evaluation scores, which ranged from 11 to 70, were computed by comparing the searcher's synonyms to a list of "correct" synonyms and scoring a point for each correct synonym they recorded. When the searchers' psychometric scores were compared to their average precision and recall values, little correlation was found between search results and psychometric results (Table 4.3). Additionally, there was almost no relationship found between the searchers' performance and their perceived knowledge of the topics, satisfaction with the search, search confidence, the ease of system use, or understanding of the task. Also, there was almost no relationship found between the searchers' performance and their interactions with the systems (Table 4.4). It was however interesting to note that the searchers using the simple and advanced systems, on average, evaluated and saved more documents than those using the simple and passage systems. This may be due to the difficulty of the passage feedback interface.

[9] Due to an oversight, the system names we submitted to NIST were not consistent across experiments. For the sake of clarity and consistency, we have changed the system name mappings in this paper and submitted the corrected mappings to NIST. Eight subjects searched on *irisa* and *iriss*, and eight searched on *irisp* and *iriss*.

Table 4.1 Response Frequency of *irisa/iriss* Searchers on Pre-Study Questionnaire

	No Experience		Some Experience		Great Deal of Experience
1.Using a point-and-click interface				1	7
2.searching on computerized library catalogs			1	3	4
3.searching on CD ROM systems			3	4	1
4.searching commercial systems		3	4	1	
5.searching WWW search services			1	5	2
6.searching other systems	7			1	
	Never	**Once or twice a year**	**Once or twice a month**	**Once or twice a week**	**Once or twice a day**
7.Searching frequency				3	5
	Strongly Disagree	**Disagree**	**Neutral**	**Agree**	**Strongly Agree**
8.Enjoys carrying out information searches				5	3

Table 4.2 Response Frequency of *irisp/iriss* Searchers on Pre-Study Questionnaire

	No Experience		Some Experience		Great Deal of Experience
1.Using a point-and-click interface				1	7
2.searching on computerized library catalogs				6	2
3.searching on CD ROM systems		1	2	4	2
4.searching commercial systems	1	1	4	1	1
5.searching WWW search services				1	7
6.searching other systems	6		1**		1*
	Never	**Once or twice a year**	**Once or twice a month**	**Once or twice a week**	**Once or twice a day**
7.Searching frequency				4	4
	Strongly Disagree	**Disagree**	**Neutral**	**Agree**	**Strongly Agree**
8.Enjoys carrying out information searches			2	5	1

* OCLC ** Military

Table 4.3 Searchers' Average Psychometric Score, Precision and Recall

	irisa/iriss	*irisp/iriss*
Average Psychometric score	35.37	39.25
Average Precision	.7056	.7255
Average Recall	.3447	.2975
Correlation between Psychometric score and Precision	.442	.372
Correlation between Psychometric score and Recall	.308	.646

Table 4.4 Searchers' use of system features

Searchers	System	initial terms	Fbk Iteration	docs saved	docs evaluated	positive terms modified	negative terms modified	positive terms added	Negative terms added
unc7iap	*irisa*	4.857	3.600	15.829	30.629	0.143	0.057	1.543	1.143
unc7iap	*iriss*	4.118	3.118	15.882	30.618	1.353	0.147	1.176	1.176
unc7ias	*iriss*	4.079	2.947	2.921	13.342	0.105	0.000	2.211	1.632
unc7ias	*irisp*	3.898	2.510	2.571	9.245	0.000	0.020	1.102	0.224

References

Barry, C. L. (1994). User-defined relevance criteria: an exploratory study. *Journal of the American Society for Information Science*, 45(3), 149- 159.

Berry-Rogghe, G. (1974). The computation of collocations and their relevance in lexical studies. In A. J. Aitken, R. W. Bailey, & N. Hamilton-Smith (Eds.), *The Computer and Literary Studies* (pp. 103-112). Edinburgh: Edinburgh University Press.

Buckley, C., Salton, G., Allan, J., & Singhal, A. (1995). Automatic query expansion using SMART: TREC 3. In D. K. Harman (Ed.), *Overview of the Third Text REtrieval Conference (TREC-3)* (NIST Spec. Publ. 500-225, pp. 69-80). 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).*

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.

Dumais, S. T. (1993). LSI meets TREC. In D. K. Harman (Ed.), *Proceedings of the First Text REtrieval Conference (TREC-1), 137-152.*

Fishburn, P. C. (1970). *Utility theory for decision making.* New York: John Wiley & Sons.

Frakes, W. B., & Baeza-Yates, R. (Eds.). (1992). *Information retrieval: Data structures & algorithms.* Englewood Cliffs, NJ: Prentice Hall.

Haas, S. W., & Losee, R. M. (1994). Looking into text windows: Their size and composition. *Information Processing and Management, 30*, 619-629.

Ide, E. (1971). New experiments in relevance feedback. In G. Salton (Ed.), *The Smart System-- experments in automatic document processing* (pp. 337-354). Englewood Cliffs, NJ: Prentice-Hall, Inc.

Krovetz, R. (1993). Viewing morphology as an inference process. *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 191-203.

Losee, R. M. (1994). Term dependence: Truncating the Bahadur Lazarsfeld expansion. *Information Processing and Management, 30*, 293-303.

Lewis, D., Croft, W. B., & Bhandaru, N. (1989). Language-oriented information retrieval. *International Journal of Intelligent Systems, 4,* 285-318.

Maglaughlin, K.L., Meho L., Yang, K., & Tang, R. (1998). Utilizing Users' Relevance Criteria in Relevance Feedback. *Unpublished manuscript.*

Melucci M. (1998). Passage retrieval: A probabilistic technique. *Information Processing & Management. 34*, 43-68.

Porter, M. (1980). An algorithm for suffix stripping. *Program, 14,* 130-137.

Rocchio, J. J., Jr. (1971). Relevance feedback in information retrieval. In G. Salton (Ed.), *The SMART Retrieval System: Experiments in Automatic Document Processing* (pp. 313-323). Englewood Cliffs, NJ: Prentice-Hall.

Salton, G. (1968). *Automatic Information Organization and Retrieval.* McGraw-Hill.

Salton, G., & Buckley, C. (1990). Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science, 41*, 288-297.

Savoy, J., Calve, A., & Vrajitoru, D. (1997). Report on the TREC-5 experiment: Data fusion and collection fusion. In E. M. Voorhees & D. K. Harman (Eds.), *The Fifth Text REtrieval Conference (TREC-5).*

Schamber, L. (1991a). Users' criteria for evaluation in a multimedia environment. *Proceedings of the American Society for Information Science*, Washington, DC, (pp. 126-133). Medford. N.J.: Learned Information, Inc.

Shaw, W. M., Jr. (1986). On the foundatin of evaluation. *Journal of the American Society for Information Science, 37*, 346-348.

Singhal, A., Buckley, C., & Mitra, M. (1996). Pivoted document length normalization. *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 21-29.

Sumner, R. G., Jr., & Shaw, W. M., Jr. (1997). An investigation of relevance feedback using adaptive linear and probabilistic models. In E. M. Voorhees & D. K. Harman (Eds.), *The Fifth Text REtrieval Conference (TREC-5).*

Sumner, R. G., Jr., Yang, K., Akers, R., & Shaw, W. M., Jr. (1998). Interactive retrieval using IRIS: TREC-6 experiments. In E. M. Voorhees & D. K. Harman (Eds.), *The Sixth Text REtrieval Conference (TREC-6).*

Sumner, R. G., Yang, K., & Dempsey, B. (1998). Interactive WWW search engine for user-defined collections. *Digital 98 Libraries: The Third ACM Conference on Digital Libraries.* 307-308.

Voorhees, E., Gupta, N. K., & Johnson-Laird, B. (1995). The Collection fusion problem. In E. M. Voorhees & D. K. Harman (Eds.), *Overview of the Third Text REtrieval Conference (TREC-3).*

Voorhees, E., & Harman, D. (1997). Overview of the Fifth Text Retrieval Conference. In E. M. Voorhees & D. K. Harman (Eds.), *The Fifth Text REtrieval Conference (TREC-5).*

Wong, S. K. M., & Yao, Y. Y. (1990). Query formulation in linear retrieval models. *Journal of the American Society for Information Science, 41*, 334-341.

Wong, S. K. M., Yao, Y. Y., & Bollmann, P. (1988). Linear structure in information retrieval. *Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 219-232.

Wong, S. K. M., Yao, Y. Y., Salton, G., & Buckley, C. (1991). Evaluation of an adaptive linear model. *Journal of the American Society for Information Science, 42*, 723-730.

Yang, K., & Yang, K. (1997). Graded relevance in information retrieval. *Unpublished manuscript.*

Figure 1.1 Initial Query Modification Interface for *iriss* and *irisp*

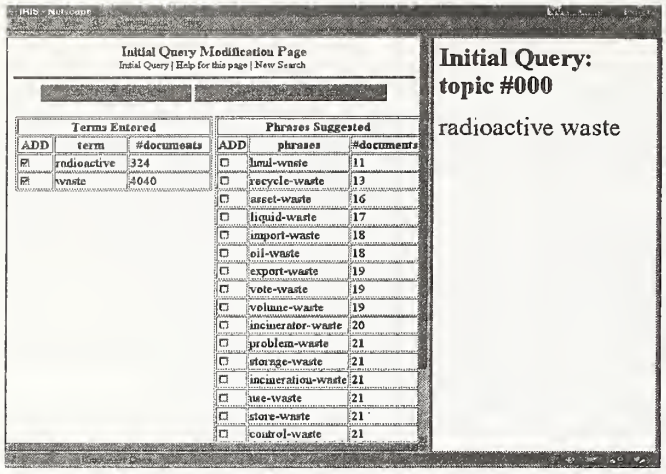


Figure 1.2 Initial Query Modification Interface for *irisa*

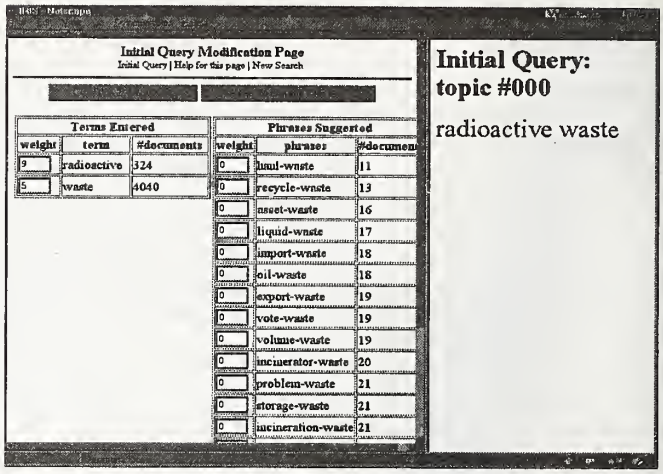


Figure 2.1 Relevance feedback interface for *iriss* and *irisa*

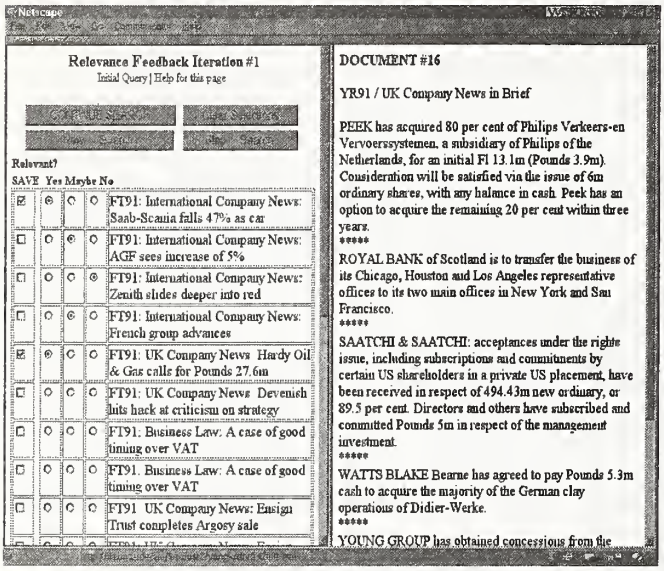


Figure 2.2 Relevance Feedback Interface for *irisp*

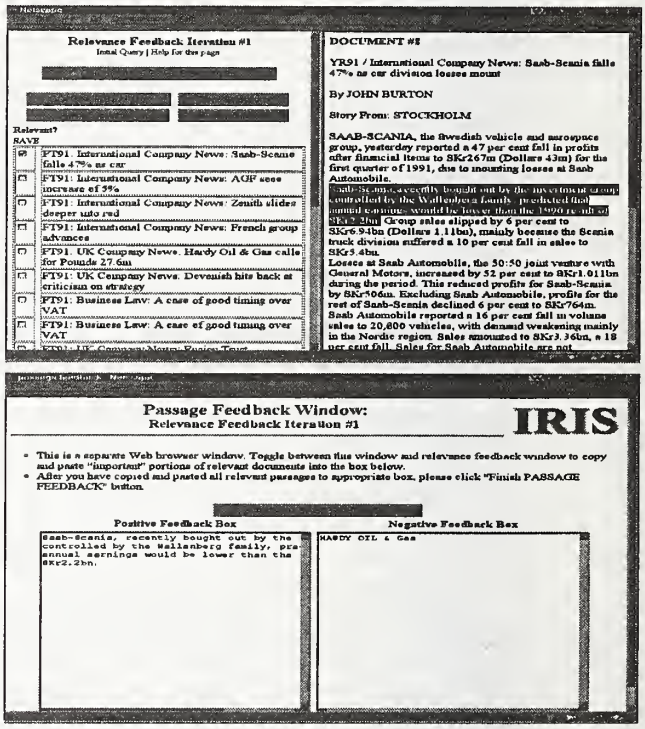


Figure 3.1 Feedback Query Modification Interface for *iriss* and *irisp*

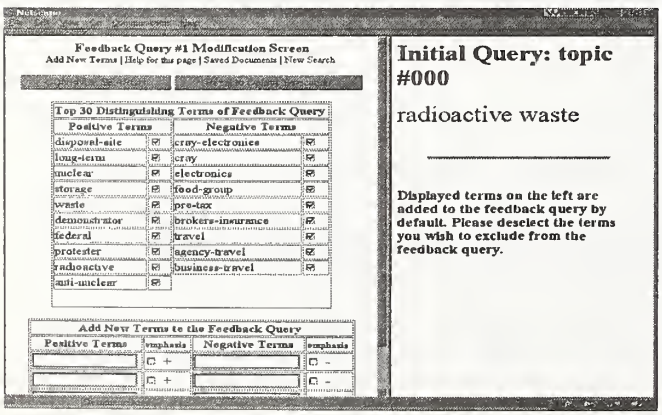
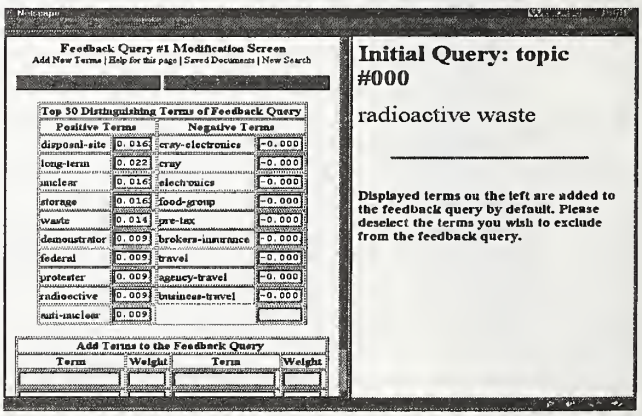


Figure 3.2 Feedback Query Modification Interface for *irisa*



Information Space Gets Normal

Gregory B. Newby
School of Information and Library Science
University of North Carolina at Chapel Hill
Chapel Hill, NC, 27573
gbnewby@ils.unc.edu

Experiments are presented based on unofficial results for TREC-7. Eigensystems analysis of a term co-occurrence matrix is compared to eigensystems analysis of a term correlation matrix. For each matrix type, the effect of term weighting and document length normalization is assessed. Recall-precision curves and other TREC statistics indicate that the use of the correlation matrix improves performance regardless of what term weighting or document length normalization is used.

Introduction

This paper presents unofficial results of an IR experiment conducted as part of TREC-7 participation. The system used, Ispace, was an adaptation of the author's prior work (Newby, 1997; Newby, 1996). There were three variables studied in the experiment:

1. Eigensystems of term a co-occurrence matrix versus eigensystems of a term correlation matrix
2. Term weighting versus no term weighting
3. Document length normalization versus no normalization

Title-only runs were also included. The overall retrieval approach is comparable to latent semantic indexing (LSI; see Deerwester et al., 1991). Rather than representing documents as a collection of the terms they contain, the approach of Ispace is to build an "information space" based on relations among terms in the collection, and then locate documents at the center of their terms. The effect is to base document relations (that is, their relative locations in the information space) on non-independent term relations. Information space is defined here as the contents and relations among them held by an information system (cf. Ingwersen, 1996).

The information space is derived from a matrix of pairwise term relations. This term by term matrix is subjected to an eigensystems analysis, which reduces the dimensionality of the matrix by identifying and collapsing linear trends. The resulting matrix may be trimmed at k dimensions to simplify computation and remove "noise" in the full matrix. Each term in the matrix may be thought of as a row (or vector) in the k-dimensional matrix of eigenvectors. Retrieval proceeds by locating queries in the information space and retrieving those documents closest to the query based on geometric (Euclidean) distance.

Conceptually, this process is one of representing a term by its relation to other terms. This is an important point of distinction from approaches which assume orthogonality among terms (such as Boolean systems and the basic vector space model, although term weighting explicitly addresses term relations).

The balance of this paper presents the methodology and results for assessing the value of term correlation versus co-occurrence in the context of term weighting, document length normalization, and title-only retrieval.

Methodology

The full TREC-7 collection was used with topics 351-400. Porter stemming was applied. Because a full information space cannot be built for the collection, due to the large number of terms (over 180,000 after stemming), a subset of terms was selected. Terms were selected by:

- Choosing only noun and adjective terms from the 50 topics, as identified by Eric Brill's part of speech tagger (Brill, 1994). After stemming, this yielded 535 unique terms.
- Building a full 180K by 180K term correlation matrix and selecting the most closely related terms for each of the topic terms identified.
- Deselecting stopwords (a version of the SMART stoplist was used)

- Keeping the most frequently occurring terms, regardless of part of speech. An additional 599 terms were added.

The result was 1134 terms for further consideration, which is a comfortable number for eigensystems analysis in a dense matrix. This number of terms is arbitrary, and was chosen to benefit from expanding on the terms in the topics without greatly overshadowing them. An 1134 by 1134 term co-occurrence matrix was extracted from the full 180K by 180K matrix. In the matrix, terms that occur in the same document are counted as having co-occurred (this is a relatively simple measure of term relatedness).

Two information spaces were built from this matrix. The first was simply an eigensystems analysis of the co-occurrence matrix. This matrix was about 30% sparse. The first 10 eigenvalues accounted for about 25% of the variance. 1134 dimensions were identified, although only the first 300 were used.

The second information space was from an eigensystems analysis of the Pearson product moment correlation scores from the co-occurrence matrix. This matrix was completely dense (no 0 cells). The first 10 eigenvalues accounted for about 25% of the variance. 1062 dimensions were identified (due to redundancy in the correlation matrix), although only the first 300 were used. 300 is an arbitrary cutoff value that has been used in past LSI research, but is not further investigated here.

All 535K TREC-7 documents were located in the information space at the center of whichever of the 1134 terms they contained, with 3 variations for each type of space:

1. A variation with no term weighting or document length normalization
2. A variation with term weighting only
3. A variation with term weighting and document length normalization

Each of topics 351-400 was then searched with Ispace to retrieve the closest documents in each space. Topics were subjected to the same condition as the space variation (weighting and/or normalization). In addition, a title-only version for each topic was run to assess the viability of the Ispace approach for shortened topics.

TF*IDF weighting was relatively simplistic, as taken from Frakes & Baeza-Yates (1992, pp. 372-375). Term TF weights assign a score to the importance of each term in a particular document. The formula used for TF weighting is:

$$tf_{ij} = [\ \log2\ (freq_{ij} + 1\ )]\ /\ [\ \log2\ (length_j)]$$

where:

$freq_{ij}$ = the frequency of the i'th term in the j'th document
$length_j$ = the number of terms in the j'th document

This formula has the property of ranging from greater than 0 to 1 (provided all documents have a length of at least 2 terms). Term IDF weights assign a score to the importance of a term in a collection. The formula used for IDF weighting is:

$$IDF_i = [\ (\log2\ N)\ /\ n_i\ ] + 1$$

where:

N = the total # of documents in the collection
n_i = the number of occurrences for the i'th term

This formula has the property of always being greater than 1. For the TREC-7 collection utilized here, the range was from 1 to about 21.

Results

These results were not judged by TREC assessors, so we cannot know for certain that a higher proportion of judged documents in the retrieved sets for each query would not have an impact. However, the results seem clear and in the direction anticipated, based on post-hoc analysis of TREC-7 Qrels files from NIST.

Table 1: Summary outcomes of different conditions. "Weight" is whether TF*IDF weighting was applied. "Normal" is whether unit length vectors were used. "Title" is whether only the topic title field was used. "Rel_Ret%" is the percentage of total relevant documents retrieved across all topics. "AveP" is the average precision (non-interpolated) across all relevant documents. "P @ 20" is the precision at 20 documents. "Rank" is how well this condition performed, based on AveP.

Matrix type	Weight?	Normal?	Title only?	Rel_Ret %	AveP	P @ 20	Rank
Co-occur	N	N	N	8.4	0.0060	0.0250	11
Co-occur	N	N	Y	9.9	0.0142	0.0190	9
Co-occur	Y	N	N	7.3	0.0047	0.0190	12
Co-occur	Y	N	Y	10.1	0.0118	0.0190	10
Co-occur	Y	Y	N	18.4	0.0254	0.0800	7
Co-occur	Y	Y	Y	12.9	0.0187	0.0450	8
Correlate	N	N	N	21.1	0.0807	0.2010	3
Correlate	N	N	Y	31.1	0.0452	0.1020	6
Correlate	Y	N	N	29.1	0.0731	0.1780	4
Correlate	Y	N	Y	29.7	0.0455	0.1010	5
Correlate	Y	Y	N	46.5	0.1476	0.3110	1
Correlate	Y	Y	Y	38.7	0.0958	0.1760	2

Table 1 shows that all conditions in which the correlation matrix was used, even with no document length normalization or weighting, outperformed all conditions when only the co-occurrence matrix was used.

Within each matrix type, the best performance was achieved with document length normalization and term weighting. Title-only runs under-performed relative to their full-topic counterparts, which tends to support notions of LSI-like approaches being particularly well suited to "query by example," when a long query or existing relevant document is available.

Figure 1: Recall-Precision curves for the various categories. "pc" indicates Pearson Correlation, "ev" indicates eigensystems on co-occurrence matrix only. N=normalization, W=weighting, t=title-only.

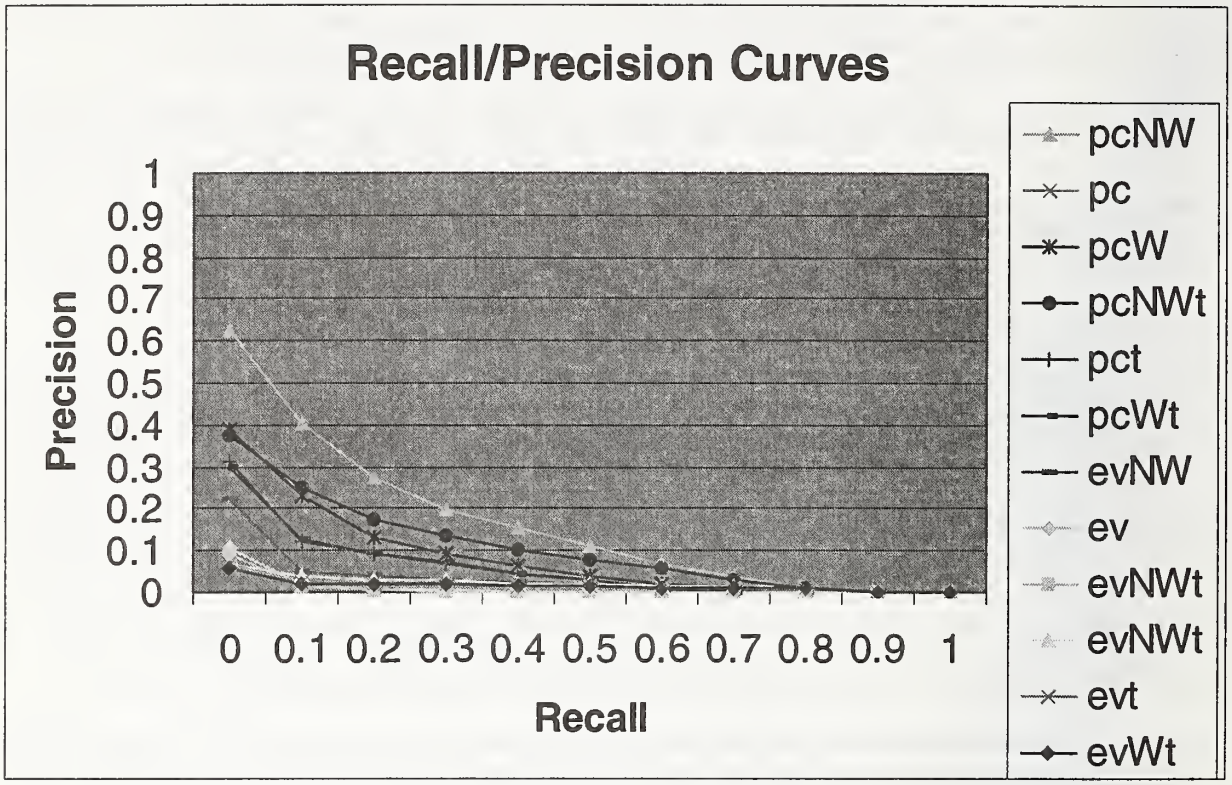


Figure 1 shows the relations among the various conditions graphically. Although the rankings are not quite the same as when only AveP is taken into account (Table 1), they are comparable and reinforce the value of term weights and document length normalization.

Conclusion

The benefits from document length normalization and term weighting that have proven themselves in many retrieval systems and have been shown to be effective here as well. Although this is confirmation of an expected result, not discovery of a new principle, the confirmation is a necessary step towards further progress. A reasonable expectation is that more advanced term weighting methods will further benefit retrieval effectiveness for the information space techniques discussed here.

A result that makes sense but had not previously been confirmed is the utility of computing the correlation matrix, rather than taking eigenvectors from the co-occurrence matrix alone. This approach, which is nearly identical to Principal Components Analysis (PCA) or Multidimensional Scaling (MDS, see Kruskal & Wish 1978), as employed in social science and elsewhere, pre-identifies linear term relations before the eigensystems analysis is performed. Although all the data required to build the correlation matrix is contained in the co-occurrence matrix, we add value by completing the correlation analysis.

Further research with Ispace will emphasize both practical and conceptual matters. Conceptually, modeling of the term relation process is in order. How is the LSI approach of examining a (sparse) term by document matrix different from the Ispace approach of analyzing a term by term matrix (the mathematical relation is well-known, but the conceptual relation less so)? What is the impact of choosing a far greater or smaller number of terms for inclusion in the information space? To what extent are non-orthogonal term vectors different than weighted but orthogonal term vectors as found in other IR approaches?

Practical matters need to focus on the efficiency of operation. Instead of only analyzing the documents that contain query terms, as is the case with almost all existing systems, Ispace requires evaluating each document in

the collection relative to a query – essentially, a nearest neighbor search with hundreds of thousands of items in *k*-dimensional space. Other practical issues include the implementation of modern term weighting schemes, instead of the simple scheme used here, and empirically determining good cutoff values for the number of dimensions to keep. Different approaches for measuring term co-occurrence may also prove valuable.

References:

Frakes, William B. & Baeza-Yates, Ricardo (Eds.). 1992. Information Retrieval Data Structures & Algorithms. Englewood Cliffs, New Jersey: Prentice-Hall.

Brill, Erik. 1994. "Some advances in rule-based part of speech tagging." Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), Seattle, Washington.

Deerwester, Scott; Dumais, Susan; Landauer, Thomas K.; Furnas, George W.; & Harshman, Richard A. (1990) "Indexing by latent semantic analysis." Journal of the Society for Information Science 41(6), 391-407.

Ingwersen, Peter. 1996. "Cognitive Perspectives of Information Retrieval Interaction: Elements of a Cognitive IR Theory." Journal of Documentation 52 (1): 3-50.

Kruskal, Joseph B. & Wish, Myron. 1978. Multidimensional Scaling. Beverly Hills: Sage.

Newbym Gregory B. 1996. "Metric Multidimensional Information Space." Text REtrieval Conference (TREC-5) Proceedings. Gaithersburg, MD: National Institute of Science and Technology.

Newby, Gregory B. 1997. "Context-Based Statistical Sub-Spaces." Text REtrieval Conference (TREC-6) Proceedings. Gaithersburg, MD: National Institute of Science and Technology.

ClickIR: Text Retrieval using a Dynamic Hypertext Interface

Richard C. Bodner and Mark H. Chignell

Interactive Media Laboratory
University of Toronto
5 King's College Road
Toronto, Ontario
Canada M5S 3G8
{rbodner, chignel}@mie.utoronto.ca

Abstract

In this report we describe our model of dynamic hypertext and how the ClickIR system uses this model to assist users in interactive search. The system was used in both the ad hoc task and the interactive track. In the context of the ad hoc task we were interested in the effects relevance feedback would have on our system. Comparison of ClickIR performance with and without relevance feedback showed that relevance feedback was critical in boosting the performance of the system from below median performance to the upper rank of TREC-7 systems. In the interactive track we compared the ClickIR (experimental) system where the tasks of querying and browsing were integrated, with a system which closely approximated a Web search engine, where the task of querying is separated from the task of browsing a list of hits. A trade-off between recall and precision was observed, with ClickIR leading to significantly greater recall, but at the expense of significantly lower precision and longer time taken to perform the task.

1.0 Introduction

The Interactive Media Lab at the University of Toronto participated in the manual ad hoc task and interactive track of TREC-7, following on from earlier participation in TREC-3 and TREC-4 (Charoenkitkarn *et al.*, 1995; Charoenkitkarn *et al.*, 1996). Our approach in these studies has been to support a person's decision-making ability while relieving some of the cognitive load placed on a person during searching (e.g., the tasks of querying and browsing). In TREC-3 and TREC-4 we used a query-based markup approach where searchers could mark up queries directly on the text of documents through click and drag operations. Since that time we have moved from using systems where queries are marked up on text to systems where queries are inferred from selections of text within documents. Instead of expressing queries, the searcher then simply has to click on sections of text to indicate his or her interest, and the system infers a query. For TREC-7, we were interested in comparing our model of dynamic hypertext with the functionality of a typical Web search engine. Our model of dynamic hypertext attempts to blend the tasks of querying and browsing whereas a standard search engine typically separates these tasks.

2.0 Dynamic Hypertext Information Retrieval Model

Standard, static hypertext documents have a number of well-known problems such as link maintenance, the lost-in-hyperspace problem (Conklin, 1987), and a finite structure. The last of these problems is probably the greatest of all since an author of a hypertext document cannot implement all possible links required by all possible visitors to that document. This problem leads to many unconnected small groups of hypertext documents. These groups are difficult to navigate. This has lead to the need for search engines to assist in navigation. The use of search engines creates a "spiky" navigation pattern (see Campagnini and Ehrlich, 1989; and Parunak, 1989 for a description). Due to this spiky pattern, users navigate in two modes:

searching (e.g., querying a search engine) and browsing (e.g., reviewing the documents retrieved based by the previous query).

Our model of dynamic hypertext attempts to blend these two modes together via the user interface. In our approach there is no notion of a static link; links are created on the fly based on knowledge of the corpus and the interests of the user. We infer the user's interests by recording his/her interaction (links clicked on) with the system. Various query formulation algorithms can be used to infer a query once a link or a section of text has been selected. In the current implementation of the ClickIR system, the sentence that the link occurs in is sent to the search engine as a query. It is assumed in this approach that the user selected the link due to an interest in the content surrounding the link. Thus the dynamic hypertext acts like a form of sentence-based relevance feedback.

In ClickIR, markup of links within text is dynamic. Words are selected as links based on the user's previous interactions with the system. The words in the previous queries are used as seeds for selecting the future links. A "tail" representing a weighted average of the three most recent queries is used in selecting terms to be highlighted as links, and in modifying the query formed after a link is selected. This "tail" is so named because it adds inertia to the process of switching from one topic or class of query to another. The number of previous queries used in forming the tail is a parameter that can be changed. The weighting vector used to capture the diminishing importance of older queries relative to more recent queries can also be changed. The value currently used to weight the important of past queries (thereby defining the "tail size") was arrived at through informal experiments carried out in our lab. For a further description of our dynamic hypertext model see Bodner *et al.*, 1997 and Tam *et al.*, 1997.

3.0 System Descriptions

The Interactive Media Lab implemented two systems specifically for participation in TREC-7. For the ad hoc task, only the experimental system (ClickIR) was used. In contrast, the interactive track required both a control and an experimental system. Both the control and experimental systems used the Inquery (version 3.1) search engine software (Caltan *et al.*, 1992). Our systems provide a hypertext user interface to the search engine. This was accomplished through the use of CGI scripts. The scripts convert the user's interactions with the system into queries, which are sent to Inquery. The query results are then converted into hypertext documents. Neither the experimental nor the control system allowed the use of any Boolean or special search operator. Users were only allowed to enter natural language queries.

3.1 The Control System

The control system used in the interactive track was designed to mimic the hypertext interface provided by most search engines on the Web today. The typical search engine interface separates the tasks of querying and browsing. The user must constantly switch between modes during a search session. In our control system the user was presented with a startup screen where he/she could enter an initial query. From the initial query a resulting list of document titles, which were linked to the full document text, were presented in ascending rank order (provided by Inquery). On this screen, the user could also enter new queries to continue the search session (see Figure 1). When the user selected a title link, he/she entered the document view screen. On this screen there was a link that users could click on to mark the current document as being relevant to the current search and there was another link which led to a review of the user's relevant document list. The user iterated between these two screens during the search session (task switching between querying and browsing).

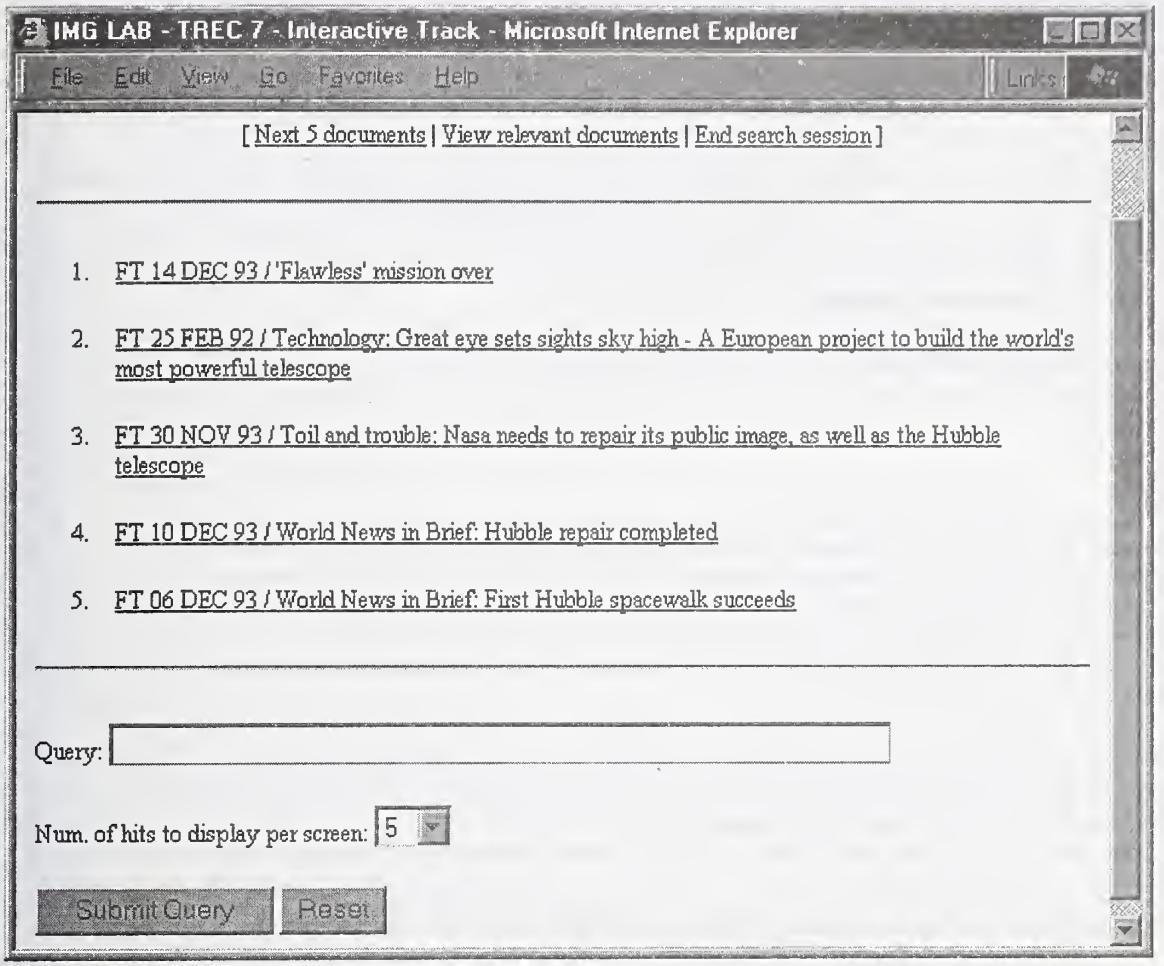


FIGURE 1. Result list screen for the control system.

3.2 The Experimental System

The experimental system, known as ClickIR, was used in both the interactive track and the manual ad hoc task. The experimental system implemented our model of dynamic hypertext (described above in Section 2.0) in which the tasks of querying and browsing are blended together. As with the control system, the user was presented with an initial query screen. The query results screen (see Figure 2) displayed the entire text of one or more documents. Although the user could select the number of documents to display per results page, the default was to display two documents. Through informal pilot studies this number of screens was found to be easiest to handle, both in terms of scrolling and in terms of minimizing information overload. The CGI scripts collected the results from Inquery, and used the document context and terms found in the user's queries as sources of information to guide the markup of the documents, which were then presented as dynamic hypertext documents. The user queried the system by clicking on hypertext links in the documents. The system also provided a form to enter a new query or expand the current query. This form was included because the conditions imposed by TREC did not allow for the system to be primed with terms in order to provide more appropriate links to the user. As with the control system the user could "page" through the query result set, mark documents as being relevant, and review their relevant document list. The user did not switch between querying and browsing tasks in the experimental system.

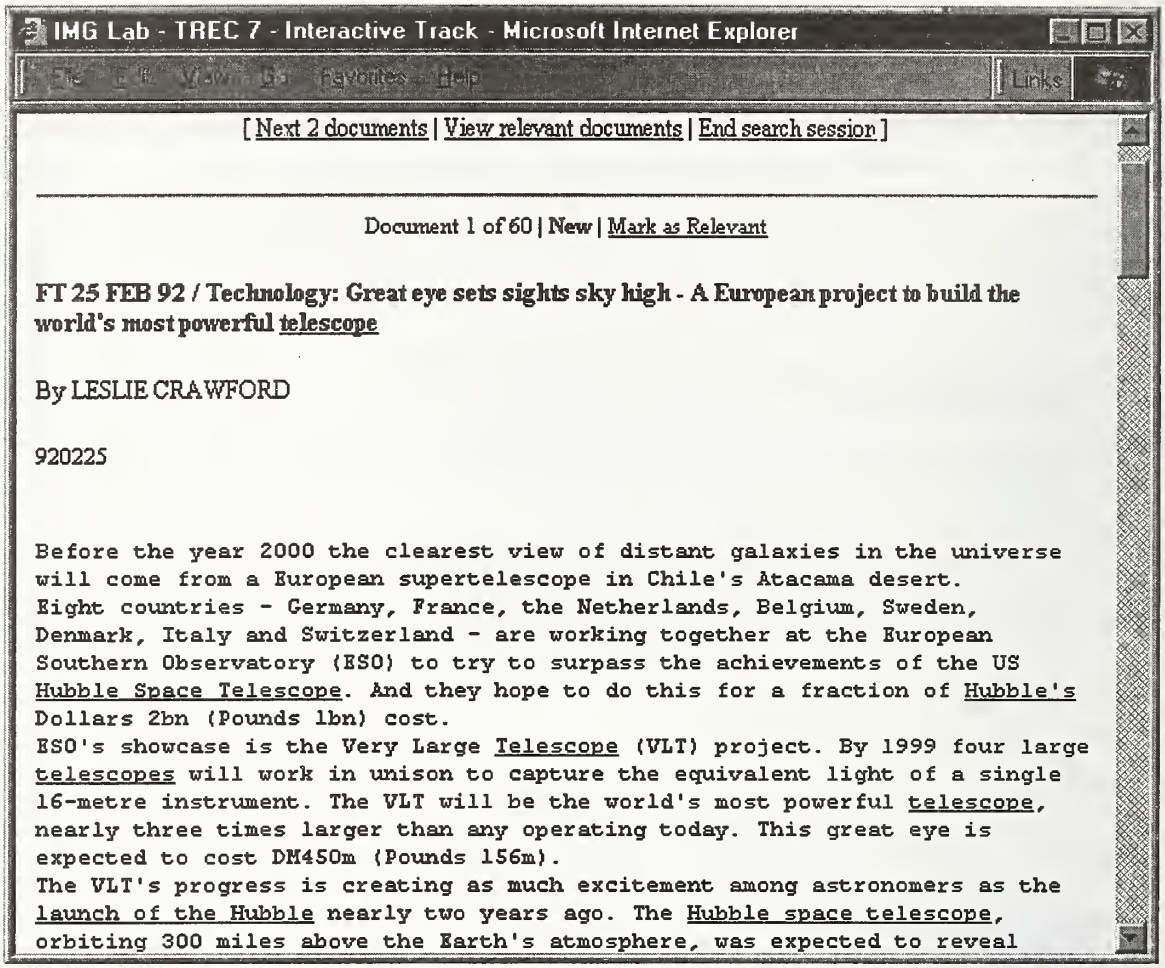


FIGURE 2. Document view screen for the experimental system (ClickIR).

One of the issues with our experimental system that we investigated at TREC-7 was phrase generation. In previous versions of our system, links were simply single terms (words) and terms were selected as links based on the product of term frequency and inverse document frequency (TF*IDF). Using this metric an upper and lower threshold could be set in order to limit or expand the number of links that were generated. This threshold method did not adequately address the problem of multiple occurrences of a term being highlighted multiple times in the same document. A user with only standard hypertext experience would then have difficulty distinguishing between the different occurrences of the term (i.e., the term/link would appear to point to the same end node, given experience with how standard hypertext tends to work). To try to solve this problem a phrase generation technique was used. We expected that the use of a phrase instead of individual terms would help users by providing more discriminable links. For example, if the term "retrieval" appeared in multiple locations within a document, a user might assume that each occurrence points to the same endpoint. In contrast, if one occurrence of the term occurred in the phrase "information retrieval" and another occurred in "retrieval of documents", the user should then be able to distinguish the different links.

Given time contrasts, we could only implement a phrase generation method based on simple heuristics to create the phrases. The heuristics operated on the following assumptions:

- A phrase is two or more words containing two or more content bearing terms. A content bearing term is defined simply as a term that is not a stopword.

- A phrase can have at most two stopwords between content bearing terms.
- A phrase must begin and end on content bearing terms.

4.0 Manual Ad Hoc Task

Although ClickIR was not designed for handling tasks such as TREC manual ad hoc, we used ClickIR in that task in an attempt to understand how relevance feedback would affect the performance of our system. The two runs we submitted to TREC were `uoftimgr` and `uoftimgu`. Only the first of these runs used relevance feedback.

As described in Section 2.0, our dynamic hypertext information retrieval model uses sentence-based relevance feedback interactively during search. The interactive search was combined with a batch search in order to conform with the requirements of the TREC manual ad hoc task, as will be further discussed in Section 4.1 of this paper.

4.1 Query Generation Process

Since ClickIR is an interactive system, the searcher was given approximately 15 minutes per topic to browse the document collection. As the searcher browsed the collection, the links that he/she clicked on were recorded (called the "interaction record"). The searcher was also asked to select documents that "seemed" relevant to the topic. Given the amount of data in the TREC-7 ad hoc collection, we felt that searchers would be able to get a sense of the type of documents the collection contained even if they did not find a large number of relevant documents.

The searcher's interaction record was then used to build a query by collecting all the sentences for the links that were clicked on (selected). The resulting query was then sent to Inquery and the top 1000 documents were selected. This was how the uoftimgu queries were generated. The uoftimgr queries also contained the queries generated from the interaction record, but in addition, the documents marked as relevant during the search were used for relevance feedback.

4.2 Results and Discussion

We found a significant difference in average precision between the uoftimgr and uoftimgu runs ($t[49]=2.18$, $p<.05$). As expected, the relevance feedback runs performed better than the simple interaction record runs. The average precision for all topics for uoftimgr was 0.276 and for uoftimgu the average precision was 0.245. Overall, the uoftimgr run retrieved 60.3% of the total relevant documents identified by TREC and 5.6% of the total documents retrieved by the system were relevant. The uoftimgr run retrieved 55.3% of the relevant documents and 5.2% of the total documents retrieved were relevant. Figure 3 shows the recall-precision curves for both runs. While this difference may not sound like much, in the context of the manual ad hoc task in TREC-7 it meant the difference between a system that was average and a system that was one of the best.

An alternative way of viewing the effect of relevance feedback is shown in Table 1. When relevance feedback was used, the average precision was above the median (i.e., the median average precision score for all 17 research groups) for 31 or 62% of the 50 topics. However, when relevance feedback was not used, the system performed better than the median only 44% of the time (or for 22 of the 50 topics). Further evidence of the major impact of relevance feedback was shown by the fact that, on average, five additional relevant documents were retrieved using relevance feedback (an average of 56.9 with, versus 51.7 without, relevance feedback) which was a statistically significant difference ($t[49]=2.04$, $p<.05$).

TABLE 1. System performance based on average precision median for all topics in the manual ad hoc task.

		Without Relevance Feedback (uoftimgu)		
		Better	**Worse**	
With Relevance Feedback (uoftimgr)	**Better**	18	13	31/50
	Worse	4	14	18/50
		22/50	27/50	

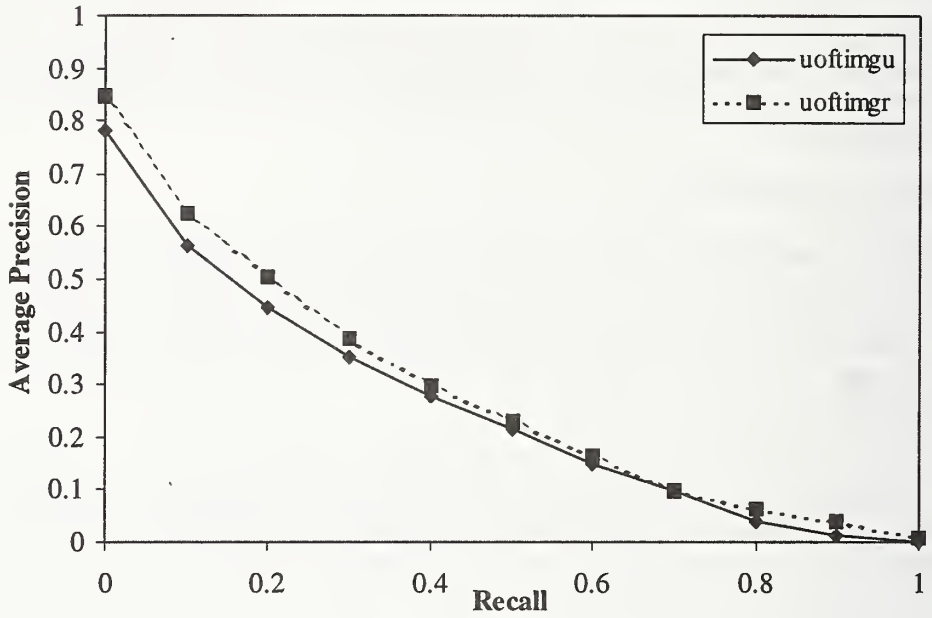


FIGURE 3. Recall-precision curves for uoftimgu and uoftimgr runs.

5.0 Interactive Track

For our participation in the interactive track, we were interested in comparing the dynamic hypertext information retrieval model implemented as ClickIR (the experimental system) with an interface which separated the tasks of querying and browsing such as found in current Web search engines (the control system).

5.1 Experimental Design

The design of the experiment in this section of the study followed the design required of the TREC-7 interactive track participants.

The eight search topics were ordered into two distinct sequences of four topics each. Within each sequence, the ordering of the topics was fixed, but the ordering of the sequences was counterbalanced, so that half of the subjects worked on one sequence of four topics followed by the other, while the remaining subjects worked on the two sequences in the reverse order.

The other factor that was manipulated in the experiment was the type of system used. The interactive experiment used ClickIR as the experimental system (as described in section 3.2). Note that ClickIR is referred to as "System A" in our online reports and the control system is referred to as "System B". The performance with this experimental system was contrasted with the control system (described earlier in Section 3.1). Half of the subjects used the experimental condition first for four trials (one of the two sequences) followed by the control condition (with the four tasks from the other sequence) and half used the two systems in the reverse order.

There were a total of four possible combinations of sequence and system. Each subject was presented with two of these four possible combinations for a total of eight trials each (two combinations with four search topics per combination).

For the purposes of analysis and interpretation, the search topics were classified according to their average level of difficulty as found by averaging the results of all the interactive track participants. Topics with an average recall of 0.3 or better were classified as "easy" and topics with lower recall were classified as "difficult". The analysis then focused on the impact of system and topic difficulty on search results.

Measures collected during the search included the number of documents viewed, the overall time taken, the time taken to find the first relevant document, and the recall and precision achieved. In addition, a standard set of six questions (involving ratings on a five point rating scale) was completed at the end of each search.

In addition to the measures collected during and immediately after the searches, a pre-experiment and a post-experiment questionnaire was administered. The pre-experiment measurements included the word associations (FA-1) task that was selected by the organizers of the track.

5.2 Subjects

The eight subjects that participated in the experiment were selected from the University of Toronto undergraduate and graduate student populations. There were three female subjects with an average age of 24 and an average FA-1 pooled score of 38. The five male subjects had an average age of 32 and an average (mean) FA-1 pooled score of 32.8. None of the subjects had previously participated in a TREC searching study and all of the subjects reported having a high degree of experience searching the Web.

5.3 Results

Multivariate analysis of variance was used to assess overall effects. There was no significant interaction between system and topic difficulty. However, there were significant main effects for topic difficulty and for system. The nature of these effects was then assessed using a fully within two factor (system by topic difficulty) analysis of variance for each of the measures. For topic difficulty, there were significant differences on a number of the measures which were in the expected direction, indicating that our results agreed with the general ordering of topic difficulty that was found across the groups participating in the TREC-7 interactive track.

For the system main effect, there were significant ($p<.05$) effects for recall, precision, and total search time and borderline significant ($p<.10$) effects for time to first relevant document, and for questions 2 through 6 of the post-search questionnaire. The experimental system had significantly higher ($F[1,7]=42.74$, $p<.001$), recall than the control (mean of .41 vs. a mean of .37), but at the expense of significantly lower ($F[1,7]=35.66$, $p<.001$) instance precision (mean of .65 for instance precision for the experimental system versus .70 for the control system). There was no significant difference in overall time taken by the two systems (with searches frequently taking the complete 15 minute maximum that was allotted), neither was there a significant difference in the time taken to find the first relevant document ($F<1$ in both cases).

ANOVA tests on the post-search questions should be interpreted cautiously, since the 5-point Likert scale responses on each question are not continuous normally distributed variates. However, the ANOVA results are presented here in the spirit of generating hypotheses for further research (with a skeptical stance concerning the precise values of F and p for each test being assumed). That being said, the effect of system

on question 2 ("was it easy to get started on this search") was borderline significant ($p<.10$) with a tendency for subjects to judge the control system as being easier to get started with. There was no significant difference in satisfaction with the results obtained when using the two systems (F was approximately 1 for the comparison on question 4). There was also no significant difference in their confidence that they had identified all the different instances of each topic (question 5). There was however a borderline ($p<.10$) effect for question 6 ("did you have enough time to do an effective search") with more participants tending to feel that they had enough time with the control system. From a methodological standpoint, it is interesting to note that this subjective question was more effective to a possible difference in how much time was needed to search using each system than was the actual measure itself.

From the exit questionnaires, subjects preferred the control system in terms of ease of learning (5/8). Ease of use was evenly split between the two systems, and 6 out of the 8 subject reported that they liked the experimental system the best.

5.4 Discussion

These results indicate that the experimental system promoted recall at the expense of precision. Earlier studies in our laboratory have found that search experts tend to have higher precision, but lower recall, than experts (Charoenkitkarn, 1996; Golovchinsky, 1997). In contrast, this studies showed (using a within-subjects design) that two different interfaces tended to move participants (as a group) to different points on a trade-off between recall and precision. Golovchinsky (1997, p. 120) classified his experimental subjects as "skimmers" (i.e., people who make many interactions with the system during a session) or "readers" (i.e., people who spend more time reading articles and making careful judgements). One intriguing possibility for future research is that the point and click nature of the experimental system encouraged a high degree of interactivity or "skimming" and that as with Golovchinsky's subjects, higher recall resulted.

An alternative viewpoint that deserves further study stems from the fact that the experimental system made more text available for a longer time within the experiment. As a result, there may be more incidental learning about the topics in the text in the dynamic hypertext version of the retrieval system than there is with a standard Web search interface.

The user interface for the experimental system was designed for ease of use. Its browsing interface can function as both a dynamic hypertext system and as a user interface for text retrieval. In an earlier study, the query-based dynamic hypertext was compared with a static hypertext (Tam, 1997; Tam *et al.*, 1997). Tam found that searching for information using the experimental interface was easier for computer/domain novices than searching for the information in a static hypertext version of the information. For experts, however, there was no significant difference in the level of performance obtained with the dynamic version of the hypertext (experts were neither helped nor hindered by the interface).

In the present study, the control system represented a highly familiar search engine interface for all of the experimental subjects. Thus it might be supposed that the novel experimental system would be at a natural disadvantage when compared with the familiar control system. In order to discount this possibility, a longitudinal study would be required where subjects use the experimental system for an extended period of time to see if their performance improves as they gain more experience with the system. However, such an analysis was outside the scope of the experiment defined for participants in the interactive track of TREC-7.

The results obtained in this study must be considered tentative due to the relatively small sample of subjects used and to the small amount of experience they had with the experimental system in contrast to their experience with the search engine interface. In spite of this caveat, there is a clear tendency for the experimental interface to promote recall at the expense of precision. This suggests that the experimental interface will prove useful in situations where recall is emphasized. Possible areas where this is the case may include patent searches and exhaustive literature reviews. In addition, it is expected that people will benefit from reading more document text, as is likely to occur in the dynamic hypertext interface. However, the demonstration of learning as a supplement to retrieval was outside the scope of this study and represents a hypothesis to be tested in subsequent research.

6.0 Conclusions

This study was the first time that a query-based dynamic hypertext interface has been tested under TREC conditions. Earlier participation by our research group had used visual mark-up based querying, which stopped short of the point and click link selection method introduced in this study of large scale text retrieval. The tendency for different interfaces/systems to produce a trade-off in recall vs. precision may provide a useful stimulus for further research. Understanding how and why this trade-off occurs may provide fundamental insights into how search behaviour changes depending on the type of system and interface. One suggestion is that the increased emphasis on recall at the expense of precision, with the dynamic hypertext system, is due to the increased availability of the text, and to the way in which query intent is expressed with respect to the text.

The experimental system yielded surprisingly good results in the ad hoc section of the TREC-7 competition, which is a notable result for a system that emphasizes interactive search over complex computational techniques. However, it should be noted that two sets of outputs were submitted to TREC, those that were output by the experimental system, and those that were "boosted" through relevance feedback. That boosting was based on the relevance judgements (document selections) made by the two subjects used in the ad hoc phase of the study. The comparison of the regular and boosted results showed that boosting with relevance feedback did in fact significantly improve the average precision, and is a useful supplement for a query-based dynamic hypertext system, where relevance judgements can be collected as an unobtrusive by-product of the interaction with the system.

The present findings concerning relevance feedback demonstrate that it makes an important improvement to the effectiveness of query-based dynamic hypertext. In addition, the findings support the notion that differences between the average precision obtained by different research groups are small enough that small differences in average precision can lead to fairly major changes in the ranking obtained in the TREC conference.

Acknowledgements

The authors would like to thanks the subjects that took part in the Interactive track portion of the study. We would also like to thank Rick Kopak for his assistance in conducting the manual ad hoc searches. This project was funded in part by National Science and Research Council of Canada (NSERC) and Communications and Information Technology of Ontario (CITO).

References

Bodner, R., Chignell, M. and Tam, J. (1997) Website authoring using dynamic hypertext. In *Proceedings of Webnet'97*, Toronto: Association for the Advancement of Computing in Education, 59-64.

Caltan, J. P., Croft, W. B., and Harding, S. M. (1992). The INQUERY retrieval system. In *Proceedings of the 3rd International Conference on Database and Expert Systems Applications*, 78-82.

Campagnini, F. R. and Ehrlich, K. (1989) Information retrieval using a hypertext-based help system. *ACM Transactions on Office Information Systems*, 7(3), 271-291.

Charoenkitkarn, N. (1996). *The effect of markup-querying on search pattern and performance in large-scale text retrieval.* Unpublished Ph.D. dissertation. Department of Industrial Engineering, University of Toronto.

Charoenkitkarn, N., Chignell, M.H., and Golovchinsky, G. (1995). Interactive exploration as a formal text retrieval method: How well can interactivity compensate for unsophisticated retrieval algorithms?

Pages 179-199. D.K. Harman (Ed.), In *Proceedings of the Third Text Retrieval Conference (TREC-3).* National Institute of Standards Special Publication 500-225. Gaithersburg, Maryland.

Charoenkitkarn, N., Chignell, M.H., and Golovchinsky, G. (1996). Is recall relevant? An analysis of how user interface conditions affect strategies and performance in large scale text retrieval. In D.K. Harman (Ed.), *Proceedings of the Fourth Text Retrieval Conference (TREC-4).* National Institute of Standards, Gaithersburg, Maryland.

Conklin, J. (1987, September) Hypertext: an introduction and survey. *Computer*, 20(9), 17-41.

Golovchinsky, G. (1997). *From information retrieval to hypertext and back again: the role of interaction in the information exploration interface.* Ph.D. Dissertation, Department of Mechanical and Industrial Engineering, University of Toronto.

Parunak, H. V. D. (1989) Hypermedia topologies and user navigation. In *Proceedings of Hypertext'89.* Pittsburgh, PA: ACM Press, 43-50.

Tam, J. (1997). *Design and evaluation of web-based dynamic hypertext.* Ph.D. Dissertation, Department of Mechanical and Industrial Engineering, University of Toronto.

Tam, J., Bodner, R. and Chignell, M. (1997) Dynamic hypertext benefits novices in question answering. In *Proceedings of the Human Factors and Ergonomics Society 41st Annual Meeting*, 350-354.

Text Retrieval via Semantic Forests: TREC7

Gregory D. Henderson, Patrick Schone, Thomas H. Crystal*
U.S. Department of Defense
Speech Research Branch
Ft. George G. Meade, MD 20755-6000

1. INTRODUCTION

In the second year of the use of Semantic Forests in TREC, we have raised our 30-document average precision in the automatic *Ad Hoc* task to 27% from 19% last year. We also contributed a significant number of unique relevant documents to the judgement pool [3]. Our mean average precisions on the SDR task are roughly the median performance for that task [4].

The Semantic Forests algorithm was originally developed by Schone and Nelson [1] for labeling topics in text and transcribed speech. Semantic Forests uses an electronic dictionary to make a tree for each word in a text document. The root of the tree is the word from the document, the first branches are the words in the definition of the root word, the next branches are the words in the definitions of the words in the first branches, and so on. The words in the trees are tagged by part of speech and given weights based on statistics gathered during training. Finally, the trees are merged into a scored list of words. The premise is that words in common between trees will be reinforced and represent "topics" present in the document. With minor modifications, queries are treated as documents.

Seven major changes were made in developing this year's system from last year's. (1) A number of pre-processing steps which were performed last year (such as identifying multi-word units) were incorporated into Semantic Forests. (2) A part of speech tagger was added, allowing Semantic Forests to use this additional information. (3) Semantic Forests distinguishes between queries and documents this year, since our experiments indicated they needed to be treated differently. (4) Only the first three letters of words which do not occur in the dictionary are retained, instead of the entire word. (5) A parameter directs Semantic Forests to break each document into segments containing at most a set number of words, typically 500. (6) The algorithms used by Semantic Forests to assign and combine word weights have been improved. (7) Quasi-relevance feedback was implemented and evaluated.

2. GENERAL SYSTEM OVERVIEW

Our overall system design is shown in Figure 1. First a filter is applied to the SGML documents and queries. For the documents, this removes any extraneous header and footer information and expands some common contractions (such as "u.s." into "united_states"). For the queries, we also remove some of the commonly used phrases which occur in queries, such as "relevant documents contain" or "would not be considered relevant". Although we experimented with a form of natural language processing for the queries, and feel that this should produce significant results, the filtering we used this year differs very little from last year. This year the query filter also adds a number of "class" words to the query documents. These words appear in the dictionary as a list of "synonyms" and are described in more detail in Section 3.

Next, Semantic Forests is used to produce a list of topic words for each document segment or query. The topic lists for the document segments are placed in a database, which organizes the information to allow for the efficient retrieval of all document segments having a given word in their topic lists. No significant change was made in this step from last year.

Given the database and the query topic lists, we produce a list of hopefully relevant documents by scoring the document topic lists against the query topic list. Our query system without feedback is very similar to last year's system. The system with quasi-relevance feedback adds an initial rough scoring pass with the original query topic list, after which the topic list is modified and the final scoring pass is performed.

*T.H.Crystal is an integree from the IDA Center for Communications Research, Princeton, NJ.

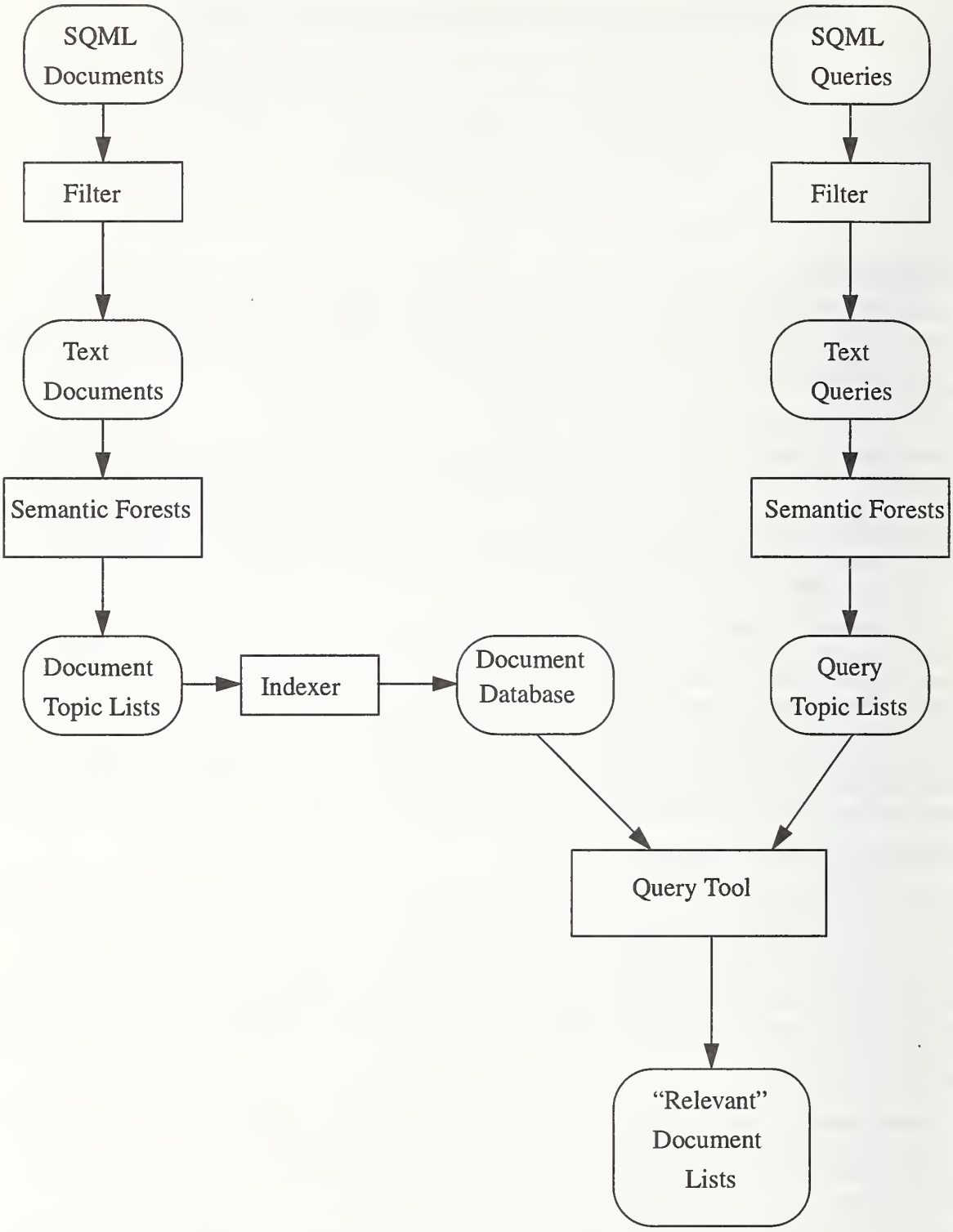


Figure 1. Schematic of the information retrieval system.

3. SYSTEM DETAILS

The Semantic Forests IR system was described in some detail in last year's conference report [2], so this section details the changes that were made for this year.

3.1. Class Words

We added a new type of topic word (implemented as the part of speech "CLASS") this year. A class is essentially a list of topics which are, in a loose sense, synonyms of the class word. For example, the class "1970s" is the list

'70s 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979.

We reasoned that any document which contains 1972, for example, is relevant to the 1970s. This was an attempt to get "topics" instead of "words" in our topic lists, but the number of classes was somewhat limited (a total of 200). Our classes were created semi-automatically: some, like 1970s, were entered by hand based on our experience; the rest were hand selected from an "inversion" of the dictionary. For instance, the definitions of argon, hydrogen, bromine, chlorine, xenon, helium, fluorine, oxygen, neon, and nitrogen all contain the phrase "gaseous element", so a class "gaseous_element" was created containing those words. To be used effectively, however, the filter which is applied to the incoming text must insert class words when appropriate. Our experiments indicate that this idea can improve performance significantly, but we did not have the opportunity to experiment further.

3.2. Topic Labeling

Significant portions of Semantic Forests were rewritten this past year to improve efficiency and performance. The original version could handle multi-word units (such as united_states) only if they had been entered into the dictionary and joined when the documents were filtered, while the new version recognizes and joins groups of words in the text which are identified as multiword units in the dictionary. The new version also handles words not found in the dictionary differently. For instance, if "Hubble" is not in the dictionary, "hub+unk" is inserted and will match any word with first three letters "hub". This is a compromise that preserves information that would otherwise be lost without adding large numbers of new words and swamping the system. It also provides a primitive stemming for out-of-dictionary words.

Since the definition of a word depends on the way it is used, Semantic Forests should at least be able to identify parts of speech. This, however, requires error free text and natural language processing. Even though Semantic Forests was developed originally to work with imperfect speech recognizer transcripts and, given enough topic relevant text, should sort out the correct definition on its own, we felt that its performance on the *Ad Hoc* task would be improved by a part of speech tagger. To this end, a Bayesian network was added to try to identify parts of speech. While this seemed to help overall, mistakes were made. In a query about "tornados touching down", for instance, "down" was tagged as a noun and "touching" as a descriptive adjective, so "emotion" and "poignant" were given as topic words.

The algorithms for assigning weights to the root and branches of the word trees and for merging the word trees into a topic list were also changed this year.

3.2.1. The Weight of Root Words

Three statistics are used to determine the weight of a root word: *avg*, the average number of occurrences in those training documents containing the word; *fract*, the fraction of training documents which contain the word; and *beta*, the relative importance of the part of speech of the word. Given these statistics, the weight given the root word is

$$wt(word) = beta \cdot avg^2 \cdot \left[\cos\left(\frac{\pi}{2} \cdot min(1, 3 fract)\right)\right]^4 . \qquad (1)$$

This rewards words which occur frequently in a low fraction of documents as valuable parts of speech (such as proper nouns) and replaces Equation (2) in last year's report.

3.2.2. The Weight of Branch Words

The change here is in the "dictionary salience", *D*, between a child word and its parent. This replaces Equation (4) in last year's report. First a "Z-score" is computed:

$$Z = \left(\frac{\sum freq(parent) \cdot prob(child)}{\sum freq(parent)} - \mu(child)\right) / \sigma(child) \qquad (2)$$

The sums are over all training documents and

$freq(parent)$ is the number of times the parent word appears in the training document.

$prob(child)$ is the probability that the child word appears in the training document.

$\mu(child)$ is the expected value of the probability that the child appears, evaluated over all training documents.

$\sigma(child)$ is the standard deviation of the probability that the child appears, evaluated over all training documents.

A threshold, *Thres*, is set for each child word and

$$D(child, parent) = \begin{cases} wt(child) \cdot \frac{1}{2}(1 + erf(Y))\sqrt{Y} & Y \geq 0 \\ 0 & otherwise \end{cases} , \quad (3)$$

where $Y = Z - Thres$ and the salience of all the root word's offspring are used to distribute the parent's weight to the offspring:

$$wt(child, parent) = \frac{W \cdot D(child, parent)}{\sum D(offspring, parent)} \cdot wt(parent) , \quad (4)$$

where the sum is over all offspring of the parent, and *W* is an attenuation coefficient. This algorithm is designed so that the sum of the weights of the offspring is a fixed fraction (*W*) of the parent's weight.

3.2.3. Merging Trees into a Topic List

The simplest way to combine the trees into a topic list is to sum the weights for each word that occurs in a tree. In practice, however, this does not sufficiently emphasize words in common between trees. This year we used two different methods for merging trees: the first is used for documents, and the second for the queries. The first changes the weights of the words, while the second prunes out words which don't occur close together. Our experiments indicate that this combination of algorithms performs best. The words are then sorted into a topic list and the words which are not in the original text are marked as children words. For both methods we start with a list of trees, *1,2, ...n, ... N*, where the root of the *n*th tree is the *n*th sequential word in the given text.

3.2.3.1. Documents

For documents, we compute a modified weight for each word in each tree, then sum the modified weights for each word to produce the score for that word in the document. The modified weight for a word in tree *n* is the sum of the weights for that word in trees *n* through *n+29*, unless the word is a non-root word which occurs only once in those trees. In that case the modified weight is zero.

3.2.3.2. Queries

In this method, we prune the trees of words which are not considered salient. To be salient, a word must be either the root word of the tree (the one that occurs in the query), or a child word which occurs as a child of a different parent in the tree of one of the next 29 trees. This prevents over emphasis of children words simply because their parent occurs frequently.

Next, the words in the pruned trees for the entire query are scored. It is possible for a word to occur more than once in any given tree, so we start by summing the weights for each occurrence to get a score for each word in the pruned tree. *SS(word)*, the sum of the scores over all trees, *SSSS(word)*, the square root of the sums of the squares of the scores over all trees, and *N(word)*, the number of trees in which the word occurs are all computed, and the final score for the word is given by

$$score(word) = \log \frac{SS(word)^{N(word)+1}}{SSSS(word)^{N(word)}} \qquad (5)$$

3.2.4. Training and Dictionaries

The weights and saliences that are used to make the word trees are computed from a set of training documents. We used the FT, part of FBIS, part of LA and CRE documents to train. Our experience led us to be suspicious of the Federal Register (FR) documents, and, in fact, we did not even include them in the database to be queried. We also felt that the congressional record (CRE) documents were good for training, so we left them in, although we didn't include them in the database.

The "1998 TREC-7 SPOKEN DOCUMENT RETRIEVAL (SDR) TRACK: Specification 3" document, section 10, states that "any auxiliary IR training material or auxiliary data structures such as thesauri that are used must predate the May 1, 1998 recognition/retrieval date". We had planned on modifying our dictionary for the SDR task, as we did last year, to include some common recognizer errors, but could not under our interpretation of this statement. In fact, some extra words were added to our dictionary before we received the rules and we had not kept a copy of our original dictionary. Our dictionary, therefore, complies with the spirit, but not the strict sense, of this rule.

3.3. Retrieval

The goal is to score each document topic list in the database against the query topic list, then take the high scoring messages as relevant to the query. Two scores are used: the full score and the rough score. When feedback is not being used, only the full score is used, but when feedback is being used, the rough score is used for the first pass and the full score is used for the second. Our experiments indicated that using different scores was more effective than a single score. Once we have the final scores for the document segments, they are combined into a single score for the entire document.

The following sections describe the full score, the rough score, how words in classes are handled, and how the segment scores are combined into a document score. The final section describes how the query topic list is modified between the first and second pass when feedback is being used.

3.3.1. Full Scoring of Segments

Except for the *wt* term, this is the same score we used last year:

$$score = hits^3 \cdot \sum_{words} wt \cdot mwct^2 \cdot (idf)^{1.5} \cdot 0.85^{(qrank+srank)} \quad . \qquad (6)$$

The sum is over all words which occur in both the query topic list and the segment topic list.

hits is the number of words in both the segment and query topic list. Words with rank 7 or more in the query topic list are counted as quarter hits.

wt is the weight of the word: 4 for descriptive adjectives, 1 for all others.

mwct is the number of pieces in the word. For example, "united_states" has *mwct*=2.

idf is the total number of segments divided by the number of segments which contain the word.

qrank is the rank of the word in the query topic list.

srank is the rank of the word in the segment topic list.

This score rewards segments containing words which are well ranked in both the segment and the query topic lists, occur rarely in the database, are multi-word units, or are descriptive adjectives. Segments which have many well ranked words in common with the query topic list are further rewarded.

3.3.2. Rough Scoring of Segments

We experimented with a number of scores for the first pass of feedback, with the expectation that the best score to use

would give many relevant segments in the highest 10 or 15 scores. The score given here is the one which gave the best results when feedback was used.

$$score = hits^3 \cdot \sum_{words} wt \cdot 0.85^{srank} . \quad (7)$$

As before, the sum is over all words which occur in both the query topic list and the document segment topic list.

hits is the number of words in both the segment and query topic list. Words with rank 7 or more in the query topic list are counted as quarter hits.

wt is the weight of the word: 1/3 for class words, 1 for all others.

srank is the rank of the word in the segment topic list.

This score rewards documents containing words which are well ranked in the segment topic list or are not words in a class. Segments which have many well ranked words in common with the query topic list are further rewarded.

3.3.3. Class Words and Child Words

Class words and child words (those words which do not appear in the original text) are handled differently than the other words in a topic list. Our experiments indicated that using the child words was hurting performance, so we decided not to use them. They do have an implicit influence, however, since they are given a rank and so push down the ranks of subsequent words in the topic list.

For class words, only the word in the class which increases the document score the most is used. In this sense the class behaves as a single word. The term *idf* in the full score, however, uses the average number of segments in which any word in the class appears.

3.3.4. Scoring Documents

Once the segments have been scored, they are ranked, and the scores are combined into document scores:

$$score(document) = \sum_{segments} sscore \cdot e^{-0.01(srank - brank)} . \quad (8)$$

The sum is over all the segments in the document. *sscore* is the score of the segment, *srank* is the rank of the segment, and *brank* is the rank of the best scoring segment of the document.

This takes the top score among the segments in the documents and adds in the lower scores, giving less weight to the segments which score poorly. Our intent is to ensure that a document with only one well scoring segment still scores well.

3.3.5. Using Feedback to Modify the Query Topic Lists

A brief history of our feedback experiments will be useful in understanding the algorithms we used in our submitted runs. We started with an IR system whose average precision at 30 documents was 23.5% on the TREC5 queries with the CR and FT documents. After much experimentation, we achieved an average precision at 30 documents with feedback of 27.0%. At this point we tested the TREC6 queries and were disappointed to see a precision of 23.0% with no feedback drop to 21.7% with feedback. We spent the remainder of our time until the *Ad Hoc* task submission deadline trying to determine why what had helped on the TREC5 queries had hurt on the TREC6 queries. Ultimately, we proceeded on the assumption that we had over trained on the TREC5 data, and the feedback algorithm we ultimately used for the *Ad Hoc* task submission helped the TREC5 queries only moderately, but at least did not hurt the TREC6 queries. Between the *Ad Hoc* and SDR task submission deadlines, we made several refinements to strengthen the feedback algorithm.

3.3.5.1. The *Ad Hoc* ("mild") System

The first step in modifying the query topic list by feedback is to make of list of "valuable" words from the top documents from the first pass. For the *Ad Hoc* task valuable means the word is one which occurs in 9 or more of the 12 top

scoring document segments. If a word in the query topic list is valuable, its rank is improved by 1, otherwise it is made worse by 1. Valuable words which are not in the query topic list are inserted at a rank of

$$6 + \left\lfloor \frac{brank}{5} \right\rfloor ,$$

where *brank* is the best rank of the word in the top 12 segment topic lists.

3.3.5.2. The SDR ("strong") System

For the SDR task, we clustered the 5 top scoring document segments from the first pass using a similarity between segment topic lists given by

$$similarity = \sum_{words} wt \cdot mwct^2 \cdot idf^{1.5} \cdot 0.85^{(rank1 + rank2)} . \qquad (9)$$

If there are three or more documents which cluster with the top scoring document, then we use the words in the topic lists of those documents to do feedback. If the cluster has only one or two documents, we do not use feedback for the query. We use a threshold of 75% of the similarity of the most similar pair to do the clustering.

Next the words are scored to determine which ones are valuable. The score we use is

$$score = \sum_{segments} 0.95^{slot} \cdot (0.95 - 0.009^{srank})^{(0.09 \cdot srank)} . \qquad (10)$$

The sum is over the segments whose topic list contains the word, and

$$slot = \left\lfloor \frac{rank - 1}{20} \right\rfloor , \qquad (11)$$

where *srank* is the rank (starting with *0*) of the segment among the top scoring segments and *rank* is the rank of the word in the segment topic list. A word is considered valuable if it scores better that 55% of the maximum score, which is the number of segments in the cluster.

The query topic list is now divided into a list of valuable words and a list of non-valuable words, each list ordered by rank in the original query topic list. We make a first approximation to the new list by putting the valuable list first, the non-valuable list second and reordering them top to bottom. Words which are valuable but not in the query topic list are all given the same rank as the best non-valuable word in the query topic list. Next we shift all valuable words down by $\lfloor segmentrank/4 \rfloor$, the valuable words in the original query topic list down by $\lfloor queryrank/4 \rfloor$, and the non-valuable words in the original query topic list down by $\lfloor queryrank/8 \rfloor$. If no words were valuable, we make no changes at all and so do no feedback.

4. TIMINGS

We used the same base machine as last year, namely a Sun Enterprise 5000 (250 MHz, 8 heads, and 1040 MB of RAM) with approximately 50% of the CPU time available for our work. The timings for the SDR tasks are given in Table 1. A further 6 seconds were taken in each condition to create the database.

Table 1: Time for Topic Labeling on SDR Tasks

Condition	# messages	Pre-processing &Topic Labeling
Reference	2866	1.6 min
Baseline 1	2865	2.4 min
Baseline 2	2865	2.4 min

Our *Ad Hoc* system last year required over 85 hours to train, which made experimentation difficult. This year we were able to build the whole database for the *Ad Hoc* task in just slightly under 10 hours. These 10 hours do not include the initial filtering of the data since we used the filtered output from last year. We built the database several times, and the wall-clock timings for one of the early builds are shown in Table 2. The times for the Federal Register documents are starred to indicate that we did not use the Federal Register documents in the final version of our database. It took an additional 34 minutes (2043.51 seconds of CPU time) to build the database from the topic labeled documents.

Table 2: Approximate Wall-clock Time for Topic Labeling on Ad Hoc Task

DOC TYPE	# documents	Pre-processing &Topic Labeling
FBIS (in three parts)	61,578; 26,438; 42,455	44 min; 19 min; 48 min
FR (in two parts)	26,843; 28,787	22 min *; 28 min *
FT (in two parts)	76,857; 133,301	97 min; 157 min
LA (in three parts)	43,803; 54,603; 34,210	75 min; 67 min; 41 min

Timings for the query process are given in Table 3.

Table 3: Time for Queries

TASK	Time
Ad Hoc: Without feedback	558.6 s
Ad Hoc: With feedback	1468.2 s
SDR: Reference 1 transcripts	4.2 s
SDR: Baseline 1 recognizer transcripts	4.6 s
SDR: Baseline 2 recognizer transcripts	4.5 s

5. RESULTS.

The results for the runs we submitted are given in Tables 4 and 5. We submitted two runs for the *Ad Hoc* task: our primary run used no feedback and our secondary run used feedback. Our SDR runs used feedback. The average precision at 30 documents on the *Ad Hoc* task, for instance, increased from 19% last year to nearly 27% this year. As we expected, the *Ad Hoc* feedback run shows only small changes over the non-feedback run, but in most cases it gives an improvement.

6. EXPERIMENTS

We decided to use the 50 TREC5 (1996) *Ad Hoc* queries as a development set and the 50 TREC6 (1997) *Ad Hoc* queries as a test set for our experiments. There are three document sets (CR, FR and FT) in common between these years, but our experience indicated that our system did not do well with the FR documents, so we decided to use only the CR and FT documents to query against. The CR documents are not used in TREC7, but using the FT documents alone seemed dangerous. For the SDR task, we used the 5 queries given in the TREC7 training data and queried against the IBM ltt files from set 1 and set 2.

6.1. Feedback

Table 6 shows the effect of feedback on the development and test data. "Mild feedback" refers to the method used for the *Ad Hoc* task submission (section 3.3.5.1), while "strong feedback" refers to the method used for the SDR task submission (section 3.3.5.2).

Table 4: Recall Level Precision Averages

Recall	Precision				
	Ad Hoc no feedback	Ad Hoc feedback	SDR 0% WER	SDR ~35% WER	SDR ~50% WER
0.0	0.7277	0.7184	0.7866	0.7713	0.6688
0.1	0.4203	0.4150	0.6195	0.6294	0.4909
0.2	0.2657	0.2620	0.5937	0.5414	0.4662
0.3	0.1775	0.1891	0.5387	0.4839	0.3946
0.4	0.1275	0.1441	0.4544	0.4133	0.3271
0.5	0.0824	0.1005	0.4228	0.3880	0.2830
0.6	0.0528	0.0665	0.3399	0.3080	0.2249
0.7	0.0290	0.0383	0.2476	0.2202	0.1701
0.8	0.0144	0.0238	0.2199	0.2038	0.1558
0.9	0.0032	0.0070	0.1687	0.1281	0.0946
1.0	0.0002	0.0001	0.1105	0.1002	0.0748
non-interpolated	0.1449	0.1537	0.3907	0.3640	0.2868

Table 5: Document Level Precision Averages

Number of Documents	Precision				
	Ad Hoc no feedback	Ad Hoc feedback	SDR 0% WER	SDR ~35% WER	SDR ~50% WER
5	0.4400	0.4680	0.4870	0.4870	0.3913
10	0.3700	0.3860	0.3478	0.3783	0.2957
15	0.3387	0.3360	0.2957	0.3188	0.2667
20	0.3060	0.3120	0.2696	0.2717	0.2370
30	0.2693	0.2693	0.2275	0.2362	0.2000
100	0.1612	0.1672	0.1161	0.1048	0.0983
200	0.1110	0.1171	0.0622	0.0583	0.0570
500	0.0644	0.0674	0.0277	0.0269	0.0270
1000	0.0400	0.0413	0.0146	0.0143	0.0144
Exact	0.1965	0.2048	0.3703	0.3442	0.2475

Table 6:

Task	Feedback Condition	avg. precision @30 documents	avg. precision @ R documents
Ad Hoc Development Set	none	21.481%	26.752%
	mild	22.444%	28.153%
	strong	23.259%	32.152%
Ad Hoc Test Set	none	20.069%	17.612%
	mild	20.069%	17.702%
	strong	20.069%	17.702%
TREC7 SDR training	none	28.000% (@5 docs)	23.684%
	strong	32.000% (@5 docs)	28.947%

6.2. Topics versus Words

For the SDR training data, we looked at the topic lists of the relevant documents to see how the document list for our queries could be modified to be more successful. For example, the query topic list for the third query ("What have the effects of the U.N. sanctions against Iraq been on the Iraqi people, the Iraqi economy, or world oil prices?") was:

1 - iraq<PLACE>
2 - oil<NOUN>
3 - economy<NOUN>
4 - effect<NOUN>
5 - world<NOUN>
6 - iraqi<ADJ_NOUN>
7 - sanction<NOUN>
8 - price<NOUN>
9 - sanction<ACTION_VERB>
10 - people<NOUN>

The topic lists for the documents judged relevant for this query contained a number of different words to denote the topics "iraq" (iraq<PLACE>, iraqi<ADJ_NOUN>, baghdad<PLACE>), "oil" (oil<NOUN>, gas<NOUN>, petroleum<NOUN>, crude_oil<NOUN>) and "effect" (effect<NOUN>, impact<ACTION_VERB>, impact<NOUN>). When we supplemented the topic lists for all the queries (by hand) to contain additional words from the relevant documents, our average precision at the number of relevant documents went from 28% to 50%.

We concluded that a topic labeling system which returned "topics" - collections of words, any one of which could represent the topic - had the potential of performing much better than one which only returned single words. It seems difficult to get this information, since it requires finding words which have similar meanings, which requires passing from a word to its definitions, then to words which have similar definitions. We experimented with this through the class words described above, and although our list of classes was very small it did improve performance: the average precision at 30 documents for the development set with no feedback was 20.4% with no class words and 21.3% with class words. The test set had many fewer class words (probably a deficiency in our filter), so the effect was less, but in the same direction.

6.3. Children words

The Semantic Forests algorithm is designed to produce topic words which are not necessarily in the original document ("children words"). We had hoped that these words would help performance, and were disappointed to see that they did not. For instance, on the development set, without feedback, the average precision at 30 documents was 21.3% without the children words and dropped to 20.5% with the children words. We decided not to use the children words for our submitted runs, but feel that this indicates that Semantic Forests (or the dictionary we are using) can be improved further.

6.4. Natural Language Processing of Queries

We attempted to do some very primitive natural language processing on the queries to help eliminate phrases which contained little information about the topic, such as "relevant documents will identify". Phrases which can be anticipated can be handled by a filter, but that is not a very robust solution. We tried to identify adjective-noun groups on the premise that they were the most topical, but our initial experiments indicated this was hurting the performance and we abandoned this approach for lack of time. This still seems like an important avenue to pursue, however.

6.5. Separate *vs.* Combined Databases

We saw a modest gain in performance (23.9% average precision at 30 documents increased to 24.2%) when we ran against two separate databases (CR and FT in one and FR, FBIS and LA in another) compared to one single database. Separate databases affect the scoring only in the *idf* term of equation (6), since the counts of the number of times a word appears in the documents changes. We saw a drop in performance if the databases were too small, so there may be an optimum size for the databases, or the increase we saw may be pure chance. In any case, we had difficulty implementing feedback with multiple databases, and so chose to run with a single database.

6.6. Rough Scores

Not unexpectedly, the rough score (7) performed worse than the full score (8): on the development set, the average precision at 12 documents was 26.3% for the rough score and 28.0% for the full score. We were surprised, therefore, to find that using the rough score for the first pass (and the full score for the second pass) of feedback did better than using the full score for the both passes: 23.4% for the rough score and 20.6% for the full score. The difference was not as great on the test set, but the trend was the same. We were unable to find a good explanation for this, but chose to run with two separate scores for feedback.

6.7. Federal Register (FR) Documents

When the FR documents were added to the database, the average precision at 30 documents dropped from 21.3% to 18.4% on the development set and from 20.1% to 17.6% on the test set. We are not sure why our system has such trouble with the FR documents, but the effect was dramatic and we decided not to use them.

6.8. Document Segments

Division of the documents into contiguous segments containing no more than 500 words gave an increase in the average precision at 30 documents from 18.0% to 21.3% on the development set and from 15.6% to 20.1% on the test set, a sizeable gain. The 500 word limit was chosen to give relatively small segments and a manageable database.

6.9. Different Tree-Merging Methods

The decision to use two separate tree-merging methods for documents and queries was motivated by the experiments shown in Table 7. The two methods are described in Sect 3.2.3 as the "document method" and the "query method", reflecting their ultimate use. The last row shows the best performance and the best improvement from feedback.

Table 7: Effect of Tree-Merging Methods on Ave. Precision at 30 Documents.

Method used for Documents	Method used for Queries	Dev. set no feedback	Dev. set feedback	Test set no feedback	Test set feedback
Query	Document	16.8%	18.7%	11.9%	12.1%
Query	Query	18.1%	17.6%	13.3%	13.3%
Document	Document	19.9%	21.5%	19.9%	17.2%
Document	Query	20.9%	23.2%	19.9%	19.8%

7. Summary

We continue to view Semantic Forests, especially with its current improvements, as an interesting and effective approach to IR. Implicit in this approach is the use of a dictionary, not only for building trees, but also in controlling the processing of words and in constructing classes of words. It should be noted that there are no specific stops words but rather the word statistics in the dictionary control what document words become topic words. Some of our new classes were developed by inverting the dictionary.

We continue to be disappointed that the explicit use of child words as topic words is not useful. (We seem to be reaccumulating the wisdom of others who have had problems with query and document expansion.) They do, however, have great influence, through the tree building and merging process, on the selection, ranking and weighting of topic words.

Our experience with Semantic Forests has helped us identify several areas for achieving marked improvement. The first of these is parsing of the queries. Ideally, we would like to automatically produce a set of filtered text to be used to generate desired topics and a set of filtered text for topics to avoid. A first pass could be made to get all documents which match the desired topics and a second pass made to eliminate the ones which discuss unwanted aspects of the topic.

Also, we need a better way of generating lists of words which constitute topics. The few experiments we were able to perform indicate that this could improve performance significantly, but these lists seem very language dependent and we are not sure how to extract the desired information from a dictionary or thesaurus. An extreme case is one of the training queries for the SDR task, which asks for documents relating to scandals in the Clinton cabinet. To return the relevant documents, our system would have to know that "Clinton cabinet" matches "Hazel O'Leary", and this kind of information is not in an ordinary dictionary.

Finally, we would like to improve our part of speech tagger and find a way to assign probabilities to multiple definitions of a word. Our primary interest, however, is in transcribed speech, where the text is imperfect and these methods would be less effective or ineffective.

8. REFERENCES

1. Schone, P., Nelson, D., "A Dictionary-Based Method for Determining Topics in Text and Transcribed speech," 1996 IEEE International Conference on Acoustics, Speech, & Signals Processing, Atlanta, Georgia, May, 1996; Vol. 1, pp. 295-298.

2. Schone, P., Townsend, J., Crystal, T., Olano, C., "Text Retrieval via Semantic Forests," The Sixth Text Retrieval Conference (TREC-6). NIST Special Publication 500-240, 1998, pp. 761-773.

3. Voorhees, E., Harmon, D. "Overview of the Seventh Text REtrieval Conference (TREC-7)", presented at TREC-7.

4. Garafolo, J., "Overview of the Spoken Document Retrieval Track", presented at TREC-7.

TREC-7 Results

APPENDIX A

This appendix contains the evaluation results for the TREC-7 runs. The initial pages describe the evaluation measures used for the main task and many of the tracks. Each run submitted for the difference task/tracks is listed (identified by the run tag). 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. When a track uses different measures, the evaluation measures are described in the track report. The remainder of the appendix contains the evaluation results themselves, in the order given in the run list.

ADHOC RUNS

Tag	Organization	Query Method	Topic Length
att98atc	AT&T Labs Research	automatic	T
att98atdc	AT&T Labs Research	automatic	T+D
att98atde	AT&T Labs Research	automatic	T+D
acsys7al	Australian National University	automatic	T+D+N
acsys7as	Australian National University	automatic	T
LIAClass	Avignon Computer Science Laboratory	automatic	T
LIArel2	Avignon Computer Science Laboratory	automatic	T
LIAshort2	Avignon Computer Science Laboratory	automatic	T+D
bbn1	BBN Technologies	automatic	T+D+N
ok7am	City University/Univ. of Sheffield/Microsoft	automatic	T+D
ok7as	City University/Univ. of Sheffield/Microsoft	automatic	T
ok7ax	City University/Univ. of Sheffield/Microsoft	automatic	T+D+N
Cor7A1clt	Cornell University/Sabir Research, Inc.	automatic	T
Cor7A2rrd	Cornell University/Sabir Research, Inc.	automatic	D
Cor7A3rrf	Cornell University/Sabir Research, Inc.	automatic	T+D+N
pirc8Aa2	CUNY	automatic	T+D+N
pirc8Ad	CUNY	automatic	D
pirc8At	CUNY	automatic	T
fsclt7a	FS Consulting, Inc.	automatic	T
fub98a	Fondazione Ugo Bordoni	automatic	T+D+N
fub98b	Fondazione Ugo Bordoni	automatic	T+D+N
FLab7ad	Fujitsu Laboratories, Ltd.	automatic	D
FLab7at	Fujitsu Laboratories, Ltd.	automatic	T
FLab7atE	Fujitsu Laboratories, Ltd.	automatic	T
gersh2	GE Reseach and Development	automatic	T+D
gersh3	GE Reseach and Development	automatic	T+D
iit98au1	GMU/ORD/FIT/ NCR	automatic	T+D+N
iit98au2	GMU/ORD/FIT/ NCR	automatic	T
ibmg98a	IBM T.J. Watson Research Center	automatic	T
ibmg98b	IBM T.J. Watson Research Center	automatic	T+D
ibmg98c	IBM T.J. Watson Research Center	automatic	T+D
ibms98a	IBM T. J. Watson Research Center	automatic	D
ibms98b	IBM T. J. Watson Research Center	automatic	D
ibms98c	IBM T. J. Watson Research Center	automatic	D
ic98san3	Imperial College of Science, Technology and Medicine	automatic	T+D
ic98san4	Imperial College of Science, Technology and Medicine	automatic	T+D+N
MerAdRbtd	Institut de Recherche en Informatique de Toulouse	automatic	T+D
MerAdRbtnd	Institut de Recherche en Informatique de Toulouse	automatic	T+D+N
MerTetAdtnd	Institut de Recherche en Informatique de Toulouse	automatic	T+D+N
KD70000	KDD R&D Laboratories	automatic	T+D+N
KD71010q	KDD R&D Laboratories	automatic	T+D+N

ADHOC RUNS (Continued)

Tag	Organization	Query Method	Topic Length
KD71010s	KDD R&D Laboratories	automatic	T+D+N
dsir07a01	Kasetsart University	automatic	T+D
dsir07a02	Kasetsart University	automatic	T+D
kslsV1	Keio University	automatic	T+D
LNaTit7	Lexis-Nexis	automatic	T
LNaTitDesc7	Lexis-Nexis	automatic	T+D
jalbse011	MIT	automatic	T+D
jalbse012	MIT	automatic	T+D
jalbse013	MIT	automatic	T+D
nectitech	NEC Corp. and Tokyo Institute of Technology	automatic	T
nectitechall	NEC Corp. and Tokyo Institute of Technology	automatic	T+D+N
nectitechdes	NEC Corp. and Tokyo Institute of Technology	automatic	D
nttdata7Al0	NTT DATA Corporation	automatic	T+D+N
nttdata7Al2	NTT DATA Corporation	automatic	T+D+N
nttdata7At1	NTT DATA Corporation	automatic	T
nsasgrp3	National Security Agency	automatic	T+D+N
nsasgrp4	National Security Agency	automatic	T+D+N
nthu2	National Tsing Hua University	automatic	T+D+N
nthu3	National Tsing Hua University	automatic	T+D+N
mds98td	RMIT/UoM/CSIRO	automatic	T+D
mds98t	RMIT/UoM/CSIRO	automatic	T
mds98t2	RMIT/UoM/CSIRO	automatic	T
AntHoc01	Rutgers University	automatic	T+D+N
ScaiTrec7	Seoul National University	automatic	T+D+N
ETHAB0	Swiss Federal Institute of Technology (ETH)	automatic	T
ETHAC0	Swiss Federal Institute of Technology (ETH)	automatic	T
ETHAR0	Swiss Federal Institute of Technology (ETH)	automatic	T
APL985L	The Johns Hopkins University	automatic	T+D+N
APL985LC	The Johns Hopkins University	automatic	T+D+N
APL985SC	The Johns Hopkins University	automatic	T
tno7cbm25	TwentyOne	automatic	T+D+N
tno7exp1	TwentyOne	automatic	T+D+N
tno7tw4	TwentyOne	automatic	T+D+N
unc7aal1	UNC Chapel Hill	automatic	T+D+N
unc7aal2	UNC Chapel Hill	automatic	T+D+N
Brkly24	University of California	automatic	T
Brkly25	University of California	automatic	T+D+N
iowacuhk1	University of Iowa	automatic	T+D+N
iowacuhk2	University of Iowa	automatic	T+D+N
umd98a1	University of Maryland	automatic	T
umd98a2	University of Maryland	automatic	T+D+N
INQ501	University of Massachusetts, Amherst	automatic	T+D+N

ADHOC RUNS (Continued)

Tag	Organization	Query Method	Topic Length
INQ502	University of Massachusetts, Amherst	automatic	T+D+N
INQ503	University of Massachusetts, Amherst	automatic	T+D
uwmt7a0	University of Waterloo	automatic	T
acsys7mi	Australian National University	manual	
CLARIT98CLUS	CLARITECH Corporation	manual	
CLARIT98COMB	CLARITECH Corporation	manual	
CLARIT98RANK	CLARITECH Corporation	manual	
fsclt7m	FS Consulting, Inc.	manual	
gershl	GE Reseach and Development	manual	
iit98mal	GMU/ORD/FIT/ NCR	manual	
harrisl	Harris Information Systems Division	manual	
LNmanual7	Lexis-Nexis	manual	
lanl981	Los Alamos National Laboratory	manual	
t7mitil	Management Information Technologies, Inc.	manual	
nthul	National Tsing Hua University	manual	
Brkly26	University of California	manual	
uoftimgr	University of Toronto	manual	
uoftimgu	University of Toronto	manual	
uwmt7al	University of Waterloo	manual	
uwmt7a2	University of Waterloo	manual	

CROSS-LANGUAGE TRACK

Document Language: EFGI

Tag	Organization	Topic Language	Run Type
98EITdes	Eurospider	German	automatic, title + desc
98EITful	Eurospider	German	automatic, title + desc + narr
98EITtit	Eurospider	German	automatic, title
ibmcl7al	IBM T. J. Watson Research Center	English	automatic, title + desc + narr
ibmcl7as	IBM T. J. Watson Research Center	English	automatic, desc
ibmcl7cl	IBM T. J. Watson Research Center	English	automatic, title + desc + narr
ibmcl7cs	IBM T. J. Watson Research Center	English	automatic, desc
tno7ddp	TwentyOne	German	automatic, title + desc + narr
tno7edp	TwentyOne	English	automatic, title + desc + narr
tno7edpx	TwentyOne	English	automatic, title + desc + narr
tno7egr	TwentyOne	English	automatic, title + desc + narr
BKYCL7AF	University of California	French	automatic, title + desc + narr
BKYCL7AG	University of California	German	automatic, title + desc + narr
BKYCL7AI	University of California	Italian	automatic, title + desc + narr
BKYCL7ME	University of California	English	manual, title + desc + narr
umdxeof	University of Maryland	English	automatic, title + desc + narr
umdxeot	University of Maryland	English	automatic, title

CROSS-LANGUAGE TRACK

Document Language: EF

Tag	Organization	Topic Language	Run Type
ceat7e1n	CEA (Commissariat ' l'Energie Atomique)	English	automatic, desc
ceat7e2n	CEA (Commissariat ' l'Energie Atomique)	English	automatic, desc
ibmcl7ef	IBM T. J. Watson Research Center	English	automatic, title + desc + narr
lanl982	Los Alamos National Laboratory	English	manual, title + desc + narr
TW1E2EF	TextWise, Inc.	English	automatic, title
tno7eef	TwentyOne	English	automatic, title + desc + narr
RaliDicE2EF	Universite de Montreal	English	automatic, title + desc + narr
RaliDicSDAef	Universite de Montreal	English	automatic, title + desc
BKYCL7AEF	University of California	English	automatic, title + desc + narr
BKYCL7MEF	University of California	English	manual, title + desc + narr

FILTERING TRACK

Tag	Organization	Run Type
att98fb5	AT&T Labs Research	Batch, F1
att98fb6	AT&T Labs Research	Batch, F3
att98fr4	AT&T Labs Research	Routing
att98fr5	AT&T Labs Research	Routing
CLARITafF1a	CLARITECH Corporation	Adaptive, F1
CLARITafF1b	CLARITECH Corporation	Adaptive, F1
CLARITafF3a	CLARITECH Corporation	Adaptive, F3
CLARITafF3b	CLARITECH Corporation	Adaptive, F3
CLARITbfF1	CLARITECH Corporation	Batch, F1
CLARITbfF3	CLARITECH Corporation	Batch, F3
sigmaTrec7F1	Canadian Imperial Bank of Commerce	Adaptive, F1
sigmaTrec7F3	Canadian Imperial Bank of Commerce	Adaptive, F3
ok7ff12	City University/Univ. of Sheffield/Microsoft	Adaptive, F1
ok7ff13	City University/Univ. of Sheffield/Microsoft	Adaptive, F1
ok7ff32	City University/Univ. of Sheffield/Microsoft	Adaptive, F3
ok7ff33	City University/Univ. of Sheffield/Microsoft	Adaptive, F3
pirc8FA1	CUNY	Adaptive, F1
pirc8FA3	CUNY	Adaptive, F3
pirc8FB1	CUNY	Batch, F1
pirc8FB3	CUNY	Batch, F3
pirc8R1	CUNY	Routing
pirc8R2	CUNY	Routing
arc98cs	IBM Research Division - Almaden Research Center	Routing
Mer7BF1	Institut de Recherche en Informatique de Toulouse	Batch, F1
Mer7BF3	Institut de Recherche en Informatique de Toulouse	Batch, F3
MerRou	Institut de Recherche en Informatique de Toulouse	Routing
nttd7bf1	NTT DATA Corporation	Batch, F1
nttd7bf3	NTT DATA Corporation	Batch, F3
nttd7rt1	NTT DATA Corporation	Routing
nttd7rt2	NTT DATA Corporation	Routing
AntRout1	Rutgers University	Routing
AntRout2	Rutgers University	Routing
TNOAF102	TNO-TPD TU-Delft	Adaptive, F1, F3
TNOAF103	TNO-TPD TU-Delft	Adaptive, F1
IAHKaf11	University of Iowa	Adaptive, F1
IAHKaf12	University of Iowa	Adaptive, F1
IAHKaf31	University of Iowa	Adaptive, F3
IAHKaf32	University of Iowa	Adaptive, F3
IAHKbf11	University of Iowa	Batch, F1
IAHKbf32	University of Iowa	Batch, F3

FILTERING TRACK (Contunued)

Tag	Organization	Run Type
INQ510	University of Massachusetts, Amherst	Adaptive, F1
INQ511	University of Massachusetts, Amherst	Adaptive, F3
Mer7AGbF1	Institut de Recherche en Informatique de Toulouse	Adaptive, F1
Mer7ARbF1	Institut de Recherche en Informatique de Toulouse	Adaptive, F1
Mer7AGbF3	Institut de Recherche en Informatique de Toulouse	Adaptive, F3
Mer7ARbF3	Institut de Recherche en Informatique de Toulouse	Adaptive, F3

HIGH PRECISION TRACK

Tag	Organization
acsys7hp	Australian National University
Cor7HP1	Cornell University/Sabir Research, Inc.
Cor7HP2	Cornell University/Sabir Research, Inc.
Cor7HP3	Cornell University/Sabir Research, Inc.
pirc8Ha	CUNY
uwmt7h1	University of Waterloo
uwmt7h2	University of Waterloo

INTERACTIVE TRACK

Tag	Organization
ok_noRF-zp_noRF	City University/Univ. of Sheffield/Microsoft
ok_noRF-ok_withRF	City University/Univ. of Sheffield/Microsoft
J24-ZP	New Mexico State University at Las Cruces
MB-MR	Oregon Health Sciences University
clus-list	RMIT/UoM/CSIRO
RUINQ_G-RUINQ_R	Rutgers University (Belkin)
C-Z	University of California, Berkeley
irisa-iriss	UNC Chapel Hill (Newby)
irisp-iriss	UNC Chapel Hill (Newby)
a-b	University of Toronto

QUERY TRACK

Tag	Organization
CorAPL1a	Cornell University/Sabir Research, Inc.
CorAPL2a	Cornell University/Sabir Research, Inc.
CorAPL5a	Cornell University/Sabir Research, Inc.
CorAPL5b	Cornell University/Sabir Research, Inc.
CorCor1	Cornell University/Sabir Research, Inc.
CorCor2	Cornell University/Sabir Research, Inc.
CorCor3	Cornell University/Sabir Research, Inc.
CorCor4	Cornell University/Sabir Research, Inc.
CorCor5	Cornell University/Sabir Research, Inc.
APLAPL1a	The Johns Hopkins University
APLAPL2a	The Johns Hopkins University
APLAPL5a	The Johns Hopkins University
APLAPL5b	The Johns Hopkins University
APLCor1	The Johns Hopkins University
APLCor2	The Johns Hopkins University
APLCor3	The Johns Hopkins University
APLCor5	The Johns Hopkins University

SPOKEN DOCUMENT RETRIEVAL TRACK

Tag	Organization	Condition
att-r1	AT&T Labs Research	reference
att-b1	AT&T Labs Research	baseline1
att-b2	AT&T Labs Research	baseline2
att-s1	AT&T Labs Research	recognizer
att-s2	AT&T Labs Research	recognizer
cmu1-r.ret	Carnegie Mellon University	reference
cmu1-b1.ret	Carnegie Mellon University	baseline1
cmu1-b2.ret	Carnegie Mellon University	baseline2
cmu1-s1.ret	Carnegie Mellon University	recognizer
cmu1-s2.ret	Carnegie Mellon University	recognizer
cmu2-r1.ret	Carnegie Mellon University	reference
cmu2-b1.ret	Carnegie Mellon University	baseline1
cmu2-b2.ret	Carnegie Mellon University	baseline2
derasru1-r1.ret	Defense Evaluation and Research Agency	reference
derasru1-b1.ret	Defense Evaluation and Research Agency	baseline1
derasru1-b2.ret	Defense Evaluation and Research Agency	baseline2
derasru1-s1.ret	Defense Evaluation and Research Agency	recognizer
derasru1-s21.ret	Defense Evaluation and Research Agency	recognizer
nsa-r1.ret	National Security Agency	reference
nsa-b1.ret	National Security Agency	baseline1
nsa-b2.ret	National Security Agency	baseline2
mds-r1.ret	RMIT/UoM/CSIRO	reference
mds-b1.ret	RMIT/UoM/CSIRO	baseline1
mds-b2.ret	RMIT/UoM/CSIRO	baseline2
mds-s1.ret	RMIT/UoM/CSIRO	recognizer
mds-s2.ret	RMIT/UoM/CSIRO	recognizer
tno-r1.ret	TNO-TPD TU-Delft	reference
tno-b1.ret	TNO-TPD TU-Delft	baseline1
tno-b2.ret	TNO-TPD TU-Delft	baseline2
tno-s1-abb.ret	TNO-TPD TU-Delft	recognizer
cuhtk-r1.ret	University of Cambridge	reference
cuhtk-b1.ret	University of Cambridge	baseline1
cuhtk-b2.ret	University of Cambridge	baseline2
cuhtk-cr-derasru1.ret	University of Cambridge	recognizer
cuhtk-cr-shef1.ret	University of Cambridge	recognizer
cuhtk-s1.ret	University of Cambridge	recognizer
umd-r1.ret	University of Maryland	reference
umd-b1.ret	University of Maryland	baseline1
umd-b2.ret	University of Maryland	baseline2

SPOKEN DOCUMENT RETRIEVAL TRACK

Tag	Organization	Condition
umass-r1.ret	University of Massachusetts, Amherst	reference
umass-b1.ret	University of Massachusetts, Amherst	baseline1
umass-b2.ret	University of Massachusetts, Amherst	baseline2
umass-s1-dragon.ret	University of Massachusetts, Amherst	recognizer
umass-s2-dragon.ret	University of Massachusetts, Amherst	recognizer
shef-r1.ret	University of Sheffield	reference
shef-b1.ret	University of Sheffield	baseline1
shef-b2.ret	University of Sheffield	baseline2
shef-cr-htk.ret	University of Sheffield	recognizer
shef-cr-sru.ret	University of Sheffield	recognizer
shef-s1.ret	University of Sheffield	recognizer

Evaluation Techniques and Measures

Categories

The results following this section are organized according to the task accomplished by the run: ad hoc or a track task. Some tracks do not use these evaluation tools. However, the evaluation tools used are described in the results section for the track.

Ad hoc
Retrieval using an "ad hoc" topic such as a researcher might use in a library environment. In TREC this implies that the input topic has no training material such as relevance judgments to aid in the construction of the input query. Systems ran TREC topics against all documents from TREC Disks 4 and 5.

Evaluation Measures

I. Recall
A measure of the ability of a system to present all relevant items.

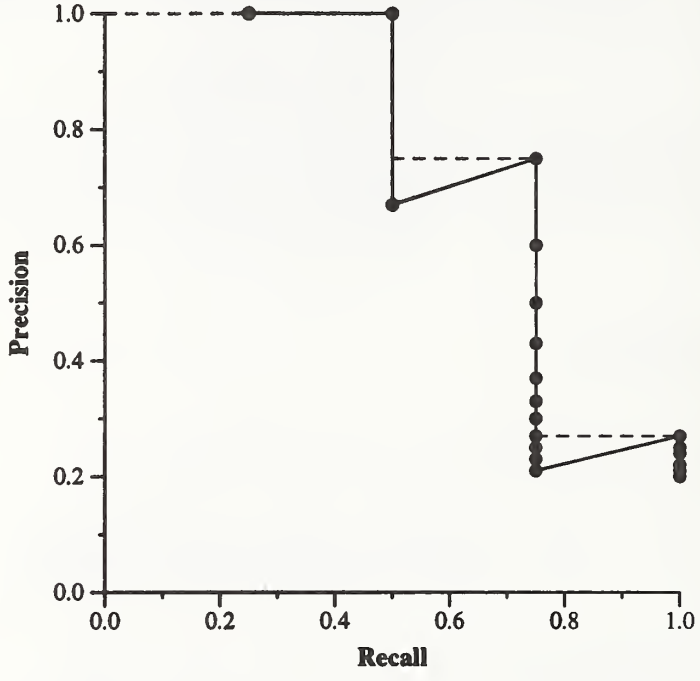
$$\text{recall} = \frac{\text{number of relevant items retrieved}}{\text{number of relevant items in collection}}$$

II. 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.



System Results Description

Each of the following pages contains the evaluation results for one run. A page is comprised of a header (containing the task and organization name), 3 tables, and 2 graphs.

Tables

Tables are generated by *trec_eval* courtesy of Chris Buckley using the SMART methodology.

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 as applicable, whether queries were constructed manually or automatically, and whether the title, description, or narrative of the topic was used.

B. Number of Topics
Number of topics searched in this run (generally 50 topics are run for each task).

C. Total number of documents over all topics (the number of topics given in B).

i. Retrieved
Number of documents submitted to NIST. This is usually 50,000 (50 topics × 1000 documents), but is less when fewer than 1000 documents are retrieved per topic.

ii. Relevant
Total possible relevant documents within a given task and category.

iii. Rel_ret
Total number of relevant documents returned by a run over all the topics.

II. "Recall Level Precision Averages" Table.
Table 2 is a sample "Recall Level Precision Averages" Table.

A. Precision at 11 standard recall levels
The precision averages at 11 standard recall levels are used to compare the performance of different systems and as the input for plotting the recall-precision graph (see below). Each recall-precision average is computed by summing the interpolated precisions at the specified recall cutoff value (denoted by $\sum P_\lambda$ where P_λ is the interpolated precision at recall level λ) and then dividing by the number of topics.

$$\frac{\sum_{i=1}^{NUM} P_\lambda}{NUM} \qquad \lambda = \{0.0, 0.1, 0.2, 0.3, \ldots, 1.0\}$$

- Interpolating recall-precision
Standard recall levels facilitate averaging and plotting retrieval results.

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

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 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,

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

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.

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.

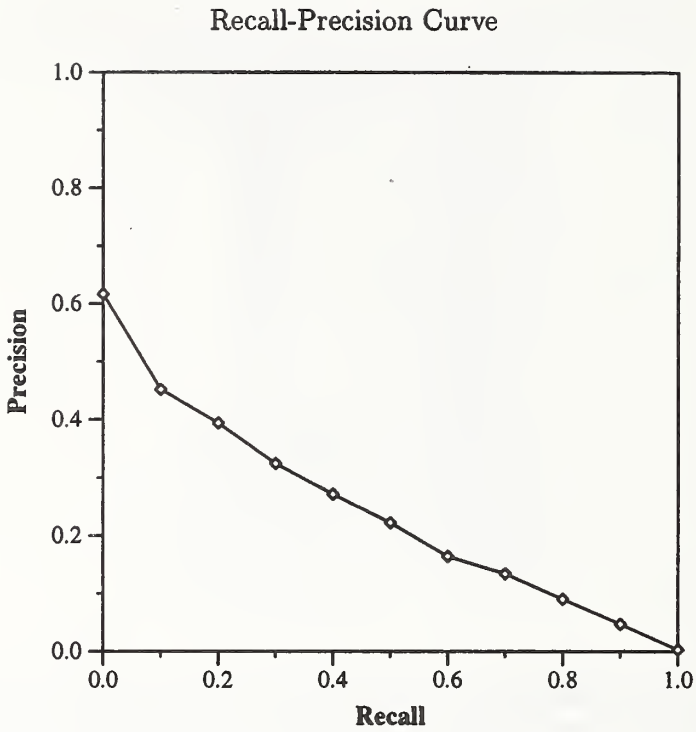


Figure 1: Sample Recall-Precision Graph.

II. Average Precision Histogram.
Figure 2 is a 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.

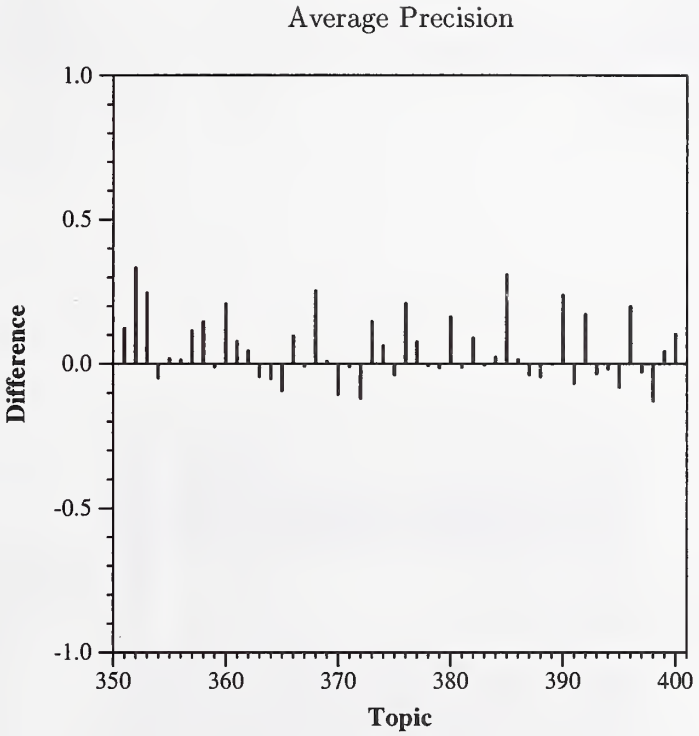
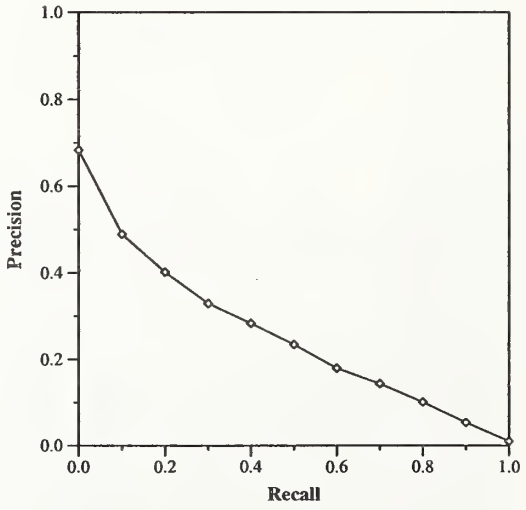


Figure 2: Sample Average Precision Histogram.

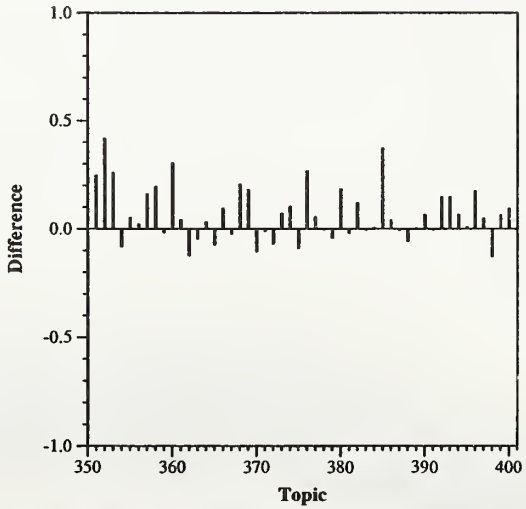
Summary Statistics	
Run Number	att98atc
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2663

Recall Level Precision Averages	
Recall	Precision
0.00	0.6835
0.10	0.4895
0.20	0.4019
0.30	0.3297
0.40	0.2838
0.50	0.2347
0.60	0.1797
0.70	0.1439
0.80	0.1011
0.90	0.0538
1.00	0.0102
Average precision over all relevant docs	
non-interpolated	0.2488

Document Level Averages	
	Precision
At 5 docs	0.5040
At 10 docs	0.4420
At 15 docs	0.4080
At 20 docs	0.3630
At 30 docs	0.3193
At 100 docs	0.2210
At 200 docs	0.1601
At 500 docs	0.0897
At 1000 docs	0.0533
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2630



Recall-Precision Curve



Difference from Median in Average Precision per Topic

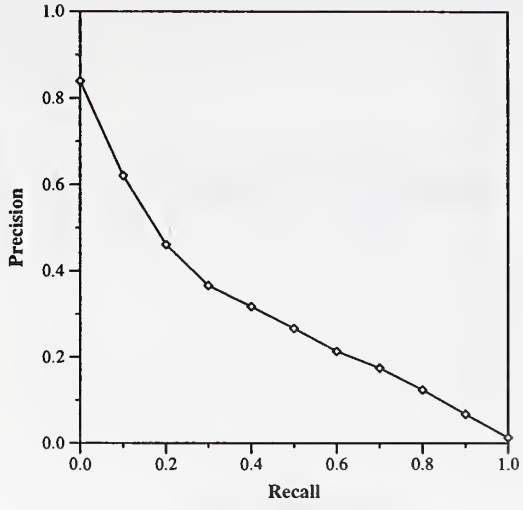
Summary Statistics	
Run Number	att98atdc
Run Description	Automatic, title + desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3260

Recall Level Precision Averages	
Recall	Precision
0.00	0.8398
0.10	0.6212
0.20	0.4615
0.30	0.3666
0.40	0.3178
0.50	0.2671
0.60	0.2141
0.70	0.1751
0.80	0.1245
0.90	0.0677
1.00	0.0130

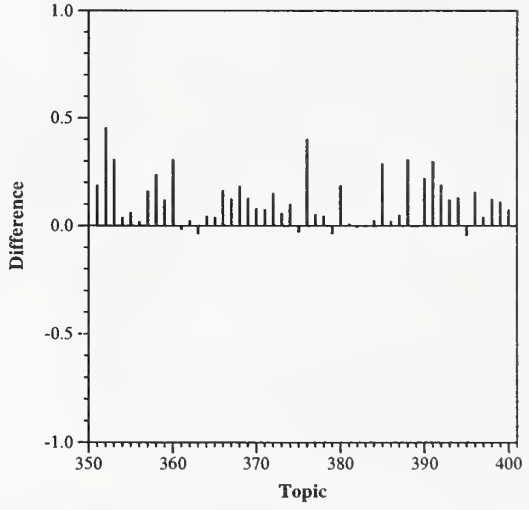
Average precision over all relevant docs	
non-interpolated	0.2961

Document Level Averages	
	Precision
At 5 docs	0.6440
At 10 docs	0.5580
At 15 docs	0.4987
At 20 docs	0.4650
At 30 docs	0.4187
At 100 docs	0.2756
At 200 docs	0.2005
At 500 docs	0.1110
At 1000 docs	0.0652

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3171



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	att98atde
Run Description	Automatic, title + desc
Number of Topics	50

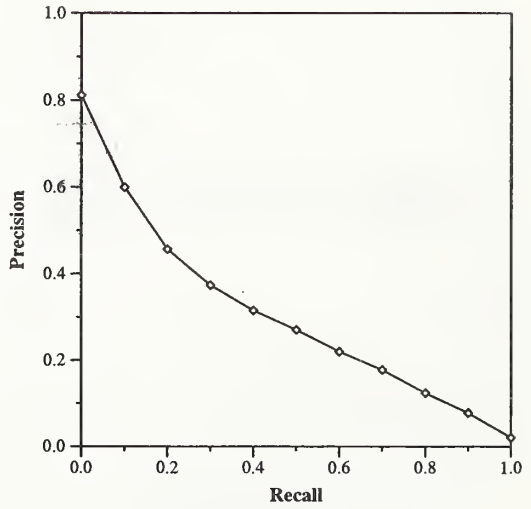
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3232

Recall Level Precision Averages	
Recall	Precision
0.00	0.8119
0.10	0.5997
0.20	0.4570
0.30	0.3738
0.40	0.3152
0.50	0.2705
0.60	0.2201
0.70	0.1779
0.80	0.1239
0.90	0.0779
1.00	0.0208

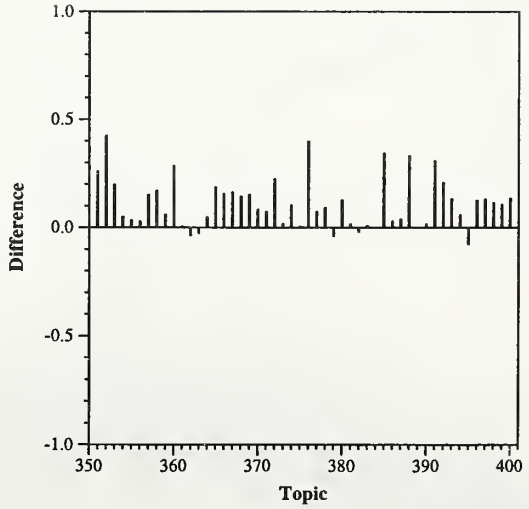
Average precision over all relevant docs	
non-interpolated	0.2940

Document Level Averages	
	Precision
At 5 docs	0.6440
At 10 docs	0.5580
At 15 docs	0.4973
At 20 docs	0.4660
At 30 docs	0.4127
At 100 docs	0.2746
At 200 docs	0.1967
At 500 docs	0.1082
At 1000 docs	0.0646

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3119



Recall-Precision Curve

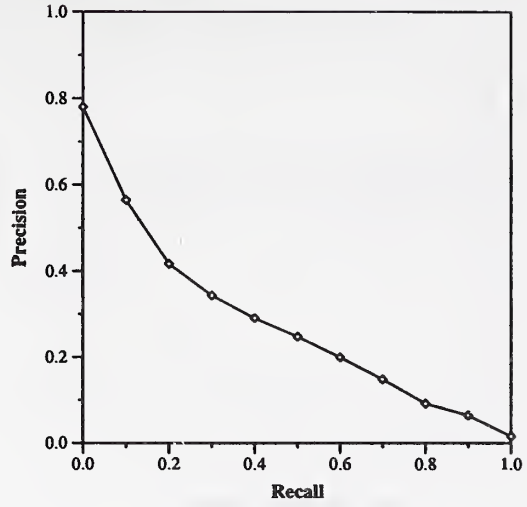


Difference from Median in Average Precision per Topic

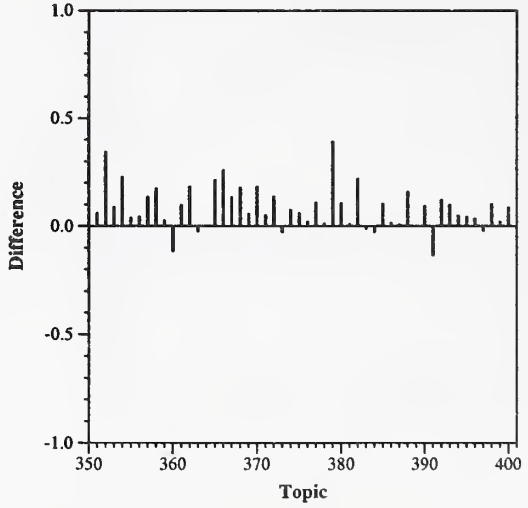
Summary Statistics	
Run Number	acsys7al
Run Description	Automatic, title + desc + narr
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3135

Recall Level Precision Averages	
Recall	Precision
0.00	0.7804
0.10	0.5647
0.20	0.4170
0.30	0.3435
0.40	0.2907
0.50	0.2478
0.60	0.1998
0.70	0.1487
0.80	0.0921
0.90	0.0648
1.00	0.0158
Average precision over all relevant docs	
non-interpolated	0.2659

Document Level Averages	
	Precision
At 5 docs	0.5400
At 10 docs	0.5040
At 15 docs	0.4747
At 20 docs	0.4390
At 30 docs	0.3780
At 100 docs	0.2368
At 200 docs	0.1723
At 500 docs	0.1018
At 1000 docs	0.0627
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2966



Recall-Precision Curve

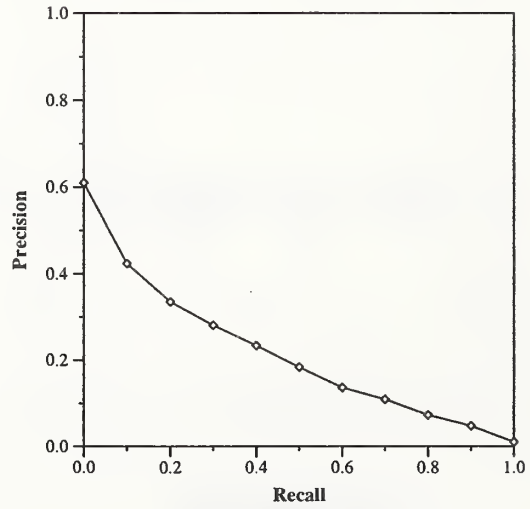


Difference from Median in Average Precision per Topic

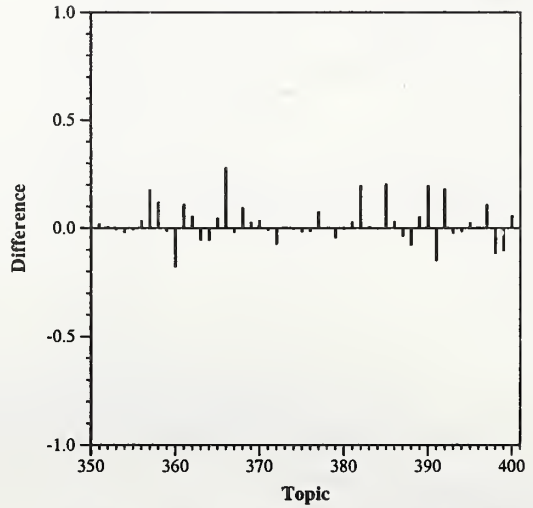
Summary Statistics	
Run Number	acsys7as
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2523

Recall Level Precision Averages	
Recall	Precision
0.00	0.6100
0.10	0.4234
0.20	0.3355
0.30	0.2811
0.40	0.2341
0.50	0.1840
0.60	0.1366
0.70	0.1094
0.80	0.0729
0.90	0.0477
1.00	0.0101
Average precision over all relevant docs	
non-interpolated	0.2045

Document Level Averages	
	Precision
At 5 docs	0.4040
At 10 docs	0.4060
At 15 docs	0.3627
At 20 docs	0.3370
At 30 docs	0.2867
At 100 docs	0.1876
At 200 docs	0.1368
At 500 docs	0.0818
At 1000 docs	0.0505
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2371



Recall-Precision Curve

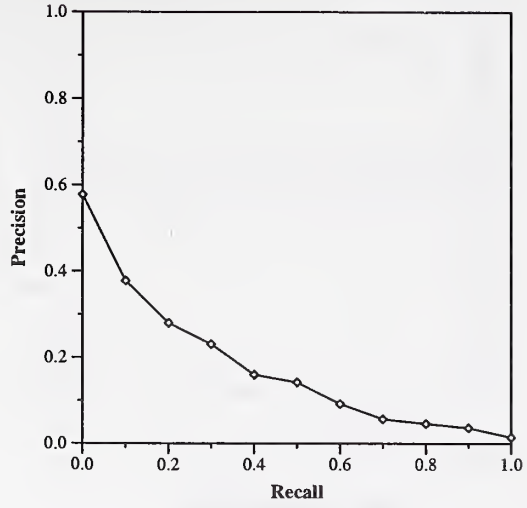


Difference from Median in Average Precision per Topic

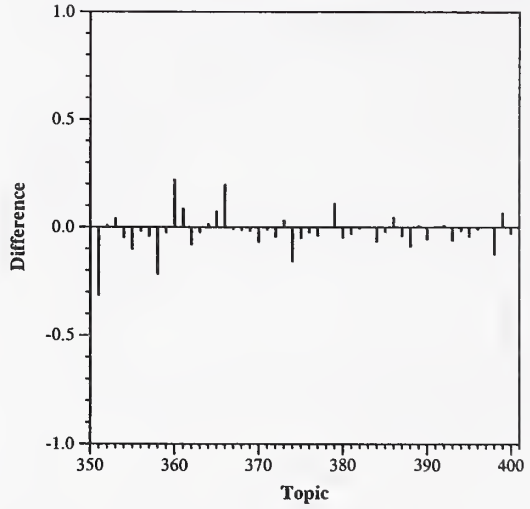
Summary Statistics	
Run Number	LIAClass
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	47488
Relevant:	4674
Rel-ret:	1992

Recall Level Precision Averages	
Recall	Precision
0.00	0.5775
0.10	0.3781
0.20	0.2802
0.30	0.2309
0.40	0.1602
0.50	0.1425
0.60	0.0925
0.70	0.0571
0.80	0.0463
0.90	0.0365
1.00	0.0147
Average precision over all relevant docs	
non-interpolated	0.1617

Document Level Averages	
	Precision
At 5 docs	0.3840
At 10 docs	0.3480
At 15 docs	0.3267
At 20 docs	0.3100
At 30 docs	0.2707
At 100 docs	0.1634
At 200 docs	0.1172
At 500 docs	0.0623
At 1000 docs	0.0398
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2191



Recall-Precision Curve

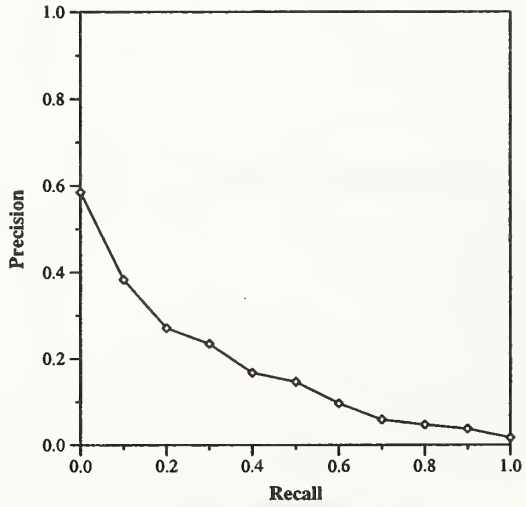


Difference from Median in Average Precision per Topic

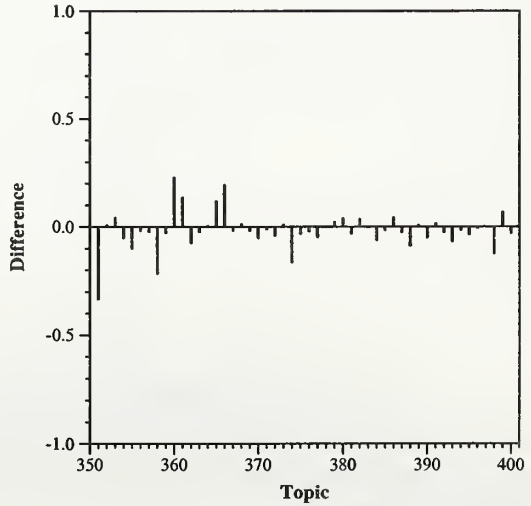
Summary Statistics	
Run Number	LIArel2
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49063
Relevant:	4674
Rel-ret:	2039

Recall Level Precision Averages	
Recall	Precision
0.00	0.5854
0.10	0.3835
0.20	0.2715
0.30	0.2352
0.40	0.1679
0.50	0.1469
0.60	0.0968
0.70	0.0589
0.80	0.0473
0.90	0.0374
1.00	0.0170
Average precision over all relevant docs	
non-interpolated	0.1648

Document Level Averages	
	Precision
At 5 docs	0.3640
At 10 docs	0.3300
At 15 docs	0.3253
At 20 docs	0.3090
At 30 docs	0.2753
At 100 docs	0.1634
At 200 docs	0.1180
At 500 docs	0.0630
At 1000 docs	0.0408
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2143



Recall-Precision Curve

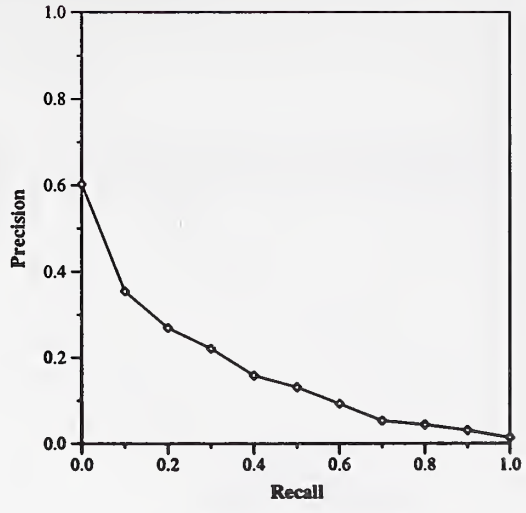


Difference from Median in Average Precision per Topic

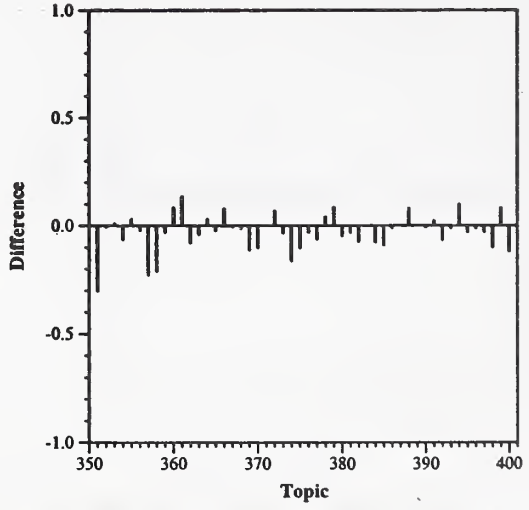
Summary Statistics	
Run Number	LIAshort2
Run Description	Automatic, title + desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	1882

Recall Level Precision Averages	
Recall	Precision
0.00	0.6029
0.10	0.3544
0.20	0.2693
0.30	0.2215
0.40	0.1587
0.50	0.1314
0.60	0.0928
0.70	0.0531
0.80	0.0440
0.90	0.0309
1.00	0.0137
Average precision over all relevant docs	
non-interpolated	0.1523

Document Level Averages	
	Precision
At 5 docs	0.3800
At 10 docs	0.3220
At 15 docs	0.2907
At 20 docs	0.2770
At 30 docs	0.2433
At 100 docs	0.1610
At 200 docs	0.1133
At 500 docs	0.0604
At 1000 docs	0.0376
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2136



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	bbn1
Run Description	Automatic, title + desc + narr
Number of Topics	50

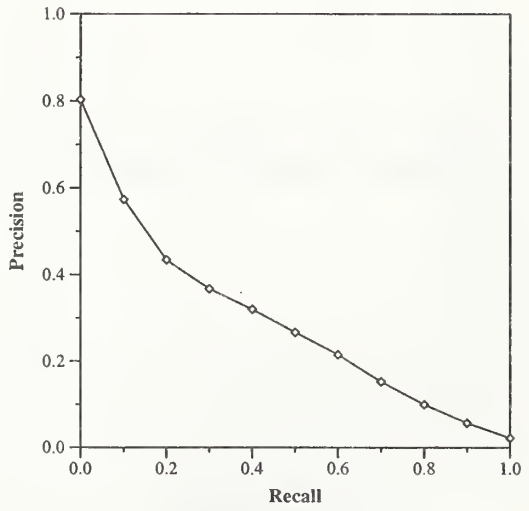
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3238

Recall Level Precision Averages	
Recall	Precision
0.00	0.8035
0.10	0.5741
0.20	0.4344
0.30	0.3679
0.40	0.3203
0.50	0.2672
0.60	0.2154
0.70	0.1528
0.80	0.0999
0.90	0.0571
1.00	0.0223

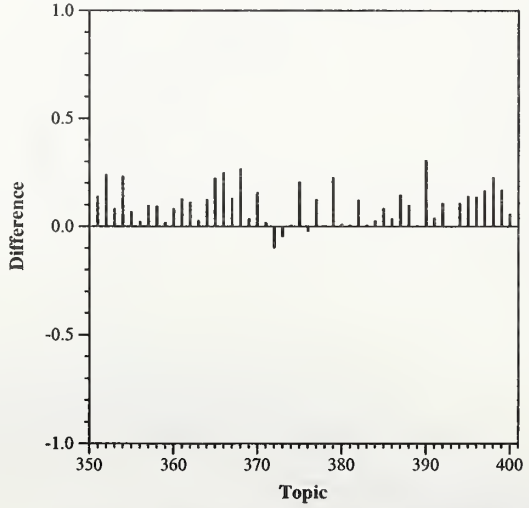
Average precision over all relevant docs	
non-interpolated	0.2797

Document Level Averages	
	Precision
At 5 docs	0.5400
At 10 docs	0.4980
At 15 docs	0.4680
At 20 docs	0.4410
At 30 docs	0.4000
At 100 docs	0.2610
At 200 docs	0.1879
At 500 docs	0.1058
At 1000 docs	0.0648

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3097



Recall-Precision Curve

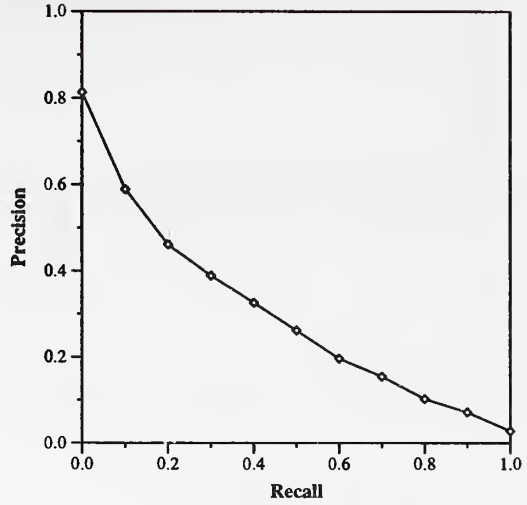


Difference from Median in Average Precision per Topic

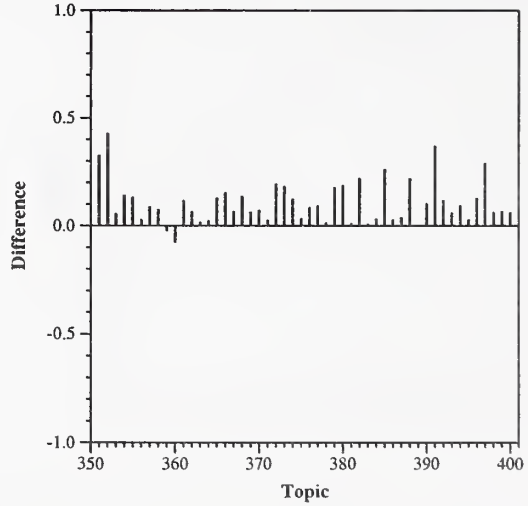
Summary Statistics	
Run Number	ok7am
Run Description	Automatic, title + desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3205

Recall Level Precision Averages	
Recall	Precision
0.00	0.8136
0.10	0.5892
0.20	0.4612
0.30	0.3890
0.40	0.3262
0.50	0.2617
0.60	0.1966
0.70	0.1548
0.80	0.1026
0.90	0.0716
1.00	0.0278
Average precision over all relevant docs	
non-interpolated	0.2876

Document Level Averages	
	Precision
At 5 docs	0.5720
At 10 docs	0.5420
At 15 docs	0.5067
At 20 docs	0.4680
At 30 docs	0.4120
At 100 docs	0.2538
At 200 docs	0.1838
At 500 docs	0.1052
At 1000 docs	0.0641
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3173



Recall-Precision Curve

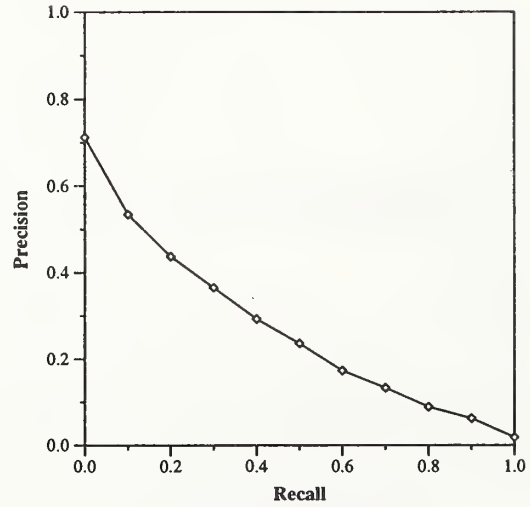


Difference from Median in Average Precision per Topic

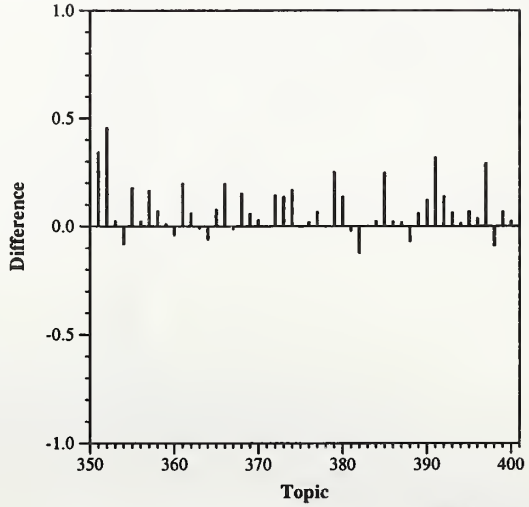
Summary Statistics	
Run Number	ok7as
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2851

Recall Level Precision Averages	
Recall	Precision
0.00	0.7123
0.10	0.5346
0.20	0.4378
0.30	0.3656
0.40	0.2934
0.50	0.2365
0.60	0.1735
0.70	0.1336
0.80	0.0894
0.90	0.0625
1.00	0.0182
Average precision over all relevant docs	
non-interpolated	0.2614

Document Level Averages	
	Precision
At 5 docs	0.5320
At 10 docs	0.4860
At 15 docs	0.4627
At 20 docs	0.4250
At 30 docs	0.3873
At 100 docs	0.2372
At 200 docs	0.1671
At 500 docs	0.0955
At 1000 docs	0.0570
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2989



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	ok7ax
Run Description	Automatic, title + desc + narr
Number of Topics	50

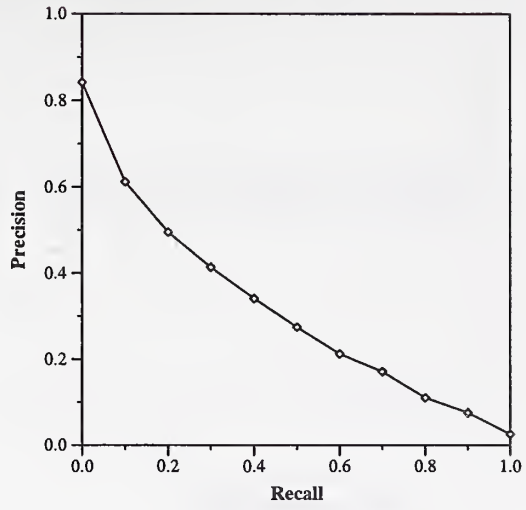
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3352

Recall Level Precision Averages	
Recall	Precision
0.00	0.8420
0.10	0.6116
0.20	0.4953
0.30	0.4142
0.40	0.3415
0.50	0.2744
0.60	0.2126
0.70	0.1713
0.80	0.1104
0.90	0.0757
1.00	0.0255

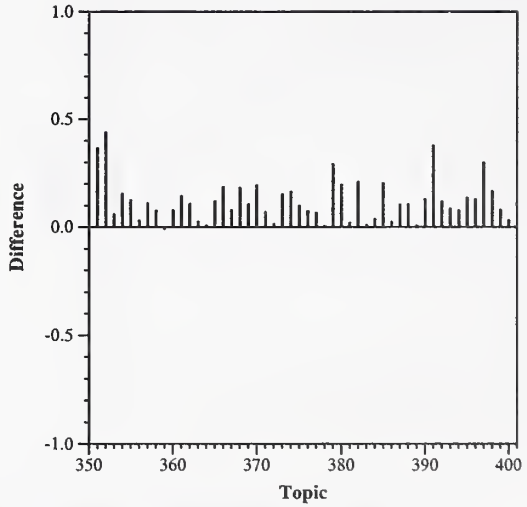
Average precision over all relevant docs	
non-interpolated	0.3033

Document Level Averages	
	Precision
At 5 docs	0.6280
At 10 docs	0.5720
At 15 docs	0.5347
At 20 docs	0.4860
At 30 docs	0.4253
At 100 docs	0.2698
At 200 docs	0.1920
At 500 docs	0.1109
At 1000 docs	0.0670

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3390



Recall-Precision Curve

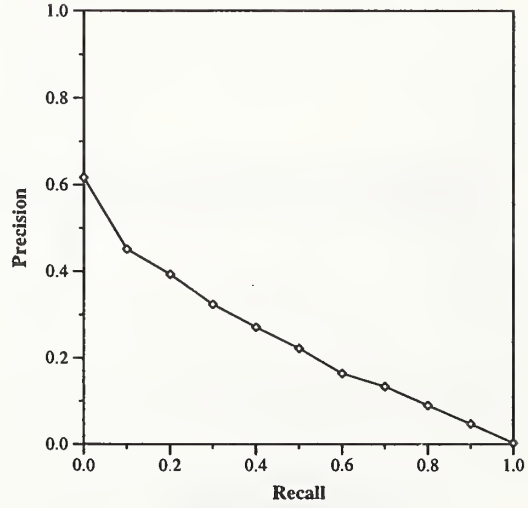


Difference from Median in Average Precision per Topic

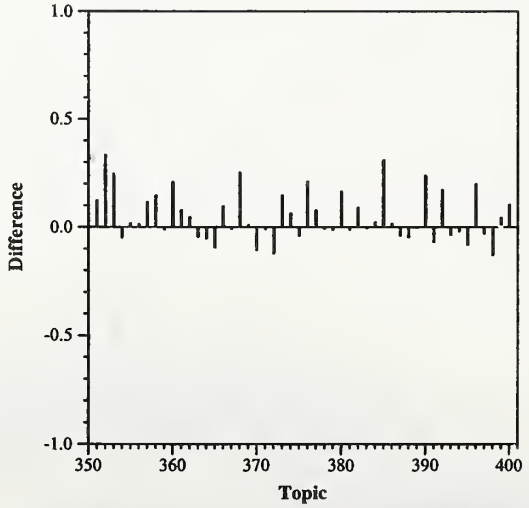
Summary Statistics	
Run Number	Cor7A1clt
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2621

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

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



Recall-Precision Curve

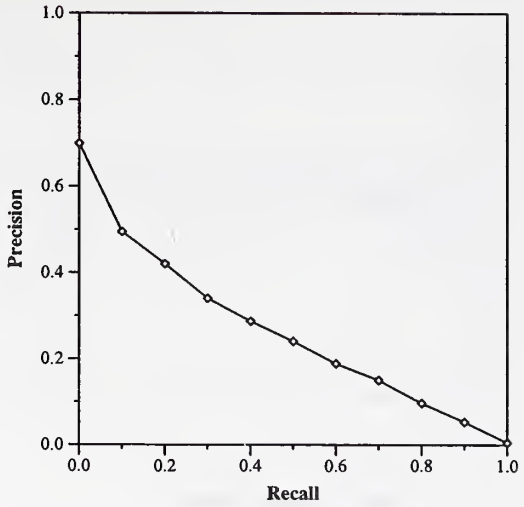


Difference from Median in Average Precision per Topic

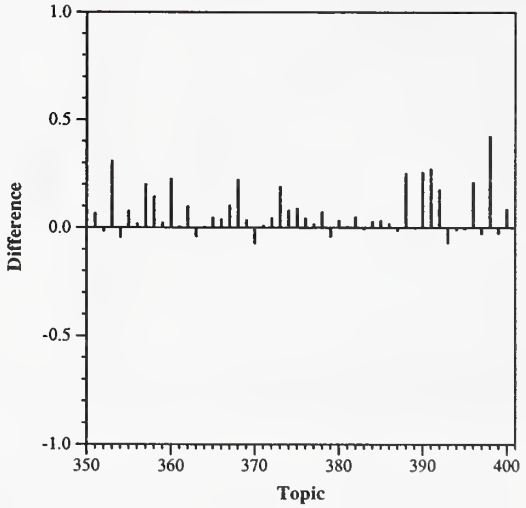
Summary Statistics	
Run Number	Cor7A2rrd
Run Description	Automatic, desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2894

Recall Level Precision Averages	
Recall	Precision
0.00	0.6991
0.10	0.4945
0.20	0.4204
0.30	0.3404
0.40	0.2876
0.50	0.2408
0.60	0.1891
0.70	0.1505
0.80	0.0973
0.90	0.0534
1.00	0.0053
Average precision over all relevant docs	
non-interpolated	0.2543

Document Level Averages	
	Precision
At 5 docs	0.5200
At 10 docs	0.4580
At 15 docs	0.4307
At 20 docs	0.4010
At 30 docs	0.3593
At 100 docs	0.2338
At 200 docs	0.1740
At 500 docs	0.0975
At 1000 docs	0.0579
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2782



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	Cor7A3rrf
Run Description	Automatic, title + desc + narr
Number of Topics	50

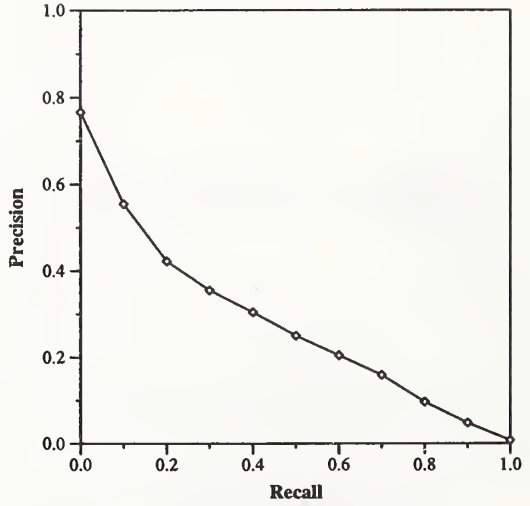
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3198

Recall Level Precision Averages	
Recall	Precision
0.00	0.7665
0.10	0.5545
0.20	0.4225
0.30	0.3556
0.40	0.3047
0.50	0.2504
0.60	0.2047
0.70	0.1591
0.80	0.0966
0.90	0.0478
1.00	0.0072

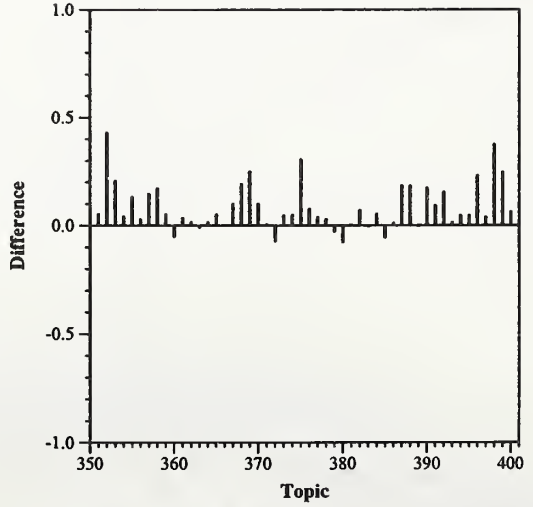
Average precision over all relevant docs	
non-interpolated	0.2674

Document Level Averages	
	Precision
At 5 docs	0.5520
At 10 docs	0.5160
At 15 docs	0.4760
At 20 docs	0.4340
At 30 docs	0.3940
At 100 docs	0.2584
At 200 docs	0.1841
At 500 docs	0.1074
At 1000 docs	0.0640

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2953



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	pirc8Aa2
Run Description	Automatic, title + desc + narr
Number of Topics	50

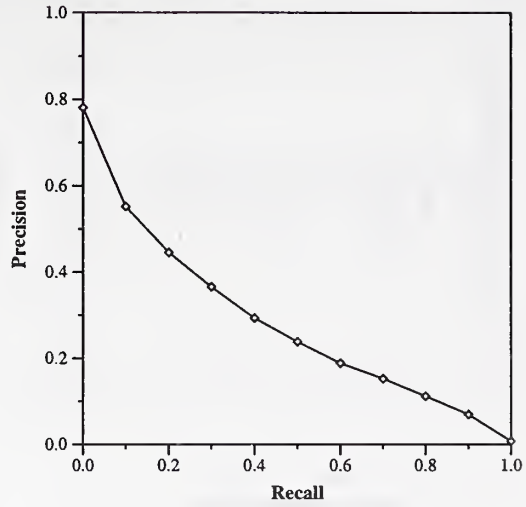
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3162

Recall Level Precision Averages	
Recall	Precision
0.00	0.7811
0.10	0.5522
0.20	0.4461
0.30	0.3662
0.40	0.2938
0.50	0.2388
0.60	0.1891
0.70	0.1532
0.80	0.1122
0.90	0.0698
1.00	0.0075

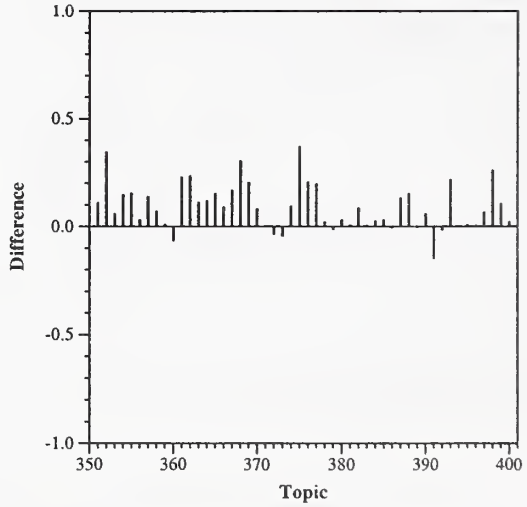
Average precision over all relevant docs	
non-interpolated	0.2723

Document Level Averages	
	Precision
At 5 docs	0.5720
At 10 docs	0.4960
At 15 docs	0.4640
At 20 docs	0.4340
At 30 docs	0.3947
At 100 docs	0.2452
At 200 docs	0.1731
At 500 docs	0.1012
At 1000 docs	0.0632

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2960



Recall-Precision Curve

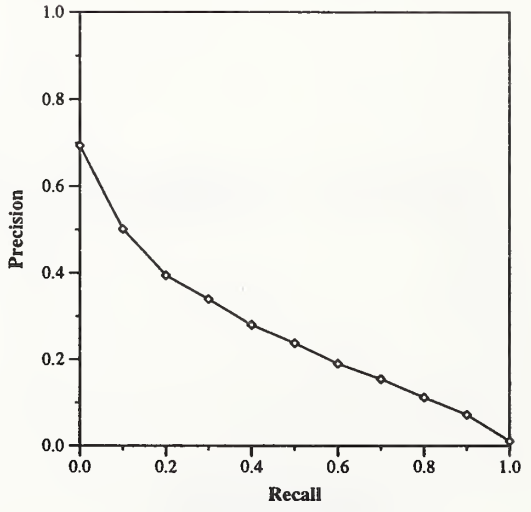


Difference from Median in Average Precision per Topic

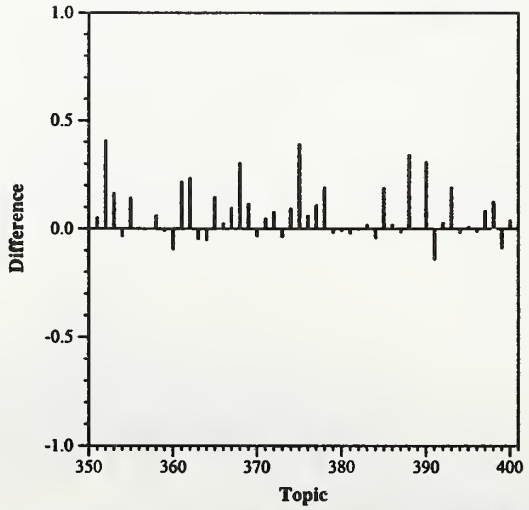
Summary Statistics	
Run Number	pirc8Ad
Run Description	Automatic, desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3034

Recall Level Precision Averages	
Recall	Precision
0.00	0.6938
0.10	0.5012
0.20	0.3948
0.30	0.3398
0.40	0.2803
0.50	0.2380
0.60	0.1907
0.70	0.1551
0.80	0.1125
0.90	0.0726
1.00	0.0109
Average precision over all relevant docs	
non-interpolated	0.2543

Document Level Averages	
	Precision
At 5 docs	0.4880
At 10 docs	0.4600
At 15 docs	0.4267
At 20 docs	0.3930
At 30 docs	0.3613
At 100 docs	0.2298
At 200 docs	0.1616
At 500 docs	0.0972
At 1000 docs	0.0607
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2831



Recall-Precision Curve

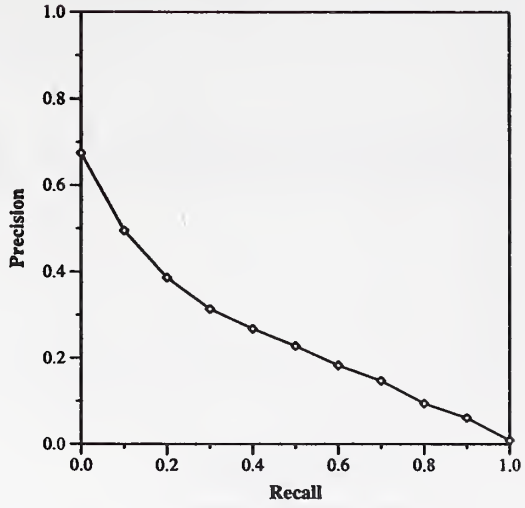


Difference from Median in Average Precision per Topic

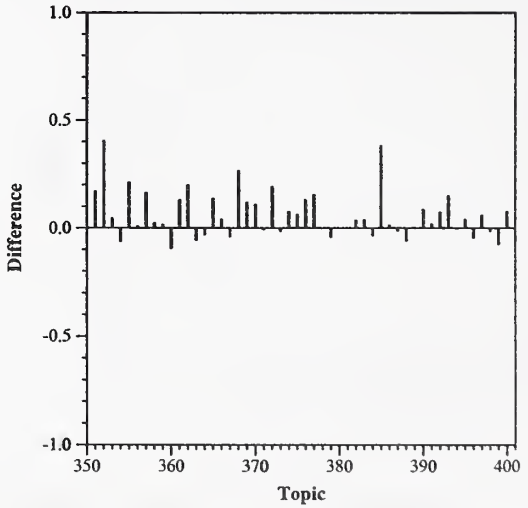
Summary Statistics	
Run Number	pirc8At
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2983

Recall Level Precision Averages	
Recall	Precision
0.00	0.6745
0.10	0.4955
0.20	0.3867
0.30	0.3138
0.40	0.2681
0.50	0.2280
0.60	0.1831
0.70	0.1471
0.80	0.0947
0.90	0.0611
1.00	0.0088
Average precision over all relevant docs	
non-interpolated	0.2427

Document Level Averages	
	Precision
At 5 docs	0.4840
At 10 docs	0.4480
At 15 docs	0.4160
At 20 docs	0.3770
At 30 docs	0.3353
At 100 docs	0.2148
At 200 docs	0.1583
At 500 docs	0.0972
At 1000 docs	0.0597
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2705



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	fub98a
Run Description	Automatic, title + desc + narr
Number of Topics	50

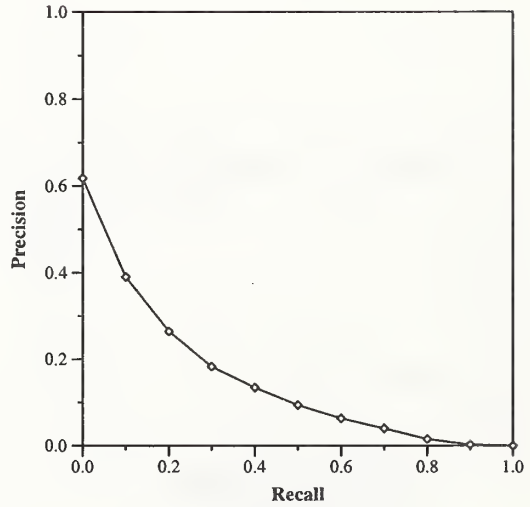
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2158

Recall Level Precision Averages	
Recall	Precision
0.00	0.6181
0.10	0.3906
0.20	0.2650
0.30	0.1835
0.40	0.1350
0.50	0.0943
0.60	0.0635
0.70	0.0402
0.80	0.0154
0.90	0.0025
1.00	0.0000

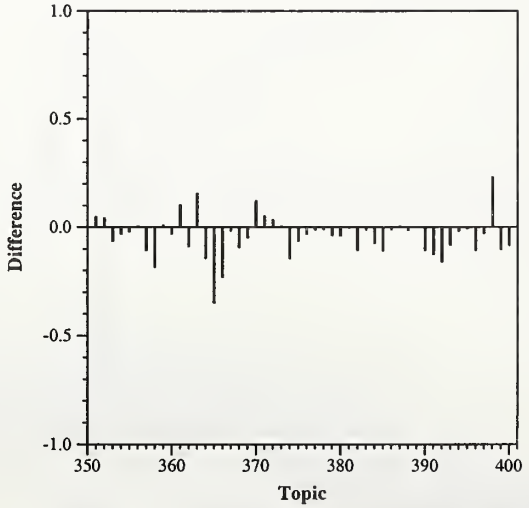
Average precision over all relevant docs	
non-interpolated	0.1409

Document Level Averages	
	Precision
At 5 docs	0.3840
At 10 docs	0.3400
At 15 docs	0.3187
At 20 docs	0.2940
At 30 docs	0.2573
At 100 docs	0.1450
At 200 docs	0.1064
At 500 docs	0.0664
At 1000 docs	0.0432

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1840



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	fub98b
Run Description	Automatic, title + desc + narr
Number of Topics	50

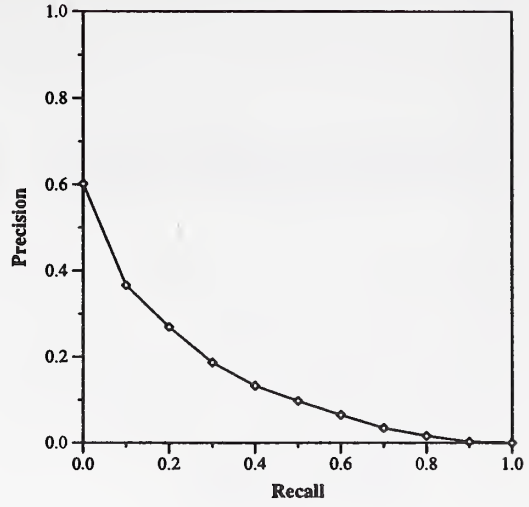
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2155

Recall Level Precision Averages	
Recall	Precision
0.00	0.6016
0.10	0.3665
0.20	0.2695
0.30	0.1869
0.40	0.1328
0.50	0.0974
0.60	0.0648
0.70	0.0347
0.80	0.0163
0.90	0.0026
1.00	0.0000

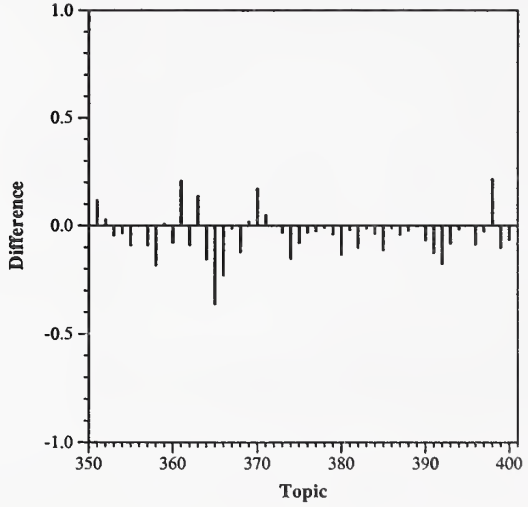
Average precision over all relevant docs	
non-interpolated	0.1390

Document Level Averages	
	Precision
At 5 docs	0.3840
At 10 docs	0.3400
At 15 docs	0.3067
At 20 docs	0.2830
At 30 docs	0.2473
At 100 docs	0.1416
At 200 docs	0.1055
At 500 docs	0.0656
At 1000 docs	0.0431

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1818



Recall-Precision Curve

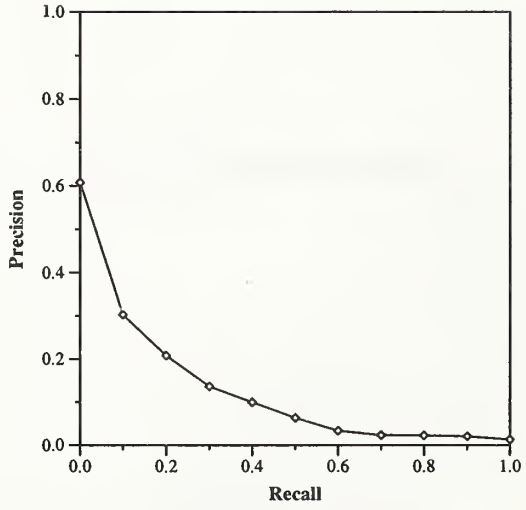


Difference from Median in Average Precision per Topic

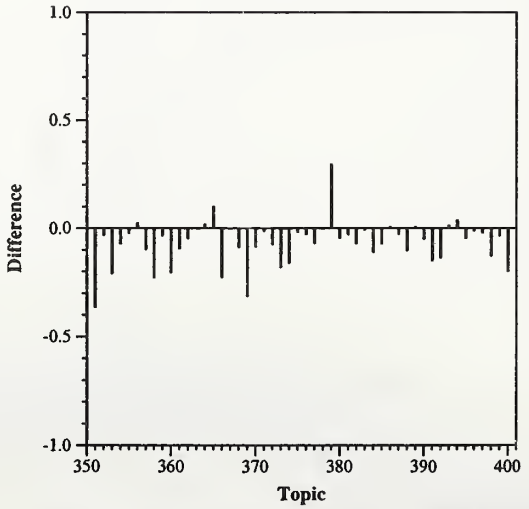
Summary Statistics	
Run Number	fsclt7a
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	9471
Relevant:	4674
Rel-ret:	950

Recall Level Precision Averages	
Recall	Precision
0.00	0.6078
0.10	0.3026
0.20	0.2080
0.30	0.1367
0.40	0.1001
0.50	0.0639
0.60	0.0339
0.70	0.0236
0.80	0.0228
0.90	0.0208
1.00	0.0132
Average precision over all relevant docs	
non-interpolated	0.1146

Document Level Averages	
	Precision
At 5 docs	0.3720
At 10 docs	0.3380
At 15 docs	0.3027
At 20 docs	0.2820
At 30 docs	0.2433
At 100 docs	0.1184
At 200 docs	0.0715
At 500 docs	0.0354
At 1000 docs	0.0190
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1678



Recall-Precision Curve

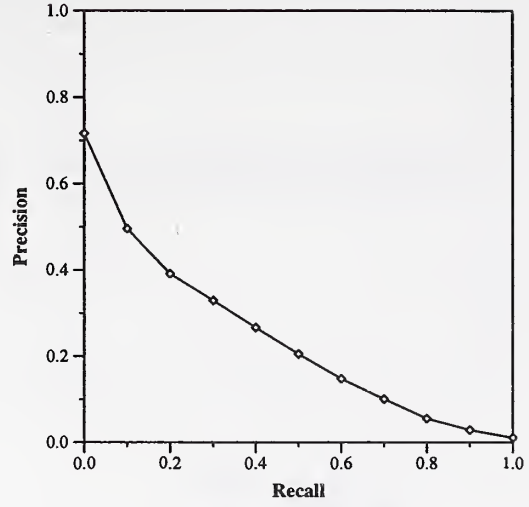


Difference from Median in Average Precision per Topic

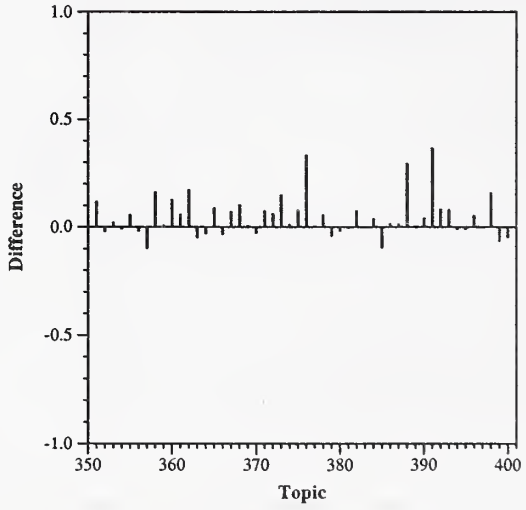
Summary Statistics	
Run Number	FLab7ad
Run Description	Automatic, desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2686

Recall Level Precision Averages	
Recall	Precision
0.00	0.7166
0.10	0.4962
0.20	0.3917
0.30	0.3298
0.40	0.2668
0.50	0.2055
0.60	0.1480
0.70	0.1006
0.80	0.0555
0.90	0.0287
1.00	0.0106
Average precision over all relevant docs	
non-interpolated	0.2296

Document Level Averages	
	Precision
At 5 docs	0.5120
At 10 docs	0.4780
At 15 docs	0.4333
At 20 docs	0.4040
At 30 docs	0.3593
At 100 docs	0.2182
At 200 docs	0.1505
At 500 docs	0.0867
At 1000 docs	0.0537
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2684



Recall-Precision Curve

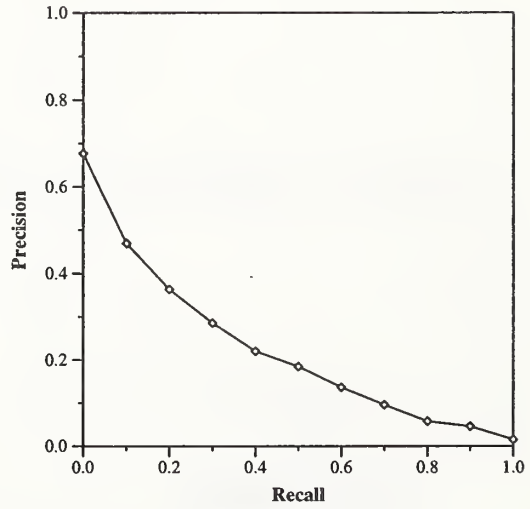


Difference from Median in Average Precision per Topic

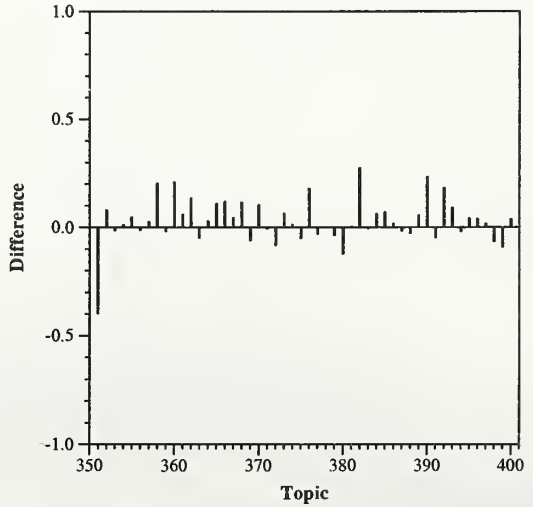
Summary Statistics	
Run Number	FLab7at
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2624

Recall Level Precision Averages	
Recall	Precision
0.00	0.6771
0.10	0.4698
0.20	0.3635
0.30	0.2857
0.40	0.2206
0.50	0.1850
0.60	0.1370
0.70	0.0963
0.80	0.0579
0.90	0.0461
1.00	0.0154
Average precision over all relevant docs	
non-interpolated	0.2126

Document Level Averages	
	Precision
At 5 docs	0.4840
At 10 docs	0.4280
At 15 docs	0.3973
At 20 docs	0.3750
At 30 docs	0.3333
At 100 docs	0.2148
At 200 docs	0.1558
At 500 docs	0.0882
At 1000 docs	0.0525
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2414



Recall-Precision Curve

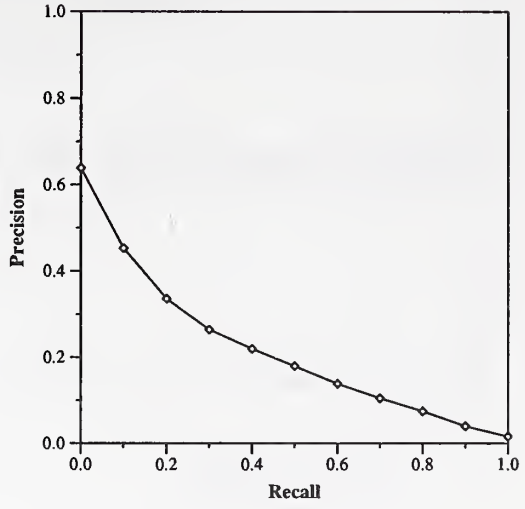


Difference from Median in Average Precision per Topic

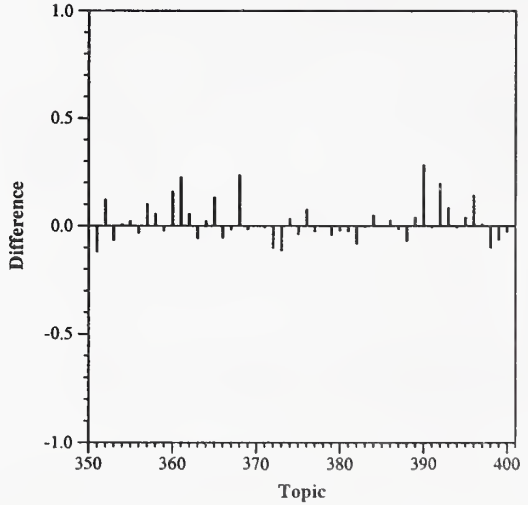
Summary Statistics	
Run Number	FLab7atE
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2695

Recall Level Precision Averages	
Recall	Precision
0.00	0.6388
0.10	0.4530
0.20	0.3358
0.30	0.2644
0.40	0.2199
0.50	0.1796
0.60	0.1387
0.70	0.1052
0.80	0.0749
0.90	0.0402
1.00	0.0161
Average precision over all relevant docs	
non-interpolated	0.2020

Document Level Averages	
	Precision
At 5 docs	0.4240
At 10 docs	0.4100
At 15 docs	0.3760
At 20 docs	0.3500
At 30 docs	0.3067
At 100 docs	0.2034
At 200 docs	0.1462
At 500 docs	0.0894
At 1000 docs	0.0539
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2363



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	gersh2
Run Description	Automatic, title + desc
Number of Topics	50

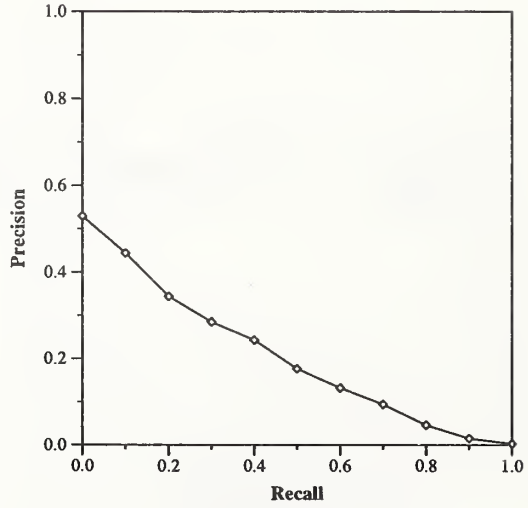
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2140

Recall Level Precision Averages	
Recall	Precision
0.00	0.5292
0.10	0.4443
0.20	0.3444
0.30	0.2853
0.40	0.2433
0.50	0.1770
0.60	0.1327
0.70	0.0944
0.80	0.0463
0.90	0.0153
1.00	0.0031

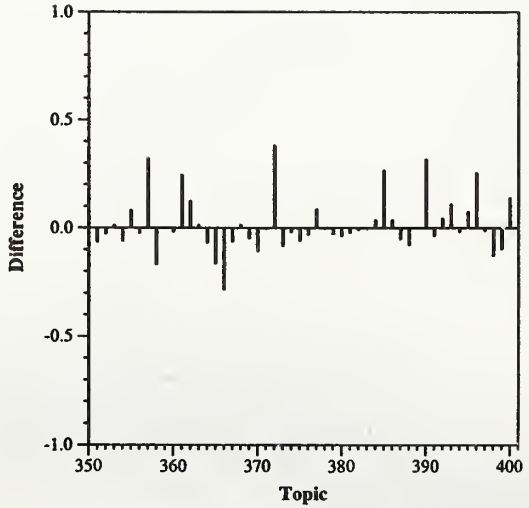
Average precision over all relevant docs	
non-interpolated	0.1972

Document Level Averages	
	Precision
At 5 docs	0.3880
At 10 docs	0.3740
At 15 docs	0.3613
At 20 docs	0.3410
At 30 docs	0.2987
At 100 docs	0.1854
At 200 docs	0.1312
At 500 docs	0.0703
At 1000 docs	0.0428

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2256



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	gersh3
Run Description	Automatic, title + desc
Number of Topics	50

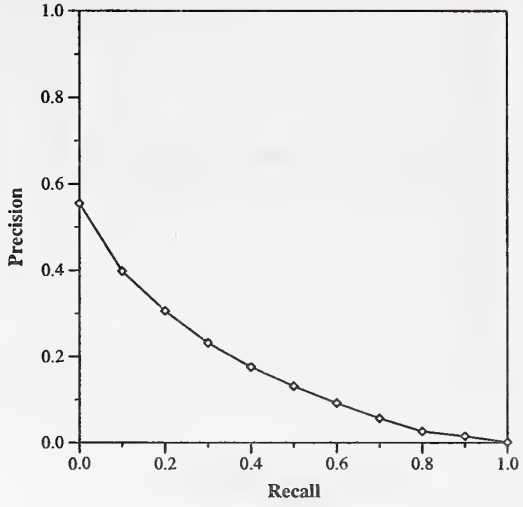
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	1816

Recall Level Precision Averages	
Recall	Precision
0.00	0.5554
0.10	0.3983
0.20	0.3063
0.30	0.2321
0.40	0.1760
0.50	0.1314
0.60	0.0921
0.70	0.0569
0.80	0.0264
0.90	0.0152
1.00	0.0011

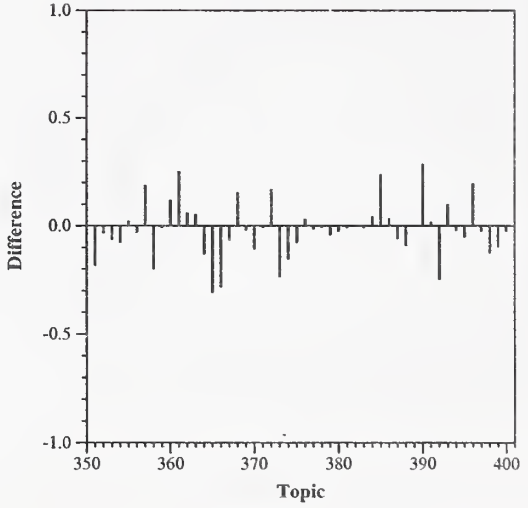
Average precision over all relevant docs	
non-interpolated	0.1648

Document Level Averages	
	Precision
At 5 docs	0.3640
At 10 docs	0.3500
At 15 docs	0.3267
At 20 docs	0.3030
At 30 docs	0.2653
At 100 docs	0.1642
At 200 docs	0.1128
At 500 docs	0.0601
At 1000 docs	0.0363

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2021



Recall-Precision Curve

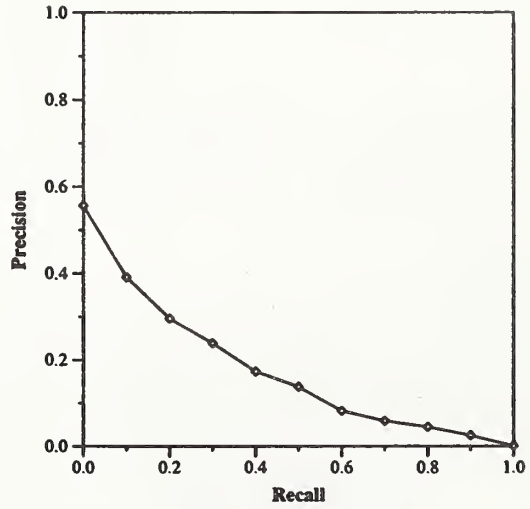


Difference from Median in Average Precision per Topic

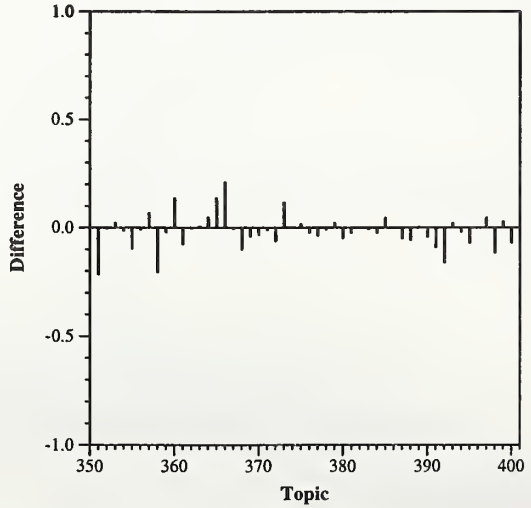
Summary Statistics	
Run Number	ibmg98a
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	46128
Relevant:	4674
Rel-ret:	2096

Recall Level Precision Averages	
Recall	Precision
0.00	0.5566
0.10	0.3917
0.20	0.2960
0.30	0.2392
0.40	0.1736
0.50	0.1380
0.60	0.0821
0.70	0.0588
0.80	0.0449
0.90	0.0253
1.00	0.0005
Average precision over all relevant docs	
non-interpolated	0.1660

Document Level Averages	
	Precision
At 5 docs	0.3680
At 10 docs	0.3600
At 15 docs	0.3427
At 20 docs	0.3230
At 30 docs	0.2913
At 100 docs	0.1698
At 200 docs	0.1227
At 500 docs	0.0698
At 1000 docs	0.0419
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2151



Recall-Precision Curve

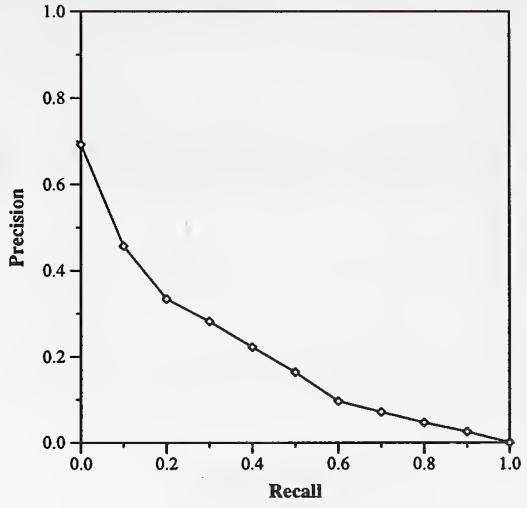


Difference from Median in Average Precision per Topic

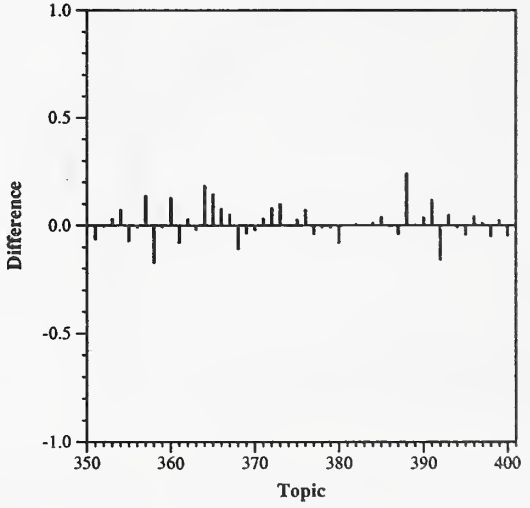
Summary Statistics	
Run Number	ibmg98b
Run Description	Automatic, title + desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2502

Recall Level Precision Averages	
Recall	Precision
0.00	0.6926
0.10	0.4577
0.20	0.3346
0.30	0.2825
0.40	0.2230
0.50	0.1645
0.60	0.0970
0.70	0.0718
0.80	0.0471
0.90	0.0259
1.00	0.0003
Average precision over all relevant docs	
non-interpolated	0.1953

Document Level Averages	
	Precision
At 5 docs	0.4520
At 10 docs	0.4140
At 15 docs	0.3787
At 20 docs	0.3670
At 30 docs	0.3313
At 100 docs	0.2014
At 200 docs	0.1418
At 500 docs	0.0805
At 1000 docs	0.0500
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2437



Recall-Precision Curve

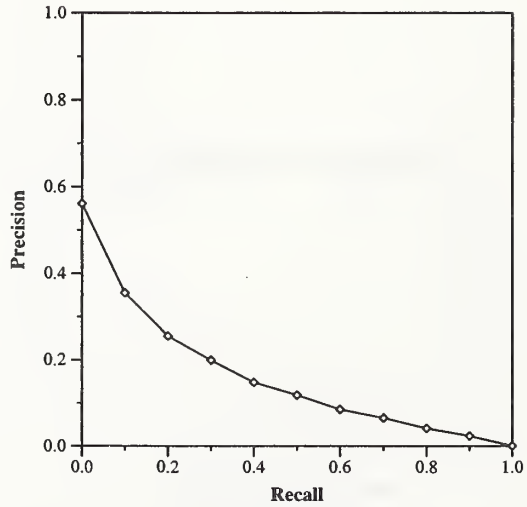


Difference from Median in Average Precision per Topic

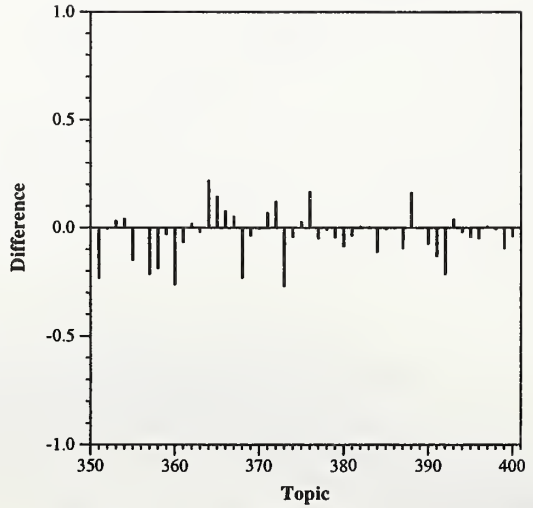
Summary Statistics	
Run Number	ibmg98c
Run Description	Automatic, title + desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2027

Recall Level Precision Averages	
Recall	Precision
0.00	0.5614
0.10	0.3553
0.20	0.2552
0.30	0.1995
0.40	0.1478
0.50	0.1185
0.60	0.0851
0.70	0.0653
0.80	0.0412
0.90	0.0236
1.00	0.0003
Average precision over all relevant docs	
non-interpolated	0.1486

Document Level Averages	
	Precision
At 5 docs	0.3560
At 10 docs	0.3300
At 15 docs	0.2973
At 20 docs	0.2920
At 30 docs	0.2573
At 100 docs	0.1568
At 200 docs	0.1080
At 500 docs	0.0642
At 1000 docs	0.0405
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2037



Recall-Precision Curve

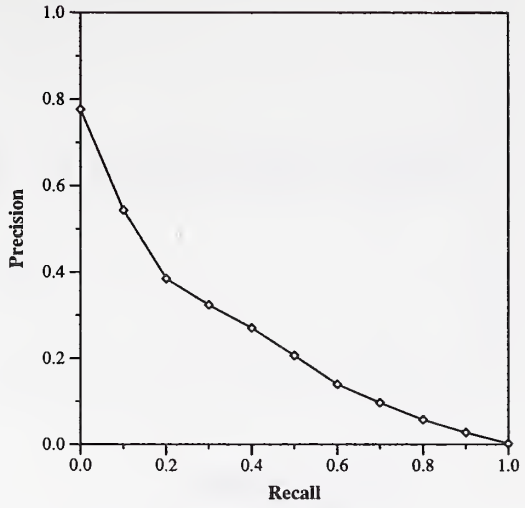


Difference from Median in Average Precision per Topic

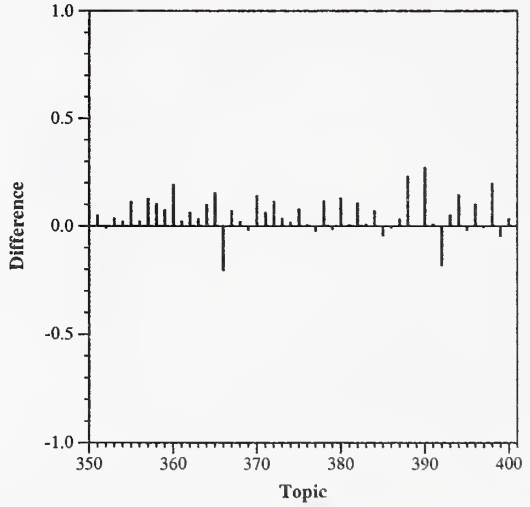
Summary Statistics	
Run Number	ibms98a
Run Description	Automatic, desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2789

Recall Level Precision Averages	
Recall	Precision
0.00	0.7767
0.10	0.5434
0.20	0.3852
0.30	0.3241
0.40	0.2707
0.50	0.2068
0.60	0.1399
0.70	0.0970
0.80	0.0573
0.90	0.0276
1.00	0.0020
Average precision over all relevant docs	
non-interpolated	0.2336

Document Level Averages	
	Precision
At 5 docs	0.5360
At 10 docs	0.4840
At 15 docs	0.4320
At 20 docs	0.4010
At 30 docs	0.3480
At 100 docs	0.2172
At 200 docs	0.1542
At 500 docs	0.0908
At 1000 docs	0.0558
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2800



Recall-Precision Curve

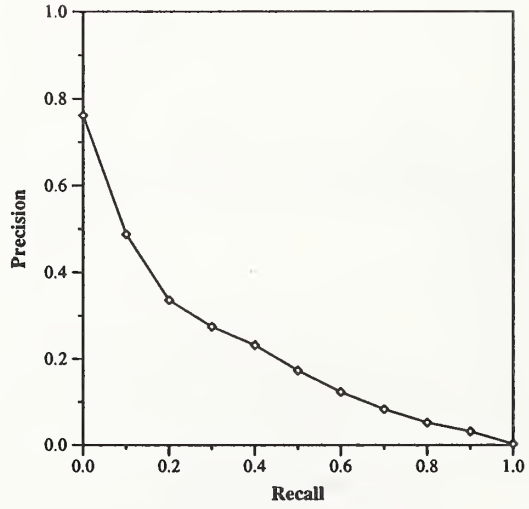


Difference from Median in Average Precision per Topic

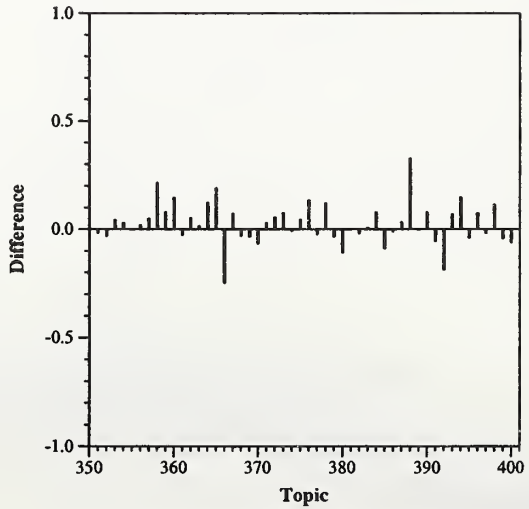
Summary Statistics	
Run Number	ibms98b
Run Description	Automatic, desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2562

Recall Level Precision Averages	
Recall	Precision
0.00	0.7617
0.10	0.4881
0.20	0.3359
0.30	0.2746
0.40	0.2320
0.50	0.1734
0.60	0.1235
0.70	0.0836
0.80	0.0526
0.90	0.0320
1.00	0.0026
Average precision over all relevant docs	
non-interpolated	0.2075

Document Level Averages	
	Precision
At 5 docs	0.4880
At 10 docs	0.4340
At 15 docs	0.4053
At 20 docs	0.3770
At 30 docs	0.3260
At 100 docs	0.2006
At 200 docs	0.1347
At 500 docs	0.0807
At 1000 docs	0.0512
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2472



Recall-Precision Curve

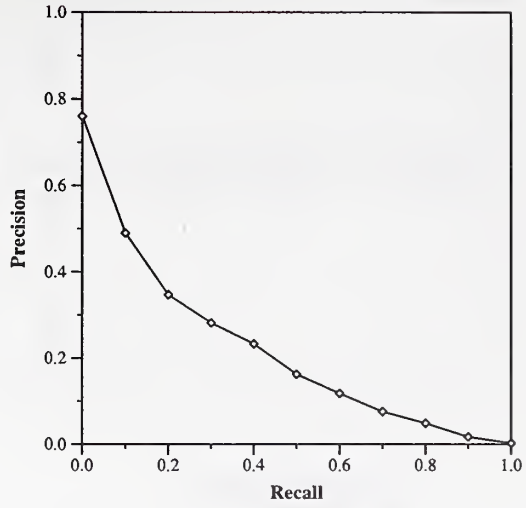


Difference from Median in Average Precision per Topic

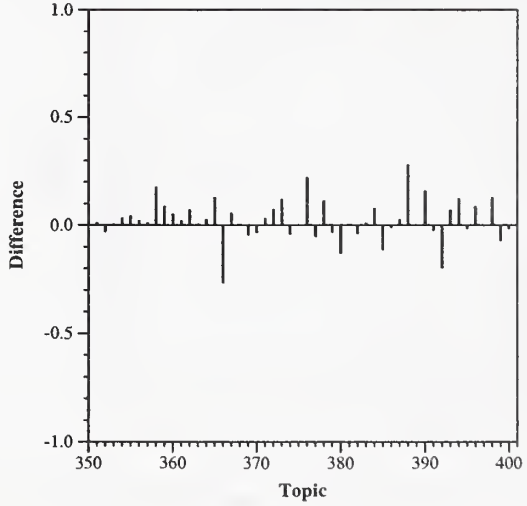
Summary Statistics	
Run Number	ibms98c
Run Description	Automatic, desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2543

Recall Level Precision Averages	
Recall	Precision
0.00	0.7606
0.10	0.4901
0.20	0.3475
0.30	0.2822
0.40	0.2334
0.50	0.1629
0.60	0.1183
0.70	0.0758
0.80	0.0490
0.90	0.0172
1.00	0.0024
Average precision over all relevant docs	
non-interpolated	0.2047

Document Level Averages	
	Precision
At 5 docs	0.4760
At 10 docs	0.4180
At 15 docs	0.3973
At 20 docs	0.3750
At 30 docs	0.3307
At 100 docs	0.2022
At 200 docs	0.1406
At 500 docs	0.0817
At 1000 docs	0.0509
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2503



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	iit98au1
Run Description	Automatic, title + desc + narr
Number of Topics	50

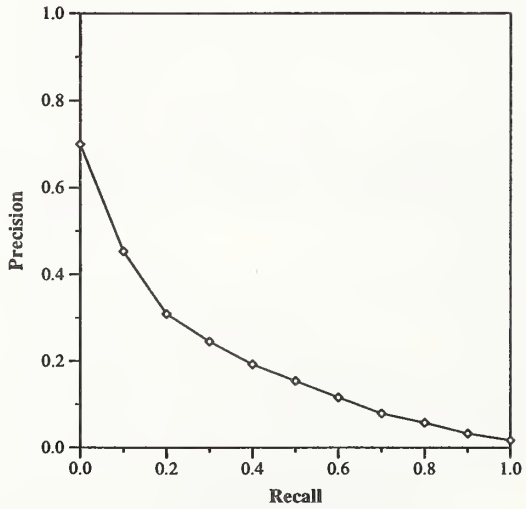
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2505

Recall Level Precision Averages	
Recall	Precision
0.00	0.7000
0.10	0.4536
0.20	0.3096
0.30	0.2457
0.40	0.1928
0.50	0.1544
0.60	0.1164
0.70	0.0790
0.80	0.0579
0.90	0.0328
1.00	0.0171

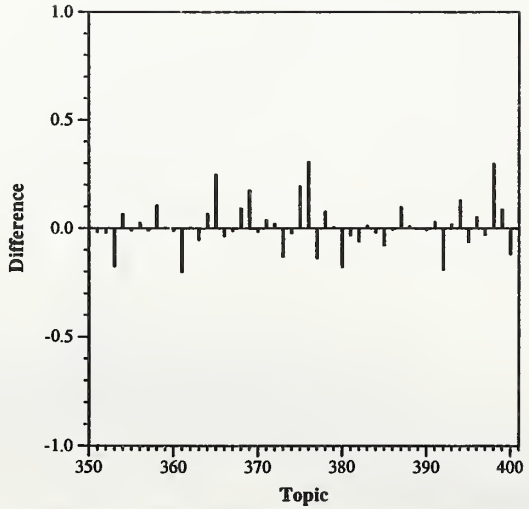
Average precision over all relevant docs	
non-interpolated	0.1929

Document Level Averages	
	Precision
At 5 docs	0.5000
At 10 docs	0.4500
At 15 docs	0.4213
At 20 docs	0.3840
At 30 docs	0.3300
At 100 docs	0.1958
At 200 docs	0.1380
At 500 docs	0.0803
At 1000 docs	0.0501

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2403



Recall-Precision Curve

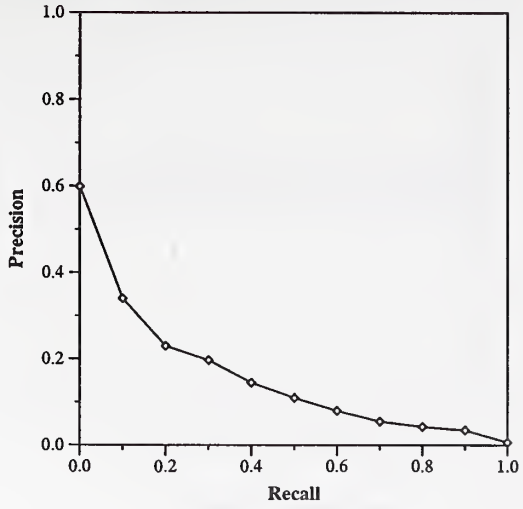


Difference from Median in Average Precision per Topic

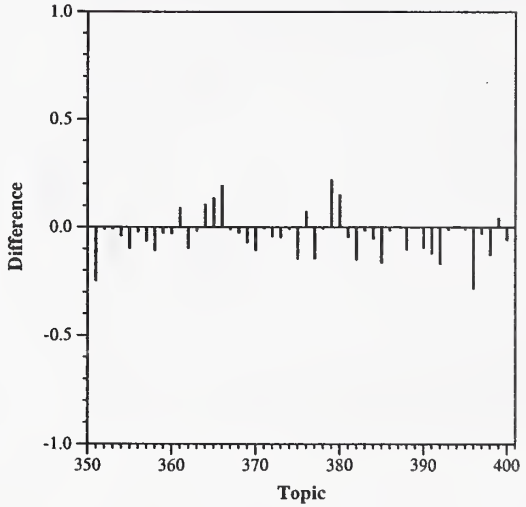
Summary Statistics	
Run Number	iit98au2
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	26763
Relevant:	4674
Rel-ret:	1676

Recall Level Precision Averages	
Recall	Precision
0.00	0.5988
0.10	0.3404
0.20	0.2302
0.30	0.1973
0.40	0.1452
0.50	0.1097
0.60	0.0795
0.70	0.0550
0.80	0.0428
0.90	0.0349
1.00	0.0065
Average precision over all relevant docs	
non-interpolated	0.1459

Document Level Averages	
	Precision
At 5 docs	0.3080
At 10 docs	0.3200
At 15 docs	0.2960
At 20 docs	0.2770
At 30 docs	0.2493
At 100 docs	0.1500
At 200 docs	0.1071
At 500 docs	0.0579
At 1000 docs	0.0335
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2000



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	ic98san3
Run Description	Automatic, title + desc
Number of Topics	50

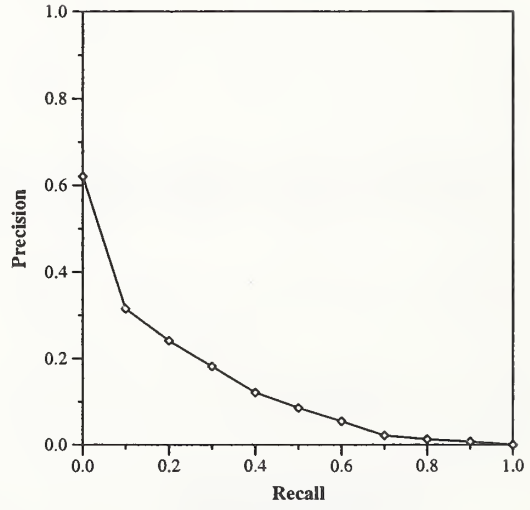
Total number of documents over all topics	
Retrieved:	49124
Relevant:	4674
Rel-ret:	2031

Recall Level Precision Averages	
Recall	Precision
0.00	0.6209
0.10	0.3153
0.20	0.2413
0.30	0.1820
0.40	0.1214
0.50	0.0859
0.60	0.0547
0.70	0.0218
0.80	0.0129
0.90	0.0075
1.00	0.0000

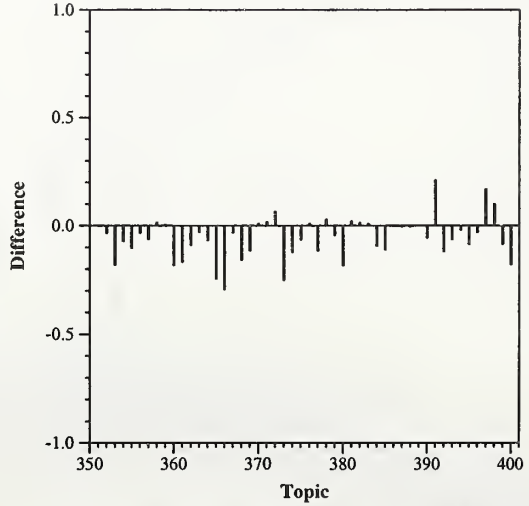
Average precision over all relevant docs	
non-interpolated	0.1259

Document Level Averages	
	Precision
At 5 docs	0.3600
At 10 docs	0.3260
At 15 docs	0.2853
At 20 docs	0.2740
At 30 docs	0.2373
At 100 docs	0.1454
At 200 docs	0.1020
At 500 docs	0.0634
At 1000 docs	0.0406

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1745



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	ic98san4
Run Description	Automatic, title + desc + narr
Number of Topics	50

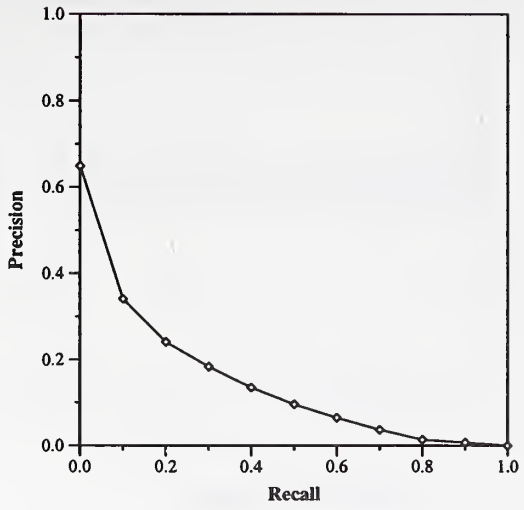
Total number of documents over all topics	
Retrieved:	49124
Relevant:	4674
Rel-ret:	2079

Recall Level Precision Averages	
Recall	Precision
0.00	0.6491
0.10	0.3413
0.20	0.2410
0.30	0.1839
0.40	0.1352
0.50	0.0959
0.60	0.0649
0.70	0.0370
0.80	0.0138
0.90	0.0073
1.00	0.0000

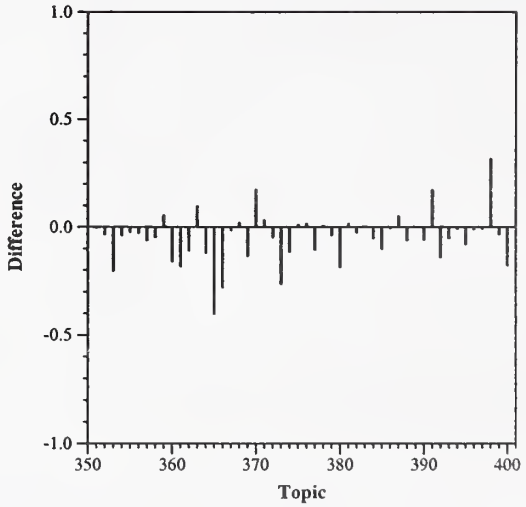
Average precision over all relevant docs	
non-interpolated	0.1333

Document Level Averages	
	Precision
At 5 docs	0.3880
At 10 docs	0.3560
At 15 docs	0.3067
At 20 docs	0.2810
At 30 docs	0.2500
At 100 docs	0.1648
At 200 docs	0.1150
At 500 docs	0.0676
At 1000 docs	0.0416

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1878



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	MerAdRbtd
Run Description	Automatic, title + desc
Number of Topics	50

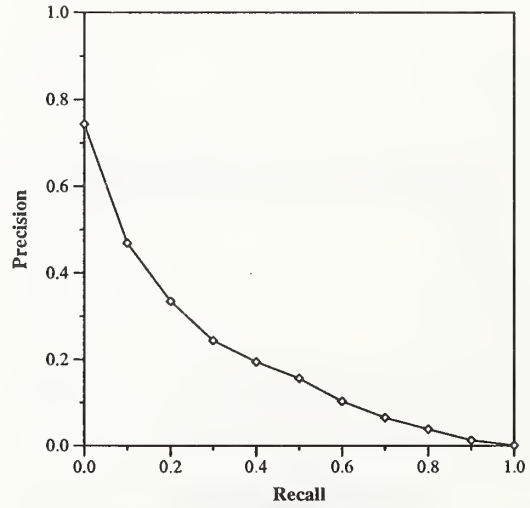
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2543

Recall Level Precision Averages	
Recall	Precision
0.00	0.7436
0.10	0.4693
0.20	0.3354
0.30	0.2442
0.40	0.1948
0.50	0.1566
0.60	0.1033
0.70	0.0655
0.80	0.0388
0.90	0.0131
1.00	0.0010

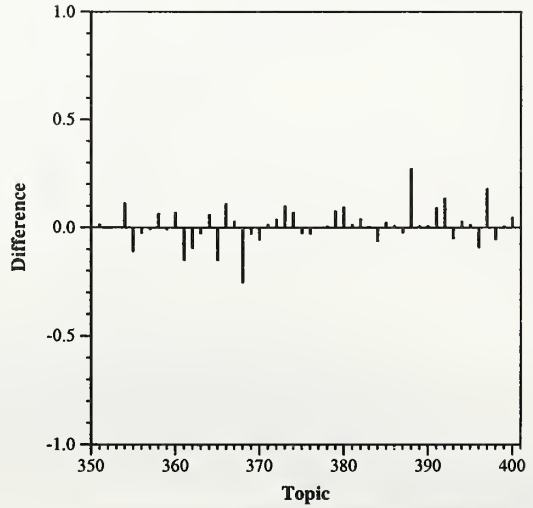
Average precision over all relevant docs	
non-interpolated	0.1918

Document Level Averages	
	Precision
At 5 docs	0.4840
At 10 docs	0.4240
At 15 docs	0.4027
At 20 docs	0.3730
At 30 docs	0.3240
At 100 docs	0.1996
At 200 docs	0.1408
At 500 docs	0.0803
At 1000 docs	0.0509

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2380



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	MerAdRbtnd
Run Description	Automatic, title + desc + narr
Number of Topics	50

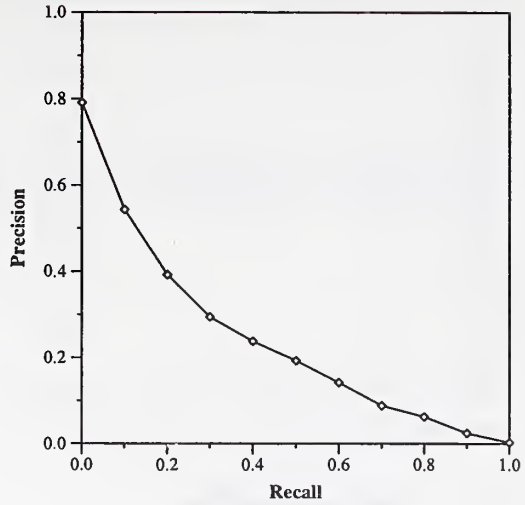
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2818

Recall Level Precision Averages	
Recall	Precision
0.00	0.7912
0.10	0.5436
0.20	0.3927
0.30	0.2947
0.40	0.2382
0.50	0.1935
0.60	0.1425
0.70	0.0889
0.80	0.0629
0.90	0.0246
1.00	0.0027

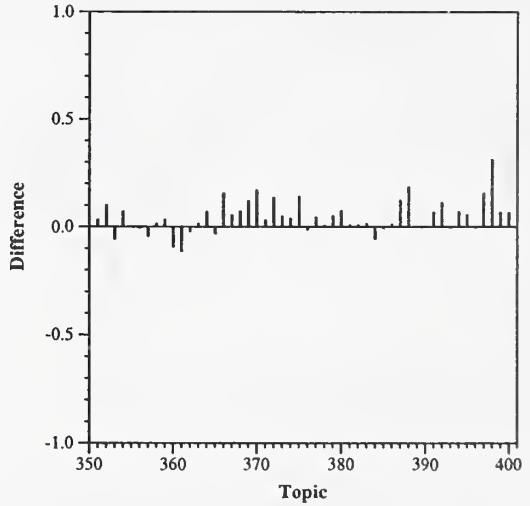
Average precision over all relevant docs	
non-interpolated	0.2278

Document Level Averages	
	Precision
At 5 docs	0.5680
At 10 docs	0.5060
At 15 docs	0.4587
At 20 docs	0.4230
At 30 docs	0.3713
At 100 docs	0.2278
At 200 docs	0.1614
At 500 docs	0.0932
At 1000 docs	0.0564

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2746



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	MerTetAdtnd
Run Description	Automatic, title + desc + narr
Number of Topics	50

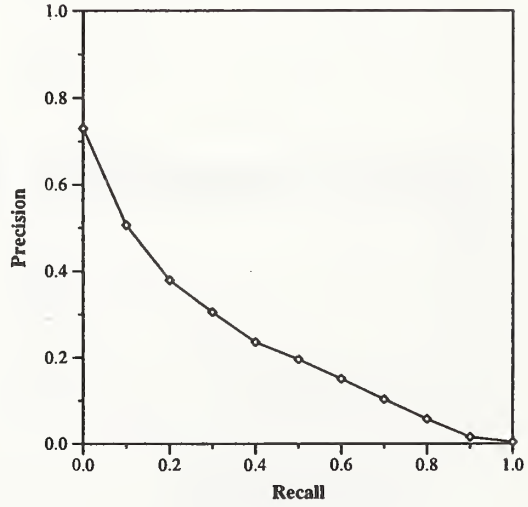
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2624

Recall Level Precision Averages	
Recall	Precision
0.00	0.7299
0.10	0.5068
0.20	0.3795
0.30	0.3056
0.40	0.2361
0.50	0.1959
0.60	0.1510
0.70	0.1035
0.80	0.0576
0.90	0.0158
1.00	0.0043

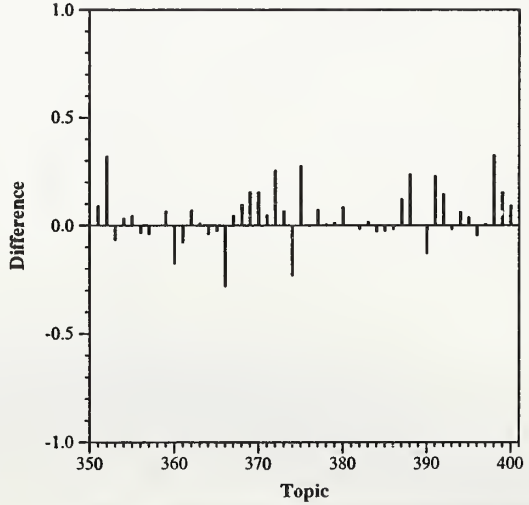
Average precision over all relevant docs	
non-interpolated	0.2237

Document Level Averages	
	Precision
At 5 docs	0.5240
At 10 docs	0.4780
At 15 docs	0.4387
At 20 docs	0.4090
At 30 docs	0.3580
At 100 docs	0.2128
At 200 docs	0.1513
At 500 docs	0.0853
At 1000 docs	0.0525

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2617



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	APL985L
Run Description	Automatic, title + desc + narr
Number of Topics	50

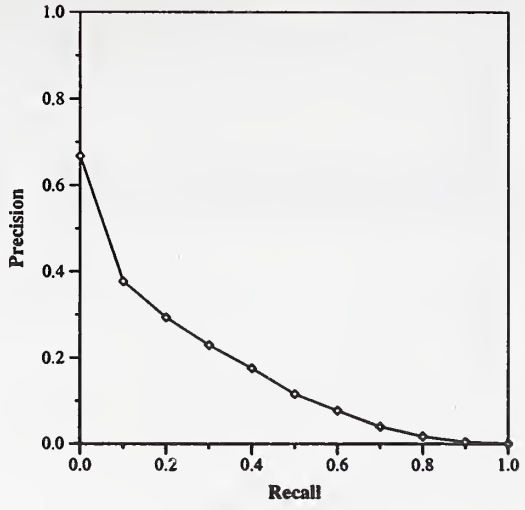
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2363

Recall Level Precision Averages	
Recall	Precision
0.00	0.6681
0.10	0.3777
0.20	0.2938
0.30	0.2298
0.40	0.1761
0.50	0.1160
0.60	0.0774
0.70	0.0402
0.80	0.0173
0.90	0.0041
1.00	0.0002

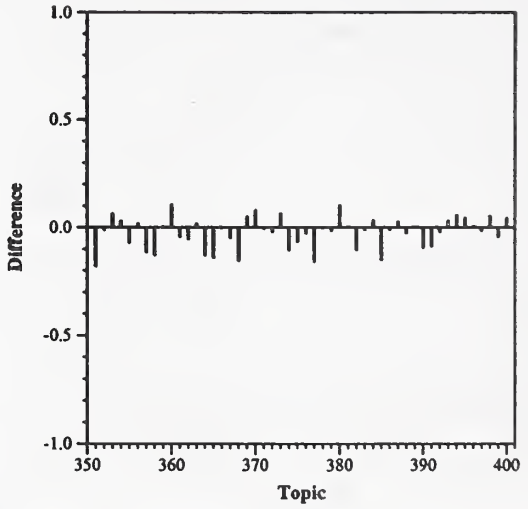
Average precision over all relevant docs	
non-interpolated	0.1576

Document Level Averages	
	Precision
At 5 docs	0.4160
At 10 docs	0.3680
At 15 docs	0.3280
At 20 docs	0.2970
At 30 docs	0.2573
At 100 docs	0.1706
At 200 docs	0.1228
At 500 docs	0.0734
At 1000 docs	0.0473

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2086



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	APL985LC
Run Description	Automatic, title + desc + narr
Number of Topics	50

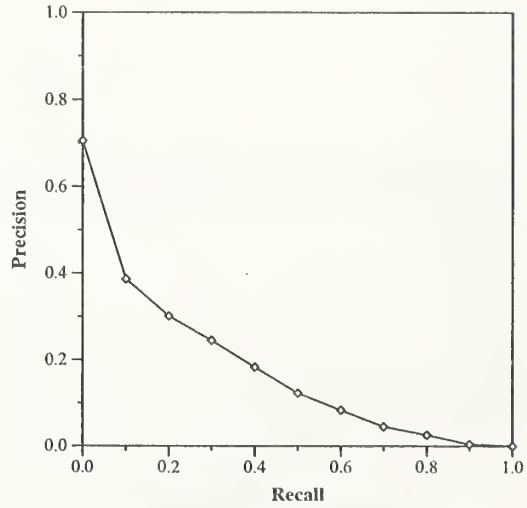
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2363

Recall Level Precision Averages	
Recall	Precision
0.00	0.7054
0.10	0.3868
0.20	0.3012
0.30	0.2447
0.40	0.1831
0.50	0.1229
0.60	0.0833
0.70	0.0452
0.80	0.0259
0.90	0.0042
1.00	0.0001

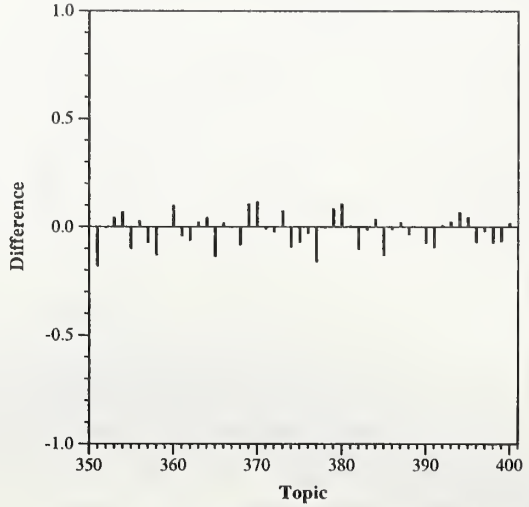
Average precision over all relevant docs	
non-interpolated	0.1649

Document Level Averages	
	Precision
At 5 docs	0.4280
At 10 docs	0.3820
At 15 docs	0.3440
At 20 docs	0.3080
At 30 docs	0.2733
At 100 docs	0.1772
At 200 docs	0.1230
At 500 docs	0.0717
At 1000 docs	0.0473

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2156



Recall-Precision Curve

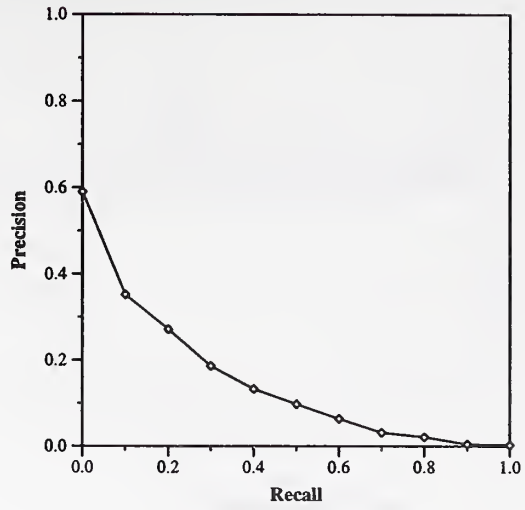


Difference from Median in Average Precision per Topic

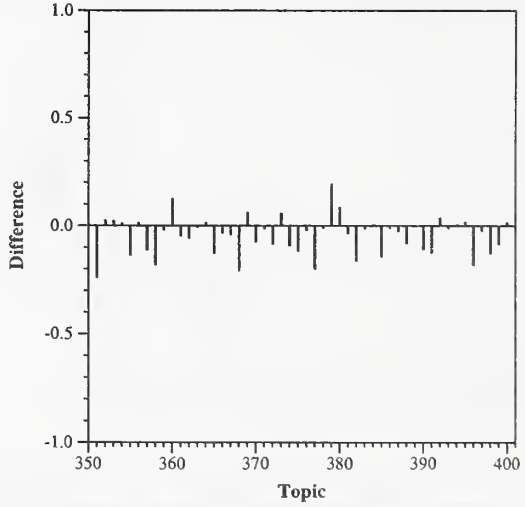
Summary Statistics	
Run Number	APL985SC
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	1931

Recall Level Precision Averages	
Recall	Precision
0.00	0.5902
0.10	0.3521
0.20	0.2717
0.30	0.1865
0.40	0.1332
0.50	0.0979
0.60	0.0639
0.70	0.0318
0.80	0.0215
0.90	0.0048
1.00	0.0025
Average precision over all relevant docs	
non-interpolated	0.1370

Document Level Averages	
	Precision
At 5 docs	0.3560
At 10 docs	0.3120
At 15 docs	0.2733
At 20 docs	0.2620
At 30 docs	0.2307
At 100 docs	0.1494
At 200 docs	0.1048
At 500 docs	0.0604
At 1000 docs	0.0386
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1855



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	dsir07a01
Run Description	Automatic, title + desc
Number of Topics	50

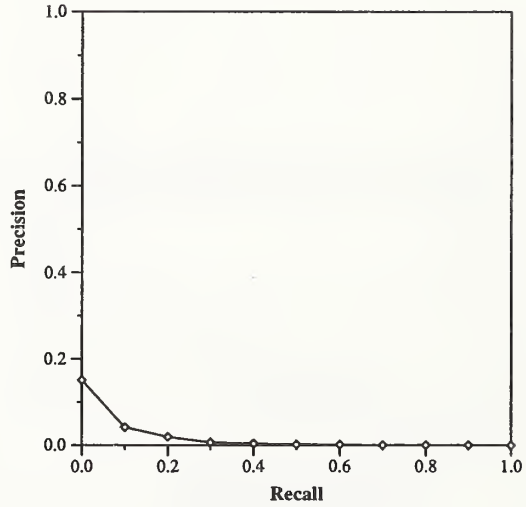
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	470

Recall Level Precision Averages	
Recall	Precision
0.00	0.1511
0.10	0.0418
0.20	0.0196
0.30	0.0068
0.40	0.0040
0.50	0.0020
0.60	0.0018
0.70	0.0005
0.80	0.0005
0.90	0.0004
1.00	0.0000

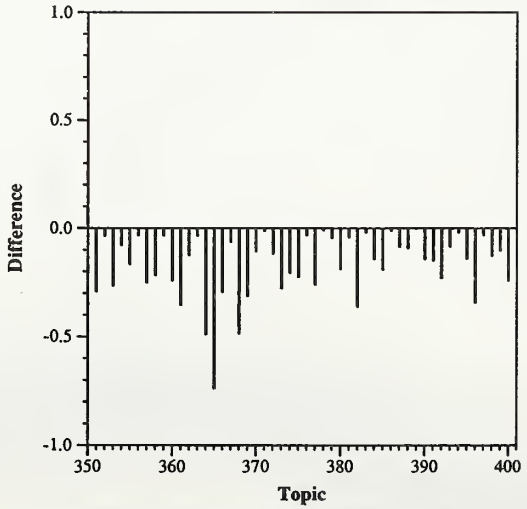
Average precision over all relevant docs	
non-interpolated	0.0117

Document Level Averages	
	Precision
At 5 docs	0.0720
At 10 docs	0.0520
At 15 docs	0.0507
At 20 docs	0.0470
At 30 docs	0.0413
At 100 docs	0.0252
At 200 docs	0.0203
At 500 docs	0.0132
At 1000 docs	0.0094

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0236



Recall-Precision Curve



Difference from Median in Average Precision per Topic

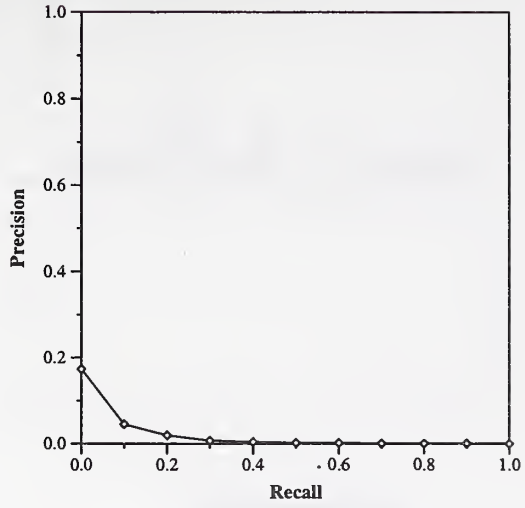
Summary Statistics	
Run Number	dsir07a02
Run Description	Automatic, title + desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	504

Recall Level Precision Averages	
Recall	Precision
0.00	0.1739
0.10	0.0454
0.20	0.0195
0.30	0.0072
0.40	0.0044
0.50	0.0021
0.60	0.0018
0.70	0.0005
0.80	0.0005
0.90	0.0004
1.00	0.0000

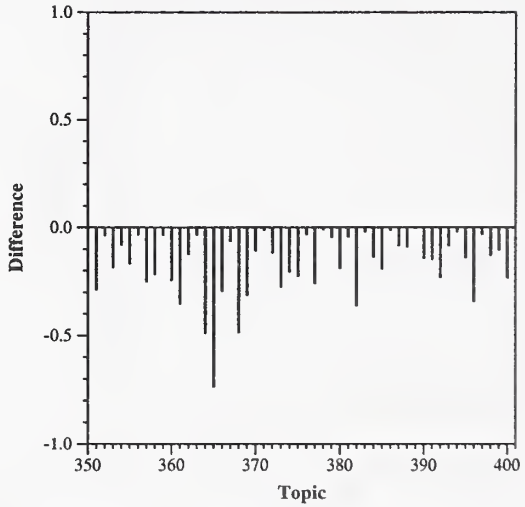
Average precision over all relevant docs	
non-interpolated	0.0137

Document Level Averages	
	Precision
At 5 docs	0.0960
At 10 docs	0.0740
At 15 docs	0.0627
At 20 docs	0.0540
At 30 docs	0.0487
At 100 docs	0.0276
At 200 docs	0.0218
At 500 docs	0.0140
At 1000 docs	0.0101

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0259



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	KD70000
Run Description	Automatic, title + desc + narr
Number of Topics	50

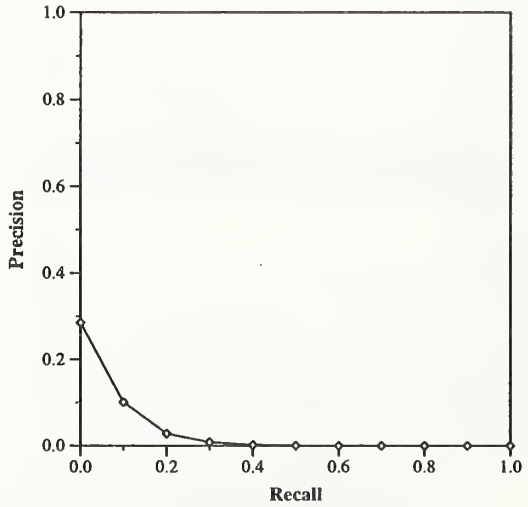
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	342

Recall Level Precision Averages	
Recall	Precision
0.00	0.2854
0.10	0.1014
0.20	0.0283
0.30	0.0087
0.40	0.0023
0.50	0.0004
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000

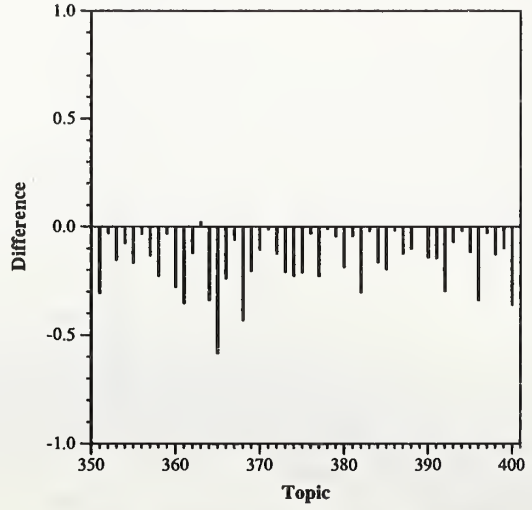
Average precision over all relevant docs	
non-interpolated	0.0250

Document Level Averages	
	Precision
At 5 docs	0.1280
At 10 docs	0.1220
At 15 docs	0.1040
At 20 docs	0.0960
At 30 docs	0.0880
At 100 docs	0.0450
At 200 docs	0.0281
At 500 docs	0.0130
At 1000 docs	0.0068

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0571



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	KD71010q
Run Description	Automatic, title + desc + narr
Number of Topics	50

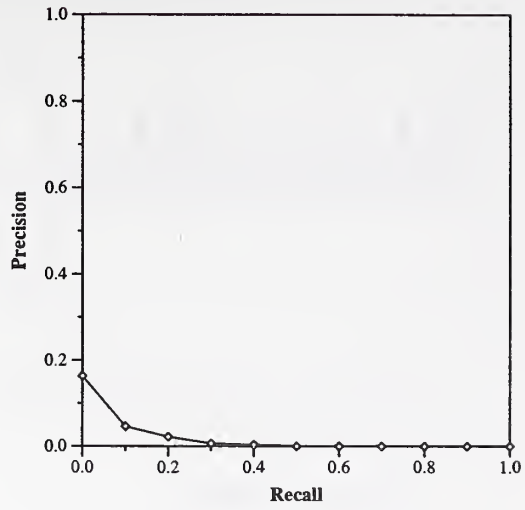
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	376

Recall Level Precision Averages	
Recall	Precision
0.00	0.1633
0.10	0.0462
0.20	0.0221
0.30	0.0067
0.40	0.0034
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000

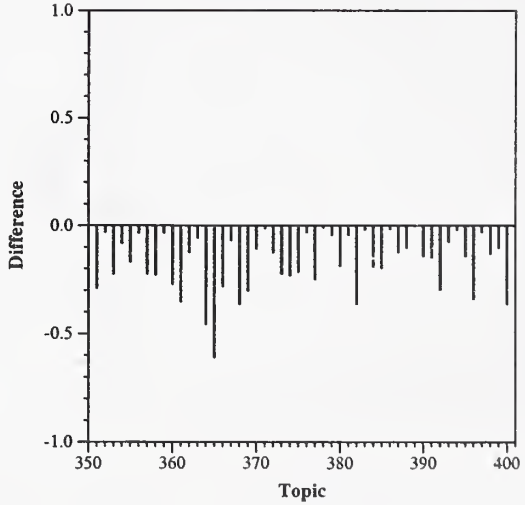
Average precision over all relevant docs	
non-interpolated	0.0125

Document Level Averages	
	Precision
At 5 docs	0.0760
At 10 docs	0.0560
At 15 docs	0.0520
At 20 docs	0.0440
At 30 docs	0.0393
At 100 docs	0.0228
At 200 docs	0.0172
At 500 docs	0.0118
At 1000 docs	0.0075

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0305



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	KD71010s
Run Description	Automatic, title + desc + narr
Number of Topics	50

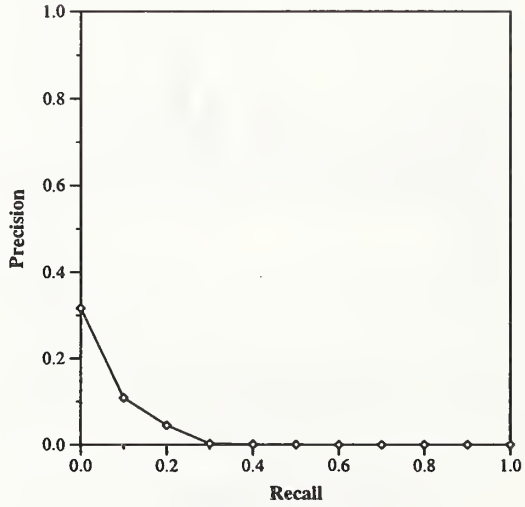
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	413

Recall Level Precision Averages	
Recall	Precision
0.00	0.3168
0.10	0.1090
0.20	0.0452
0.30	0.0028
0.40	0.0010
0.50	0.0005
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000

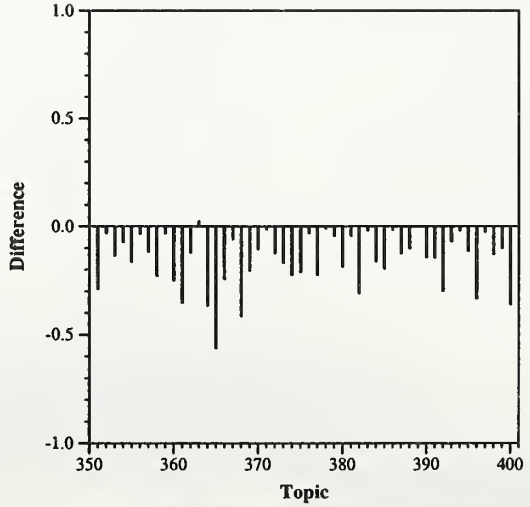
Average precision over all relevant docs	
non-interpolated	0.0281

Document Level Averages	
	Precision
At 5 docs	0.1600
At 10 docs	0.1340
At 15 docs	0.1133
At 20 docs	0.1050
At 30 docs	0.0947
At 100 docs	0.0528
At 200 docs	0.0318
At 500 docs	0.0157
At 1000 docs	0.0083

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0601



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	kslsV1
Run Description	Automatic, title + desc
Number of Topics	50

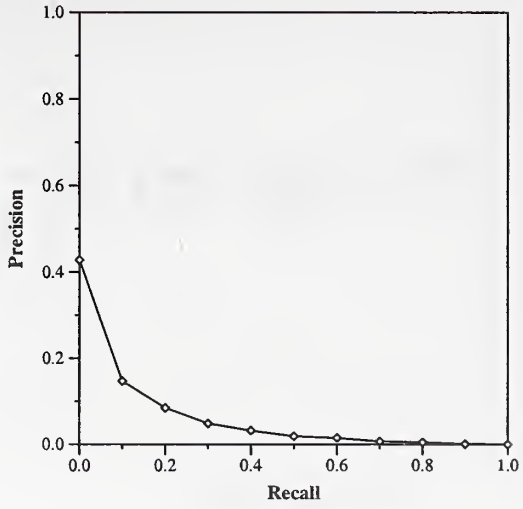
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	1086

Recall Level Precision Averages	
Recall	Precision
0.00	0.4278
0.10	0.1473
0.20	0.0852
0.30	0.0490
0.40	0.0321
0.50	0.0192
0.60	0.0152
0.70	0.0070
0.80	0.0052
0.90	0.0008
1.00	0.0000

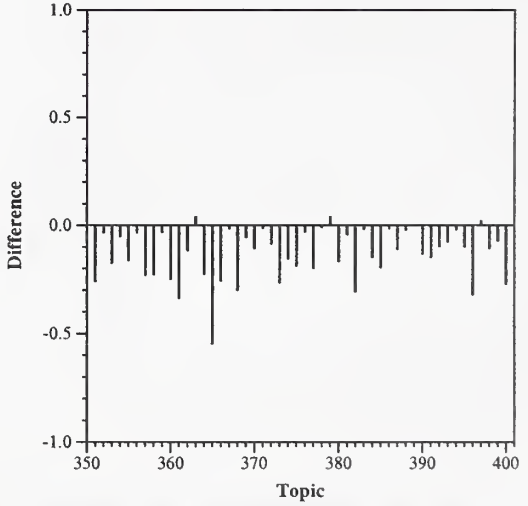
Average precision over all relevant docs	
non-interpolated	0.0499

Document Level Averages	
	Precision
At 5 docs	0.1840
At 10 docs	0.1800
At 15 docs	0.1573
At 20 docs	0.1430
At 30 docs	0.1247
At 100 docs	0.0698
At 200 docs	0.0539
At 500 docs	0.0326
At 1000 docs	0.0217

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0895



Recall-Precision Curve

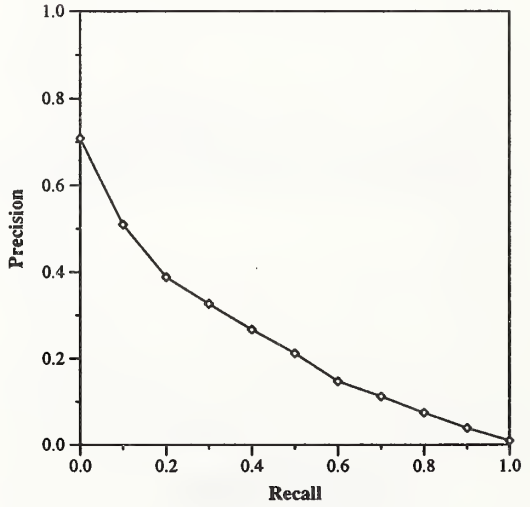


Difference from Median in Average Precision per Topic

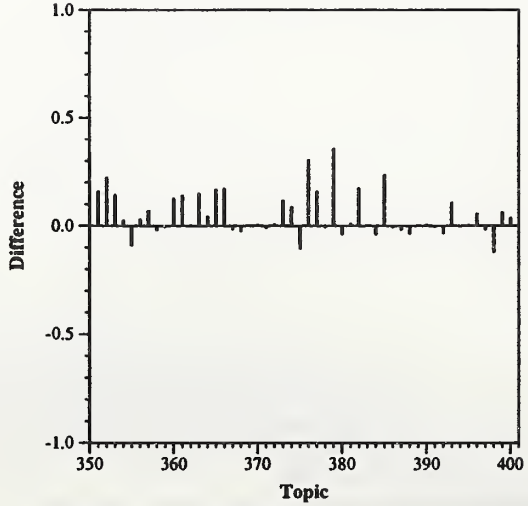
Summary Statistics	
Run Number	LNaTit7
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	48585
Relevant:	4674
Rel-ret:	2791

Recall Level Precision Averages	
Recall	Precision
0.00	0.7087
0.10	0.5099
0.20	0.3887
0.30	0.3269
0.40	0.2673
0.50	0.2125
0.60	0.1475
0.70	0.1126
0.80	0.0744
0.90	0.0392
1.00	0.0097
Average precision over all relevant docs	
non-interpolated	0.2338

Document Level Averages	
	Precision
At 5 docs	0.4880
At 10 docs	0.4620
At 15 docs	0.4173
At 20 docs	0.3910
At 30 docs	0.3480
At 100 docs	0.2098
At 200 docs	0.1519
At 500 docs	0.0914
At 1000 docs	0.0558
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2691



Recall-Precision Curve

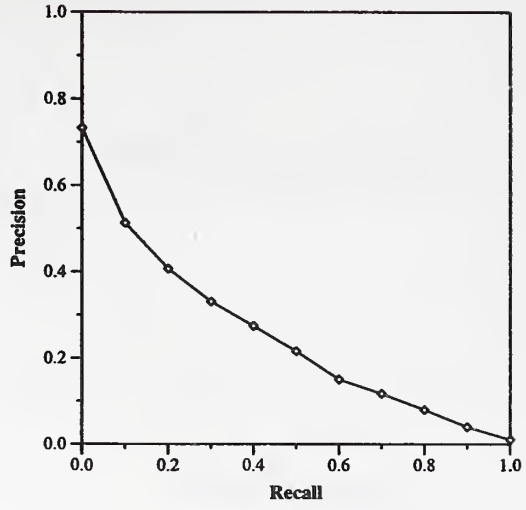


Difference from Median in Average Precision per Topic

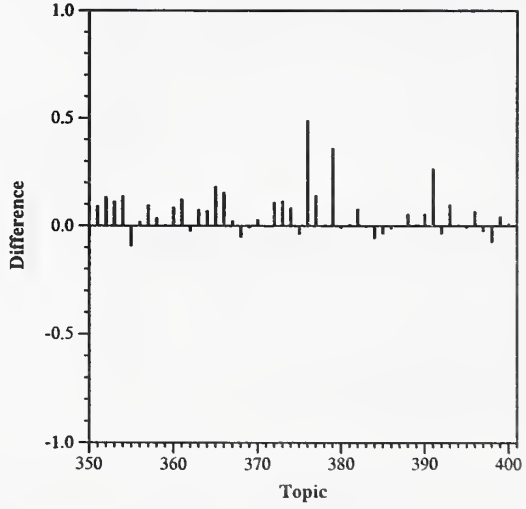
Summary Statistics	
Run Number	LNaTitDesc7
Run Description	Automatic, title + desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	49445
Relevant:	4674
Rel-ret:	3020

Recall Level Precision Averages	
Recall	Precision
0.00	0.7327
0.10	0.5131
0.20	0.4073
0.30	0.3311
0.40	0.2747
0.50	0.2160
0.60	0.1506
0.70	0.1173
0.80	0.0800
0.90	0.0400
1.00	0.0105
Average precision over all relevant docs	
non-interpolated	0.2394

Document Level Averages	
	Precision
At 5 docs	0.5000
At 10 docs	0.4580
At 15 docs	0.4267
At 20 docs	0.4050
At 30 docs	0.3613
At 100 docs	0.2206
At 200 docs	0.1605
At 500 docs	0.0959
At 1000 docs	0.0604
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2783



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	jalbse011
Run Description	Automatic, title + desc
Number of Topics	50

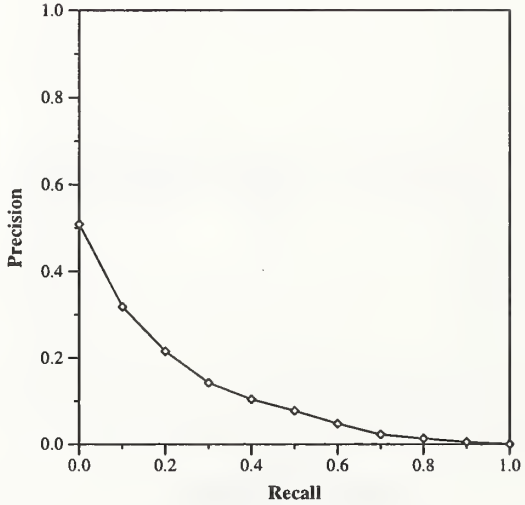
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	1853

Recall Level Precision Averages	
Recall	Precision
0.00	0.5087
0.10	0.3184
0.20	0.2156
0.30	0.1428
0.40	0.1044
0.50	0.0778
0.60	0.0481
0.70	0.0226
0.80	0.0134
0.90	0.0053
1.00	0.0002

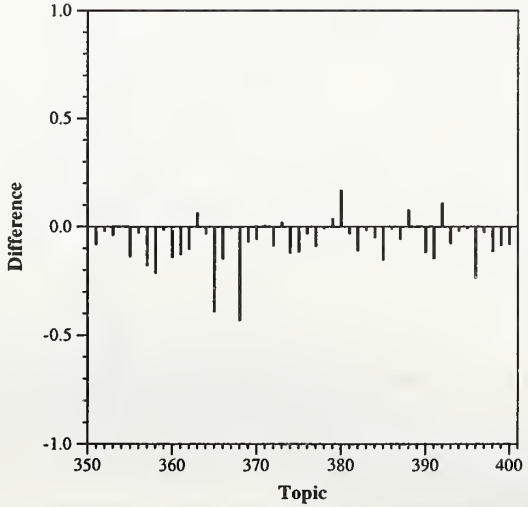
Average precision over all relevant docs	
non-interpolated	0.1119

Document Level Averages	
	Precision
At 5 docs	0.3080
At 10 docs	0.2940
At 15 docs	0.2560
At 20 docs	0.2390
At 30 docs	0.2040
At 100 docs	0.1192
At 200 docs	0.0884
At 500 docs	0.0538
At 1000 docs	0.0371

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1529



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	jalbse012
Run Description	Automatic, title + desc
Number of Topics	50

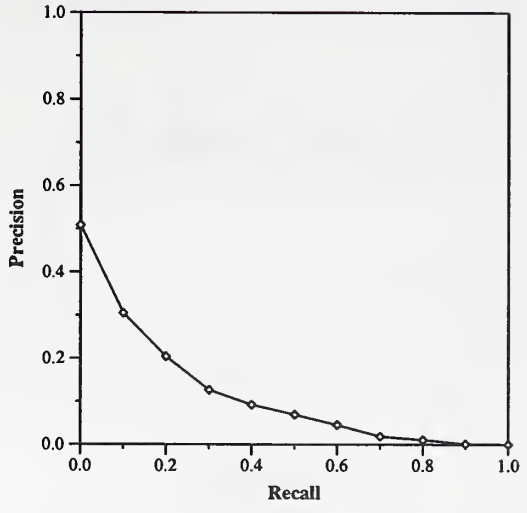
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	1749

Recall Level Precision Averages	
Recall	Precision
0.00	0.5088
0.10	0.3051
0.20	0.2045
0.30	0.1269
0.40	0.0925
0.50	0.0697
0.60	0.0455
0.70	0.0192
0.80	0.0108
0.90	0.0015
1.00	0.0002

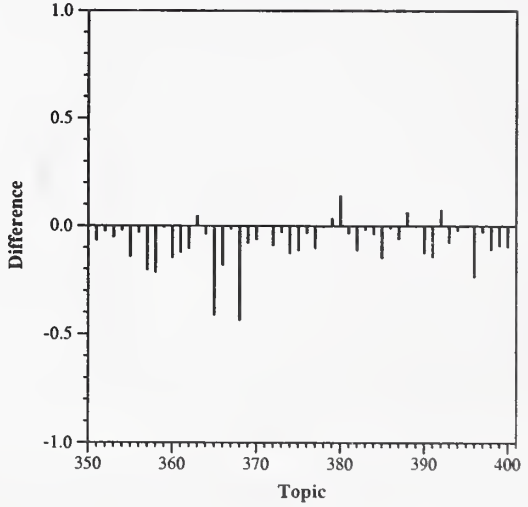
Average precision over all relevant docs	
non-interpolated	0.1063

Document Level Averages	
	Precision
At 5 docs	0.3040
At 10 docs	0.2900
At 15 docs	0.2547
At 20 docs	0.2330
At 30 docs	0.1940
At 100 docs	0.1156
At 200 docs	0.0841
At 500 docs	0.0521
At 1000 docs	0.0350

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1512



Recall-Precision Curve

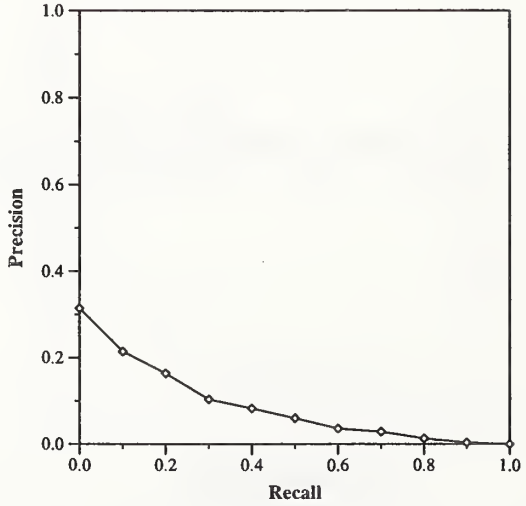


Difference from Median in Average Precision per Topic

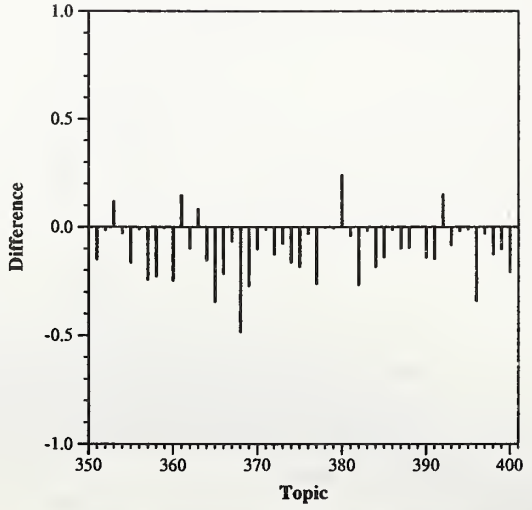
Summary Statistics	
Run Number	jalbse013
Run Description	Automatic, title + desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	1235

Recall Level Precision Averages	
Recall	Precision
0.00	0.3146
0.10	0.2149
0.20	0.1640
0.30	0.1039
0.40	0.0829
0.50	0.0601
0.60	0.0361
0.70	0.0288
0.80	0.0132
0.90	0.0035
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0817

Document Level Averages	
	Precision
At 5 docs	0.1800
At 10 docs	0.1700
At 15 docs	0.1600
At 20 docs	0.1500
At 30 docs	0.1340
At 100 docs	0.0836
At 200 docs	0.0608
At 500 docs	0.0372
At 1000 docs	0.0247
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1176



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	nthu2
Run Description	Automatic, title + desc + narr
Number of Topics	50

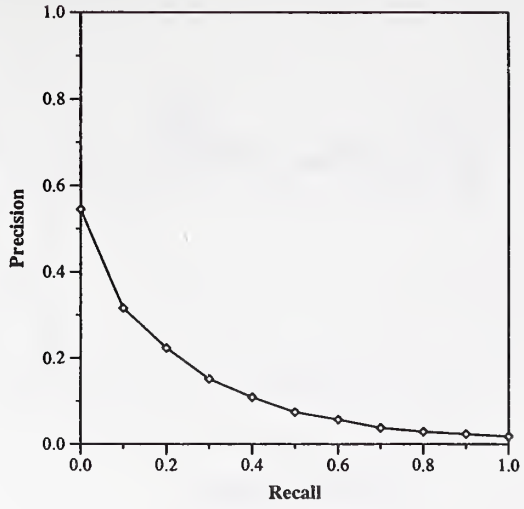
Total number of documents over all topics	
Retrieved:	48054
Relevant:	4674
Rel-ret:	1764

Recall Level Precision Averages	
Recall	Precision
0.00	0.5447
0.10	0.3159
0.20	0.2237
0.30	0.1516
0.40	0.1091
0.50	0.0745
0.60	0.0572
0.70	0.0379
0.80	0.0291
0.90	0.0234
1.00	0.0179

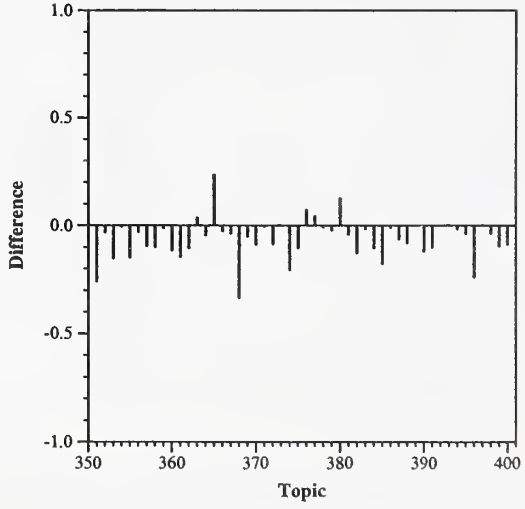
Average precision over all relevant docs	
non-interpolated	0.1206

Document Level Averages	
	Precision
At 5 docs	0.3000
At 10 docs	0.3000
At 15 docs	0.2707
At 20 docs	0.2460
At 30 docs	0.2120
At 100 docs	0.1264
At 200 docs	0.0882
At 500 docs	0.0542
At 1000 docs	0.0353

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1705



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	nthu3
Run Description	Automatic, title + desc + narr
Number of Topics	50

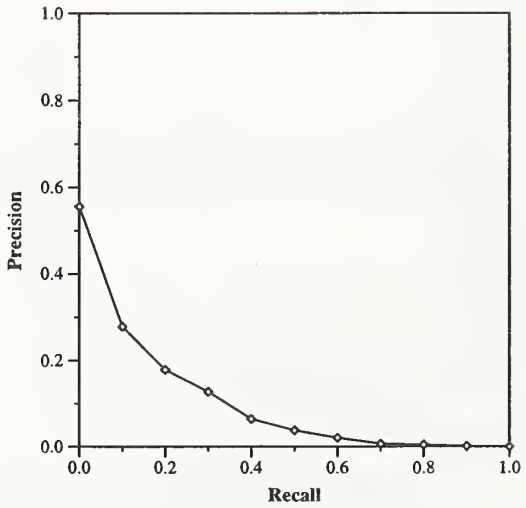
Total number of documents over all topics	
Retrieved:	42871
Relevant:	4674
Rel-ret:	1562

Recall Level Precision Averages	
Recall	Precision
0.00	0.5557
0.10	0.2786
0.20	0.1785
0.30	0.1275
0.40	0.0644
0.50	0.0379
0.60	0.0206
0.70	0.0067
0.80	0.0046
0.90	0.0012
1.00	0.0000

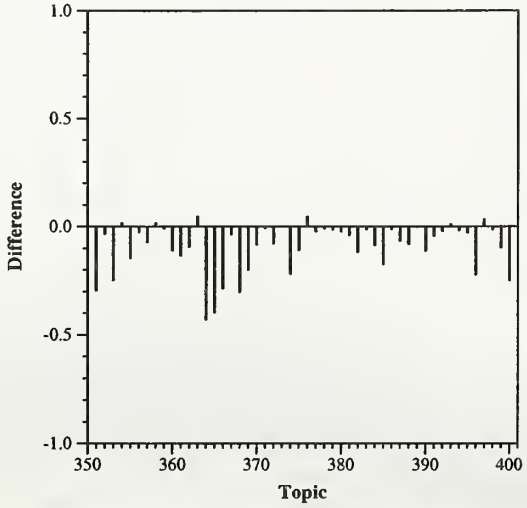
Average precision over all relevant docs	
non-interpolated	0.0901

Document Level Averages	
	Precision
At 5 docs	0.3040
At 10 docs	0.2880
At 15 docs	0.2507
At 20 docs	0.2130
At 30 docs	0.1833
At 100 docs	0.1108
At 200 docs	0.0803
At 500 docs	0.0472
At 1000 docs	0.0312

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1401



Recall-Precision Curve

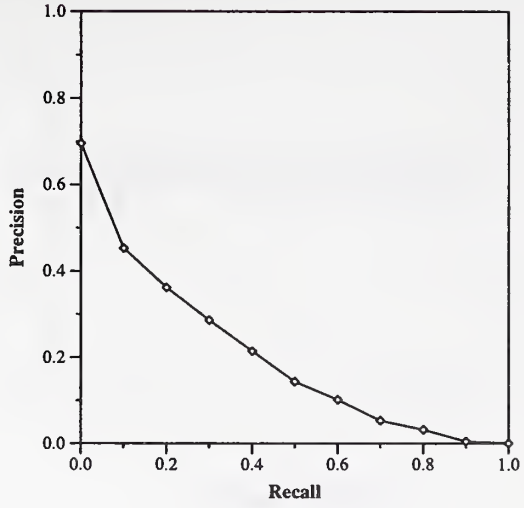


Difference from Median in Average Precision per Topic

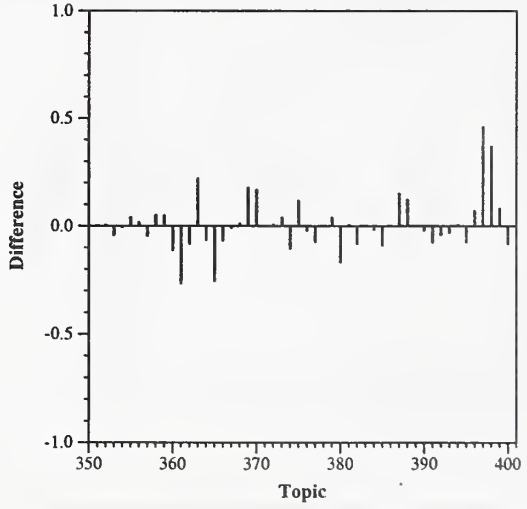
Summary Statistics	
Run Number	nectitech
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2435

Recall Level Precision Averages	
Recall	Precision
0.00	0.6957
0.10	0.4528
0.20	0.3622
0.30	0.2864
0.40	0.2148
0.50	0.1438
0.60	0.1017
0.70	0.0530
0.80	0.0321
0.90	0.0049
1.00	0.0005
Average precision over all relevant docs	
non-interpolated	0.1898

Document Level Averages	
	Precision
At 5 docs	0.4720
At 10 docs	0.4260
At 15 docs	0.4080
At 20 docs	0.3700
At 30 docs	0.3320
At 100 docs	0.2012
At 200 docs	0.1395
At 500 docs	0.0791
At 1000 docs	0.0487
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2403



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	nectitechall
Run Description	Automatic, title + desc + narr
Number of Topics	50

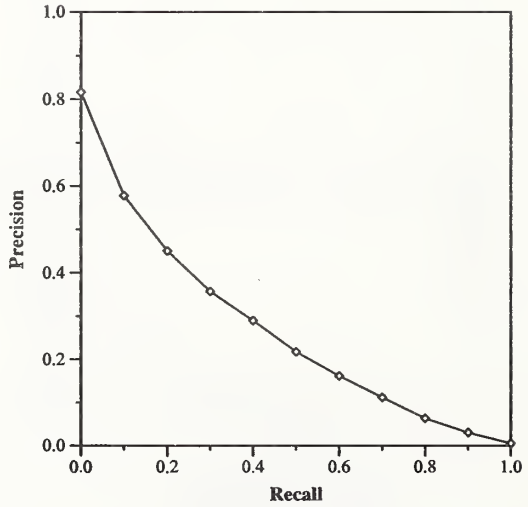
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3106

Recall Level Precision Averages	
Recall	Precision
0.00	0.8161
0.10	0.5783
0.20	0.4511
0.30	0.3575
0.40	0.2899
0.50	0.2177
0.60	0.1618
0.70	0.1121
0.80	0.0636
0.90	0.0306
1.00	0.0054

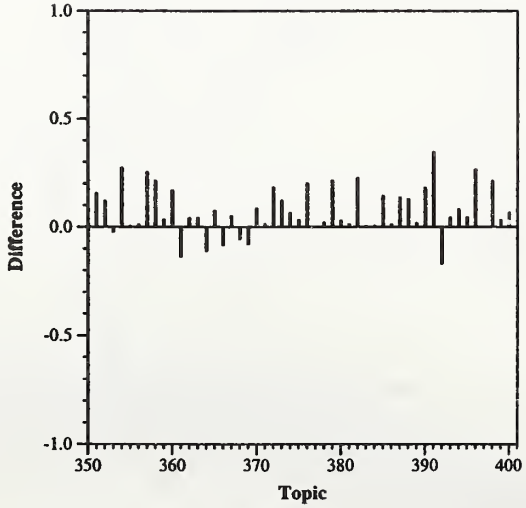
Average precision over all relevant docs	
non-interpolated	0.2565

Document Level Averages	
	Precision
At 5 docs	0.5800
At 10 docs	0.5480
At 15 docs	0.4973
At 20 docs	0.4700
At 30 docs	0.4147
At 100 docs	0.2484
At 200 docs	0.1771
At 500 docs	0.1023
At 1000 docs	0.0621

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2989



Recall-Precision Curve

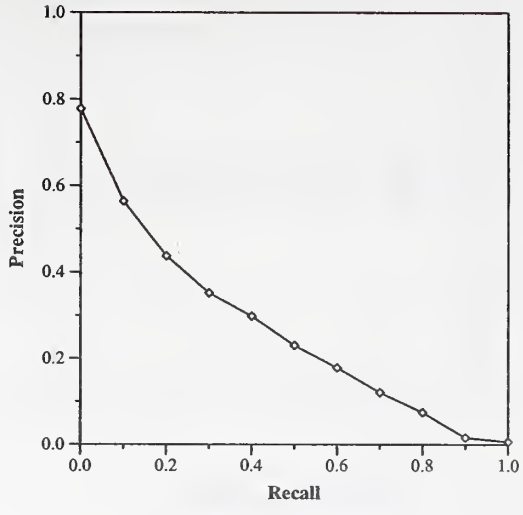


Difference from Median in Average Precision per Topic

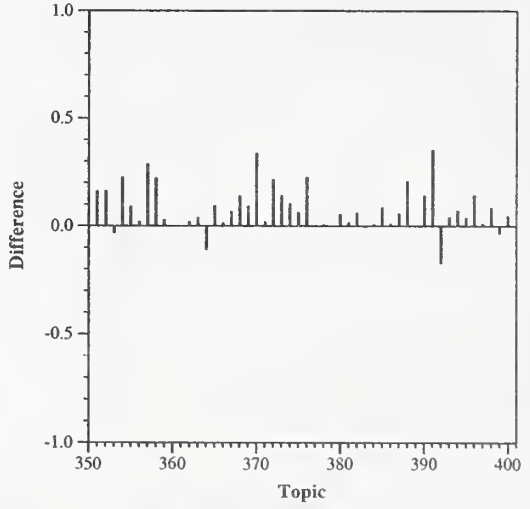
Summary Statistics	
Run Number	nectitechdes
Run Description	Automatic, desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3149

Recall Level Precision Averages	
Recall	Precision
0.00	0.7782
0.10	0.5643
0.20	0.4377
0.30	0.3519
0.40	0.2981
0.50	0.2300
0.60	0.1786
0.70	0.1212
0.80	0.0749
0.90	0.0159
1.00	0.0067
Average precision over all relevant docs	
non-interpolated	0.2584

Document Level Averages	
	Precision
At 5 docs	0.5840
At 10 docs	0.5460
At 15 docs	0.5013
At 20 docs	0.4640
At 30 docs	0.4113
At 100 docs	0.2406
At 200 docs	0.1771
At 500 docs	0.1038
At 1000 docs	0.0630
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2993



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	nttdata7Al0
Run Description	Automatic, title + desc + narr
Number of Topics	50

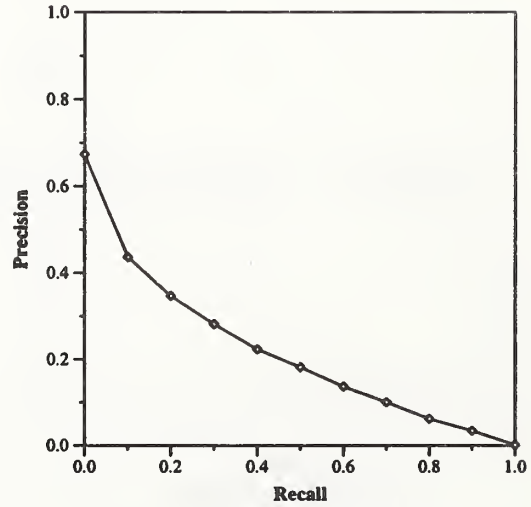
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2624

Recall Level Precision Averages	
Recall	Precision
0.00	0.6738
0.10	0.4366
0.20	0.3469
0.30	0.2816
0.40	0.2233
0.50	0.1818
0.60	0.1368
0.70	0.1007
0.80	0.0620
0.90	0.0345
1.00	0.0013

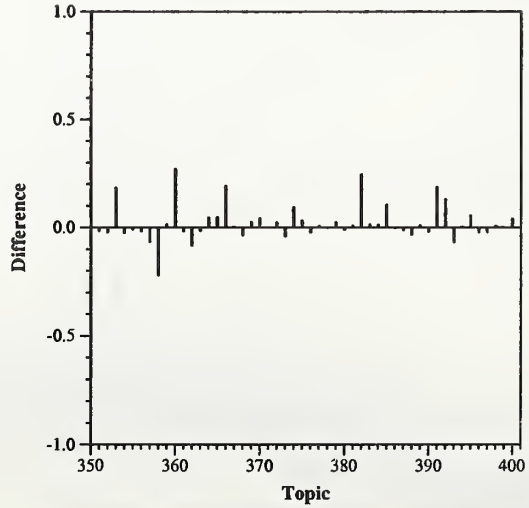
Average precision over all relevant docs	
non-interpolated	0.2032

Document Level Averages	
	Precision
At 5 docs	0.4400
At 10 docs	0.4100
At 15 docs	0.3747
At 20 docs	0.3480
At 30 docs	0.3080
At 100 docs	0.1972
At 200 docs	0.1454
At 500 docs	0.0848
At 1000 docs	0.0525

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2493



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	nttdata7Al2
Run Description	Automatic, title + desc + narr
Number of Topics	50

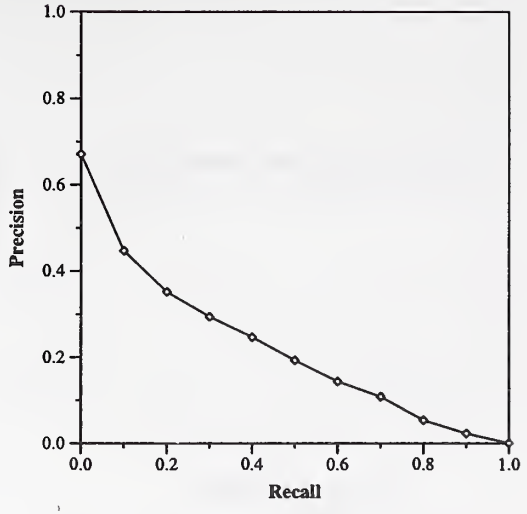
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2656

Recall Level Precision Averages	
Recall	Precision
0.00	0.6715
0.10	0.4476
0.20	0.3523
0.30	0.2947
0.40	0.2472
0.50	0.1934
0.60	0.1443
0.70	0.1090
0.80	0.0541
0.90	0.0231
1.00	0.0003

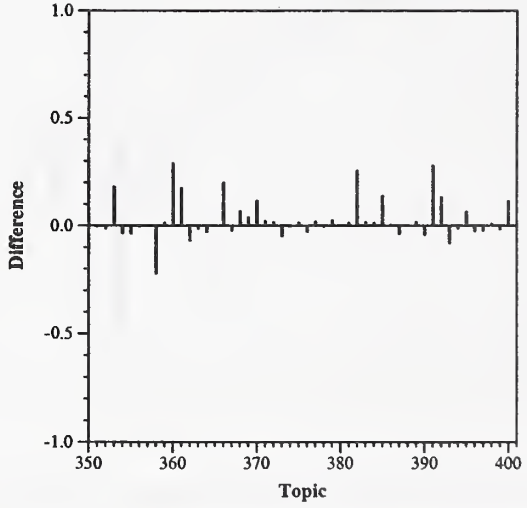
Average precision over all relevant docs	
non-interpolated	0.2113

Document Level Averages	
	Precision
At 5 docs	0.4280
At 10 docs	0.4080
At 15 docs	0.3973
At 20 docs	0.3670
At 30 docs	0.3200
At 100 docs	0.2050
At 200 docs	0.1474
At 500 docs	0.0862
At 1000 docs	0.0531

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2483



Recall-Precision Curve

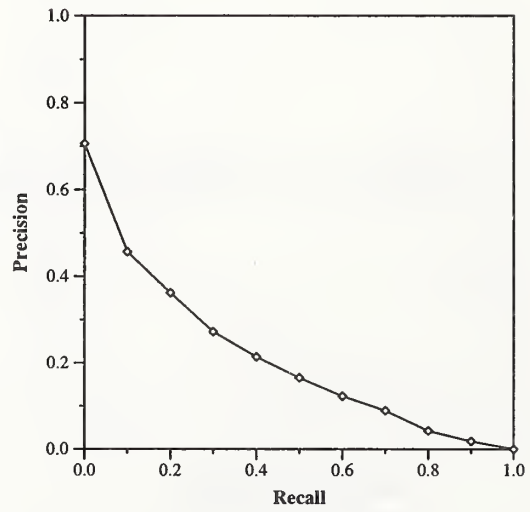


Difference from Median in Average Precision per Topic

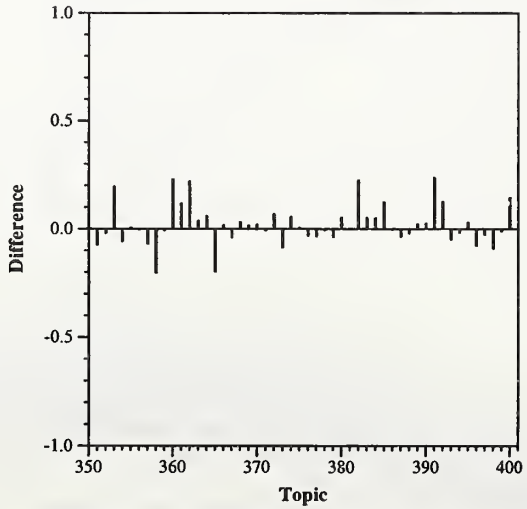
Summary Statistics	
Run Number	nttdata7At1
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2486

Recall Level Precision Averages	
Recall	Precision
0.00	0.7066
0.10	0.4570
0.20	0.3624
0.30	0.2724
0.40	0.2142
0.50	0.1656
0.60	0.1231
0.70	0.0893
0.80	0.0426
0.90	0.0182
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.2017

Document Level Averages	
	Precision
At 5 docs	0.4840
At 10 docs	0.4260
At 15 docs	0.3827
At 20 docs	0.3530
At 30 docs	0.3067
At 100 docs	0.1944
At 200 docs	0.1370
At 500 docs	0.0796
At 1000 docs	0.0497
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2459



Recall-Precision Curve

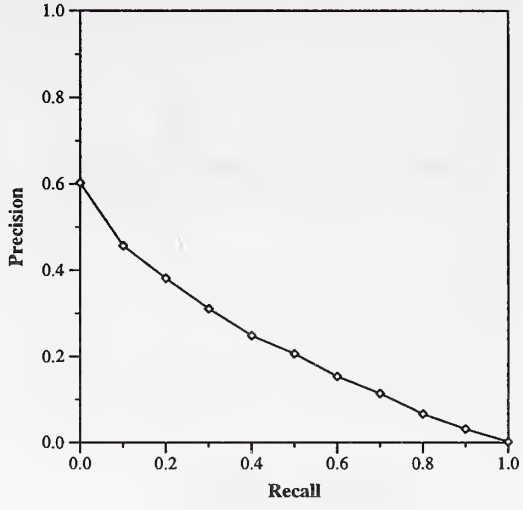


Difference from Median in Average Precision per Topic

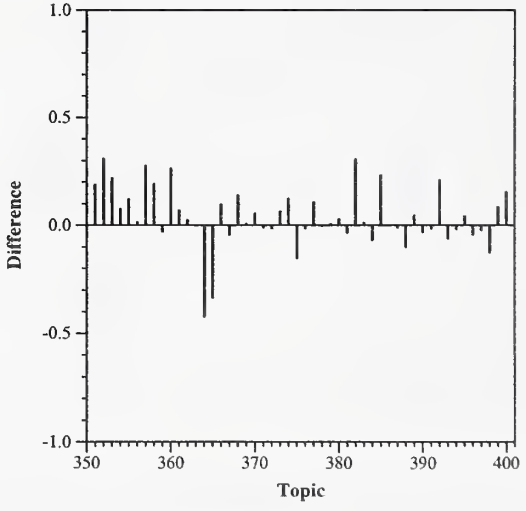
Summary Statistics	
Run Number	mds98t
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2867

Recall Level Precision Averages	
Recall	Precision
0.00	0.6025
0.10	0.4572
0.20	0.3818
0.30	0.3112
0.40	0.2487
0.50	0.2067
0.60	0.1540
0.70	0.1143
0.80	0.0668
0.90	0.0317
1.00	0.0021
Average precision over all relevant docs	
non-interpolated	0.2200

Document Level Averages	
	Precision
At 5 docs	0.4360
At 10 docs	0.4220
At 15 docs	0.3893
At 20 docs	0.3650
At 30 docs	0.3293
At 100 docs	0.2186
At 200 docs	0.1585
At 500 docs	0.0922
At 1000 docs	0.0573
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2564



Recall-Precision Curve

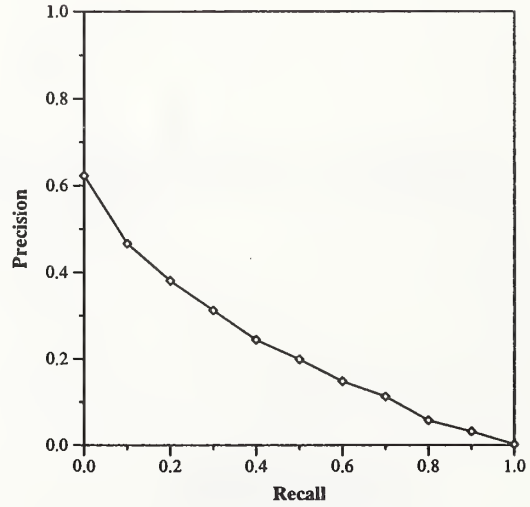


Difference from Median in Average Precision per Topic

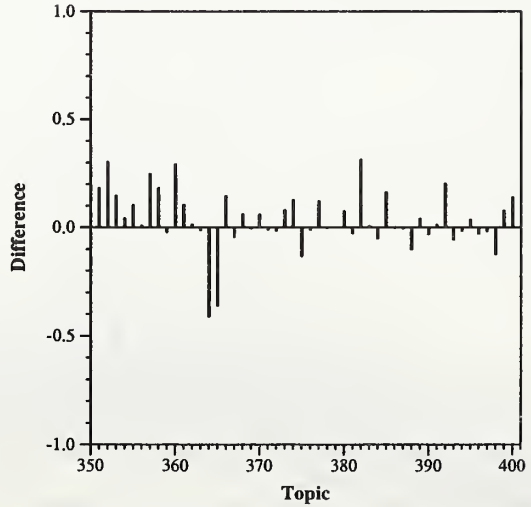
Summary Statistics	
Run Number	mds98t2
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2868

Recall Level Precision Averages	
Recall	Precision
0.00	0.6227
0.10	0.4666
0.20	0.3809
0.30	0.3125
0.40	0.2441
0.50	0.1991
0.60	0.1484
0.70	0.1132
0.80	0.0577
0.90	0.0320
1.00	0.0020
Average precision over all relevant docs	
non-interpolated	0.2181

Document Level Averages	
	Precision
At 5 docs	0.4440
At 10 docs	0.4260
At 15 docs	0.3947
At 20 docs	0.3590
At 30 docs	0.3227
At 100 docs	0.2162
At 200 docs	0.1594
At 500 docs	0.0896
At 1000 docs	0.0574
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2527



Recall-Precision Curve

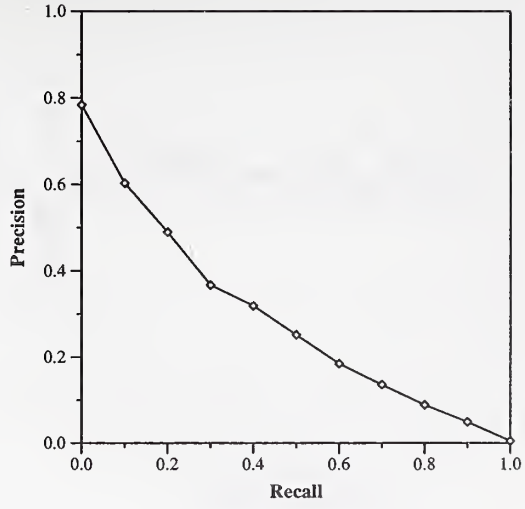


Difference from Median in Average Precision per Topic

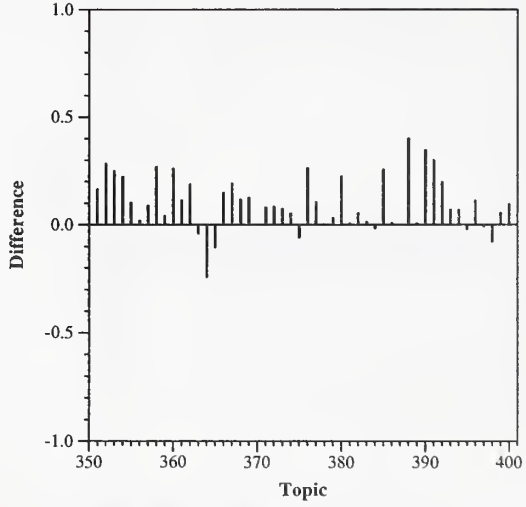
Summary Statistics	
Run Number	mds98td
Run Description	Automatic, title + desc
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3138

Recall Level Precision Averages	
Recall	Precision
0.00	0.7844
0.10	0.6032
0.20	0.4898
0.30	0.3674
0.40	0.3193
0.50	0.2519
0.60	0.1850
0.70	0.1359
0.80	0.0891
0.90	0.0494
1.00	0.0051
Average precision over all relevant docs	
non-interpolated	0.2809

Document Level Averages	
	Precision
At 5 docs	0.5720
At 10 docs	0.5260
At 15 docs	0.4907
At 20 docs	0.4460
At 30 docs	0.3947
At 100 docs	0.2610
At 200 docs	0.1867
At 500 docs	0.1054
At 1000 docs	0.0628
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3074



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	AntHoc01
Run Description	Automatic, title + desc + narr
Number of Topics	50

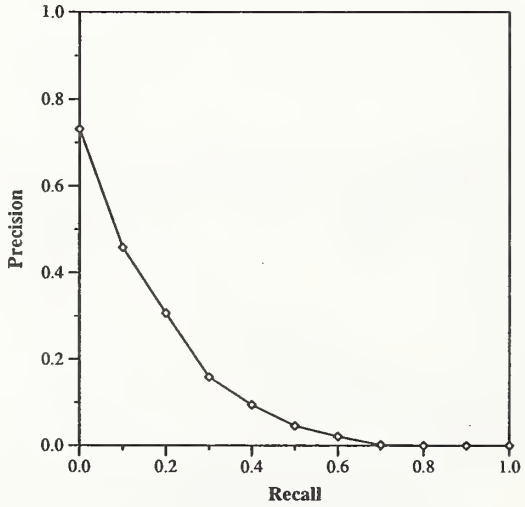
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	809

Recall Level Precision Averages	
Recall	Precision
0.00	0.7316
0.10	0.4584
0.20	0.3068
0.30	0.1592
0.40	0.0946
0.50	0.0459
0.60	0.0211
0.70	0.0016
0.80	0.0000
0.90	0.0000
1.00	0.0000

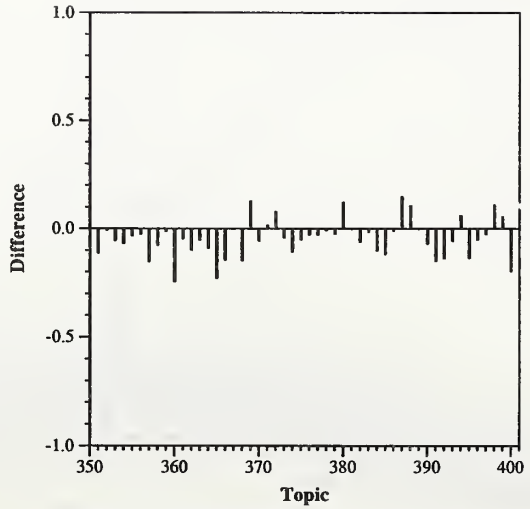
Average precision over all relevant docs	
non-interpolated	0.1375

Document Level Averages	
	Precision
At 5 docs	0.5200
At 10 docs	0.4680
At 15 docs	0.4173
At 20 docs	0.3780
At 30 docs	0.3087
At 100 docs	0.1488
At 200 docs	0.0767
At 500 docs	0.0318
At 1000 docs	0.0162

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1997



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	ScaiTrec7
Run Description	Automatic, title + desc + narr
Number of Topics	50

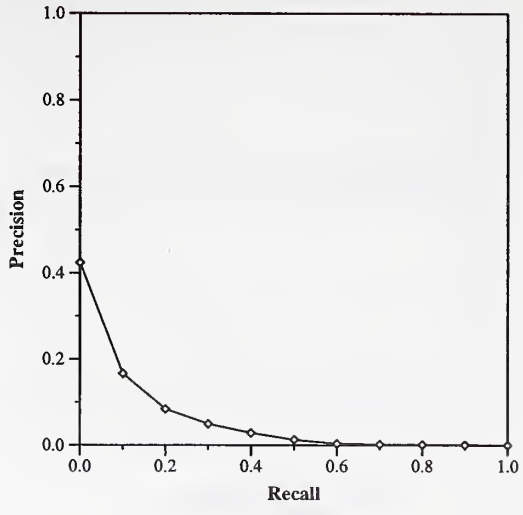
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	1255

Recall Level Precision Averages	
Recall	Precision
0.00	0.4239
0.10	0.1669
0.20	0.0844
0.30	0.0497
0.40	0.0289
0.50	0.0129
0.60	0.0037
0.70	0.0020
0.80	0.0012
0.90	0.0006
1.00	0.0000

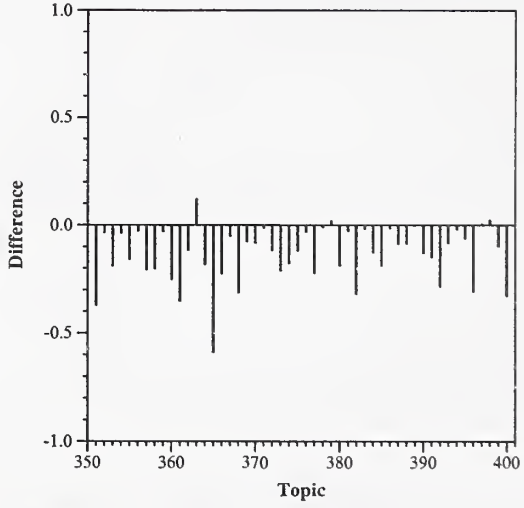
Average precision over all relevant docs	
non-interpolated	0.0477

Document Level Averages	
	Precision
At 5 docs	0.2000
At 10 docs	0.1500
At 15 docs	0.1320
At 20 docs	0.1240
At 30 docs	0.1100
At 100 docs	0.0672
At 200 docs	0.0539
At 500 docs	0.0350
At 1000 docs	0.0251

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0794



Recall-Precision Curve

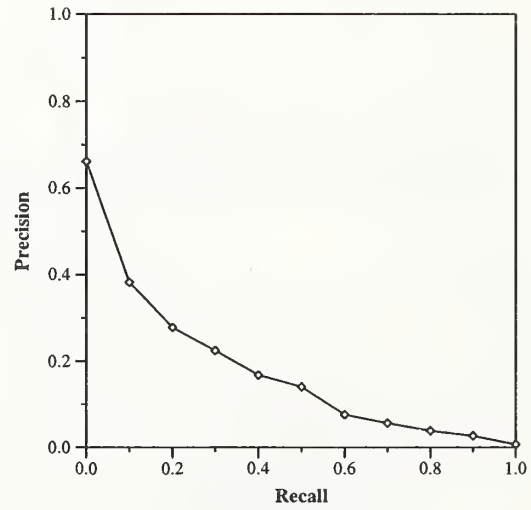


Difference from Median in Average Precision per Topic

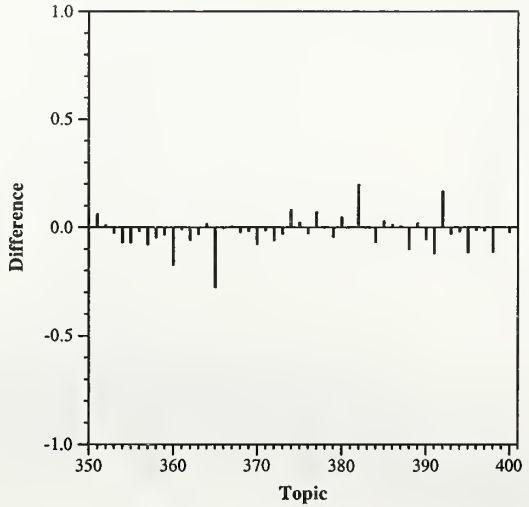
Summary Statistics	
Run Number	ETHAB0
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	46493
Relevant:	4674
Rel-ret:	1767

Recall Level Precision Averages	
Recall	Precision
0.00	0.6617
0.10	0.3829
0.20	0.2782
0.30	0.2248
0.40	0.1681
0.50	0.1409
0.60	0.0762
0.70	0.0566
0.80	0.0388
0.90	0.0270
1.00	0.0069
Average precision over all relevant docs	
non-interpolated	0.1601

Document Level Averages	
	Precision
At 5 docs	0.4280
At 10 docs	0.4100
At 15 docs	0.3653
At 20 docs	0.3330
At 30 docs	0.2793
At 100 docs	0.1550
At 200 docs	0.1094
At 500 docs	0.0609
At 1000 docs	0.0353
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2046



Recall-Precision Curve

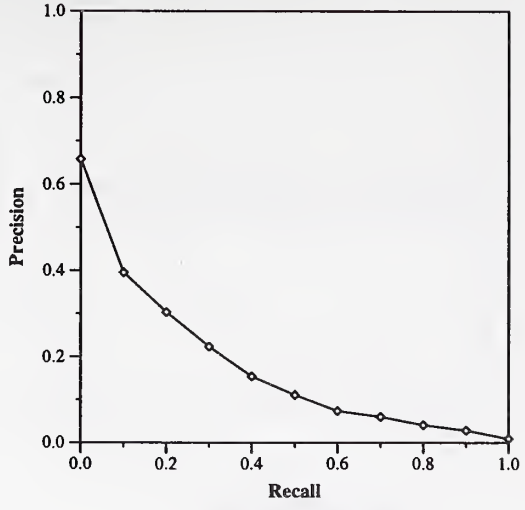


Difference from Median in Average Precision per Topic

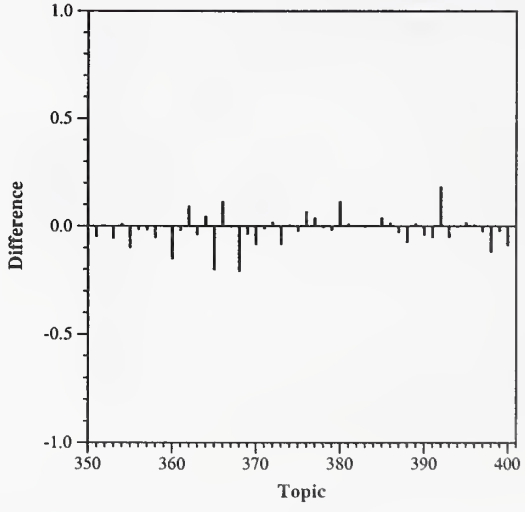
Summary Statistics	
Run Number	ETHAC0
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2236

Recall Level Precision Averages	
Recall	Precision
0.00	0.6585
0.10	0.3957
0.20	0.3037
0.30	0.2237
0.40	0.1544
0.50	0.1112
0.60	0.0740
0.70	0.0604
0.80	0.0413
0.90	0.0288
1.00	0.0092
Average precision over all relevant docs	
non-interpolated	0.1645

Document Level Averages	
	Precision
At 5 docs	0.4480
At 10 docs	0.4000
At 15 docs	0.3533
At 20 docs	0.3280
At 30 docs	0.2693
At 100 docs	0.1650
At 200 docs	0.1162
At 500 docs	0.0712
At 1000 docs	0.0447
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2102



Recall-Precision Curve

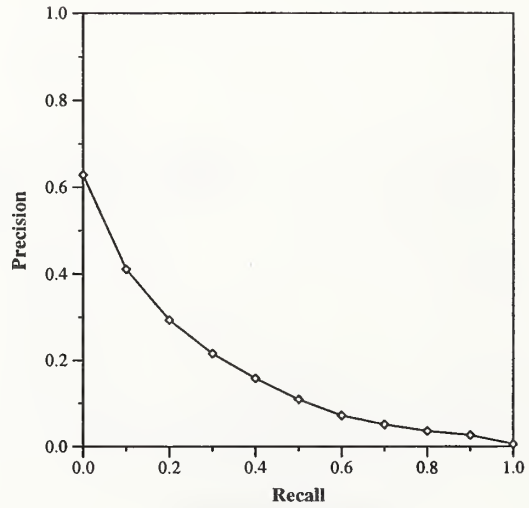


Difference from Median in Average Precision per Topic

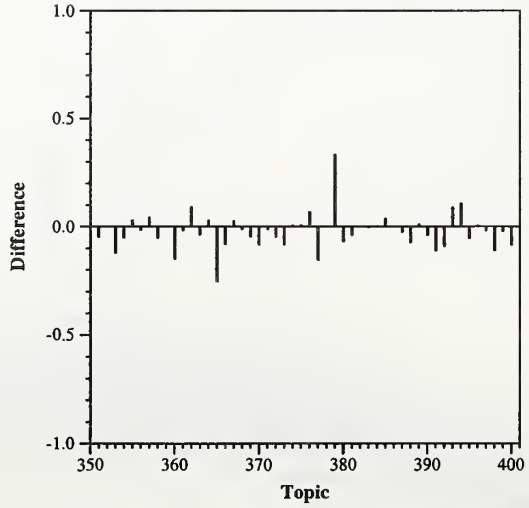
Summary Statistics	
Run Number	ETHAR0
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	46493
Relevant:	4674
Rel-ret:	2142

Recall Level Precision Averages	
Recall	Precision
0.00	0.6286
0.10	0.4111
0.20	0.2938
0.30	0.2160
0.40	0.1585
0.50	0.1098
0.60	0.0721
0.70	0.0514
0.80	0.0362
0.90	0.0267
1.00	0.0058
Average precision over all relevant docs	
non-interpolated	0.1597

Document Level Averages	
	Precision
At 5 docs	0.4400
At 10 docs	0.3780
At 15 docs	0.3333
At 20 docs	0.3030
At 30 docs	0.2680
At 100 docs	0.1554
At 200 docs	0.1075
At 500 docs	0.0677
At 1000 docs	0.0428
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1986



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	tno7cbm25
Run Description	Automatic, title + desc + narr
Number of Topics	50

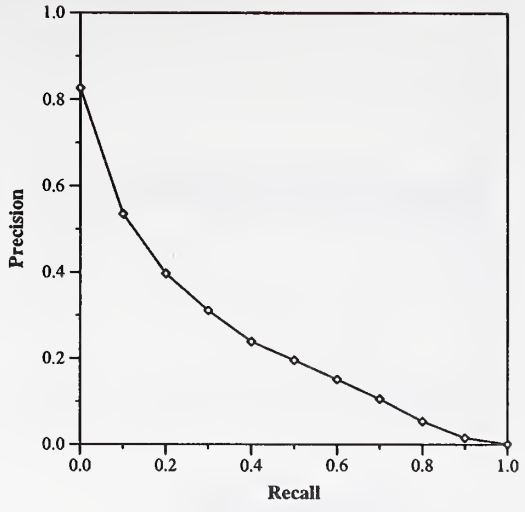
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2877

Recall Level Precision Averages	
Recall	Precision
0.00	0.8267
0.10	0.5351
0.20	0.3972
0.30	0.3116
0.40	0.2393
0.50	0.1961
0.60	0.1515
0.70	0.1061
0.80	0.0540
0.90	0.0156
1.00	0.0009

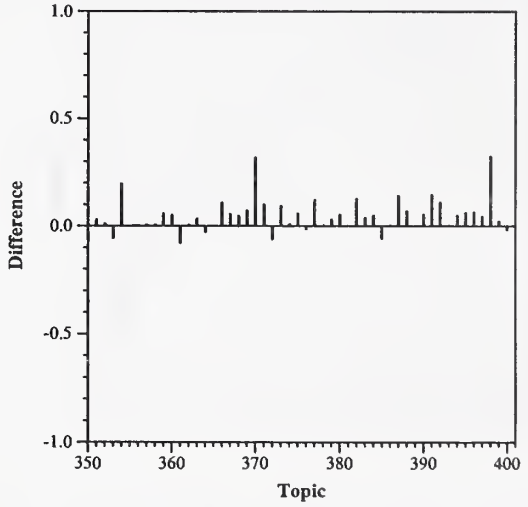
Average precision over all relevant docs	
non-interpolated	0.2315

Document Level Averages	
	Precision
At 5 docs	0.5760
At 10 docs	0.5020
At 15 docs	0.4667
At 20 docs	0.4150
At 30 docs	0.3773
At 100 docs	0.2302
At 200 docs	0.1646
At 500 docs	0.0951
At 1000 docs	0.0575

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2705



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	tno7exp1
Run Description	Automatic, title + desc + narr
Number of Topics	50

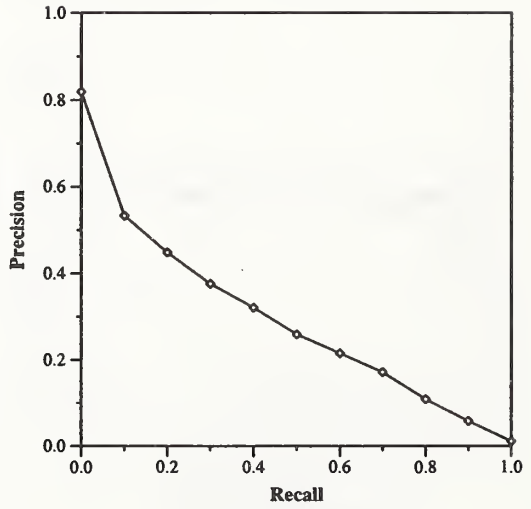
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3324

Recall Level Precision Averages	
Recall	Precision
0.00	0.8191
0.10	0.5335
0.20	0.4487
0.30	0.3766
0.40	0.3214
0.50	0.2592
0.60	0.2156
0.70	0.1716
0.80	0.1085
0.90	0.0582
1.00	0.0113

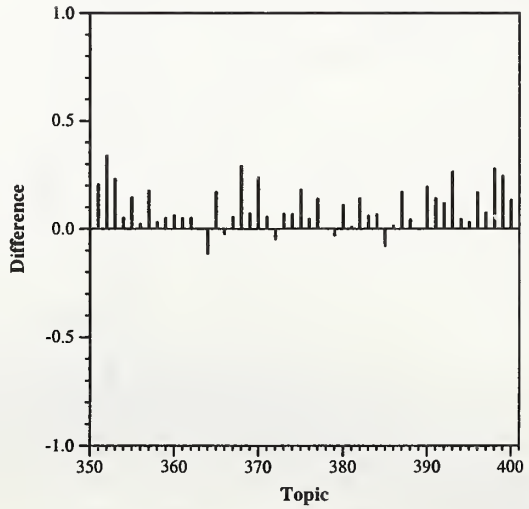
Average precision over all relevant docs	
non-interpolated	0.2785

Document Level Averages	
	Precision
At 5 docs	0.5440
At 10 docs	0.4920
At 15 docs	0.4587
At 20 docs	0.4330
At 30 docs	0.3960
At 100 docs	0.2606
At 200 docs	0.1878
At 500 docs	0.1109
At 1000 docs	0.0665

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2976



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	tno7tw4
Run Description	Automatic, title + desc + narr
Number of Topics	50

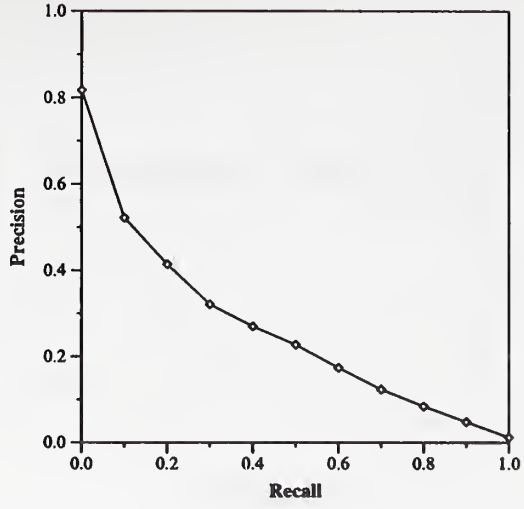
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3007

Recall Level Precision Averages	
Recall	Precision
0.00	0.8178
0.10	0.5219
0.20	0.4144
0.30	0.3214
0.40	0.2704
0.50	0.2277
0.60	0.1741
0.70	0.1238
0.80	0.0844
0.90	0.0482
1.00	0.0119

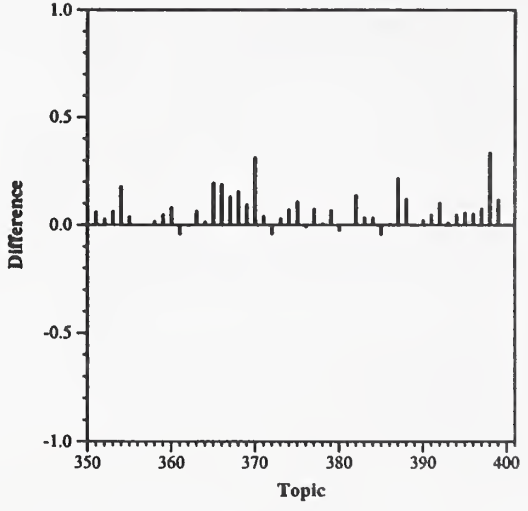
Average precision over all relevant docs	
non-interpolated	0.2490

Document Level Averages	
	Precision
At 5 docs	0.5560
At 10 docs	0.5140
At 15 docs	0.4760
At 20 docs	0.4410
At 30 docs	0.3873
At 100 docs	0.2450
At 200 docs	0.1718
At 500 docs	0.0978
At 1000 docs	0.0601

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2873



Recall-Precision Curve



Difference from Median in Average Precision per Topic

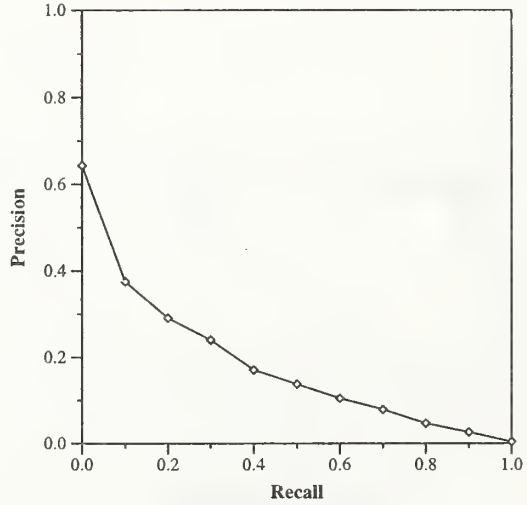
Summary Statistics	
Run Number	Brkly24
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2220

Recall Level Precision Averages	
Recall	Precision
0.00	0.6432
0.10	0.3757
0.20	0.2911
0.30	0.2403
0.40	0.1711
0.50	0.1380
0.60	0.1052
0.70	0.0793
0.80	0.0469
0.90	0.0265
1.00	0.0041

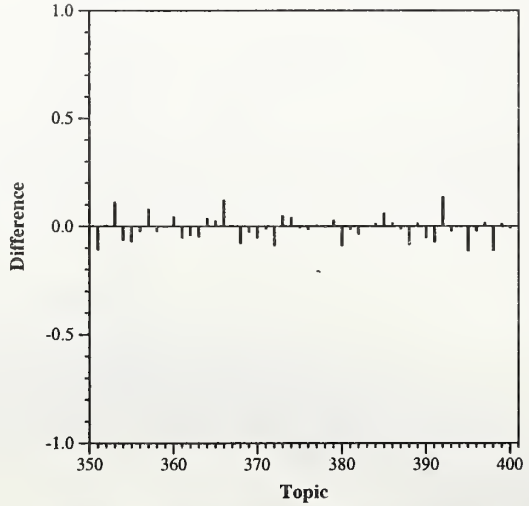
Average precision over all relevant docs	
non-interpolated	0.1714

Document Level Averages	
	Precision
At 5 docs	0.4040
At 10 docs	0.3900
At 15 docs	0.3667
At 20 docs	0.3290
At 30 docs	0.2780
At 100 docs	0.1786
At 200 docs	0.1251
At 500 docs	0.0734
At 1000 docs	0.0444

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2165



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	Brkly25
Run Description	Automatic, title + desc + narr
Number of Topics	50

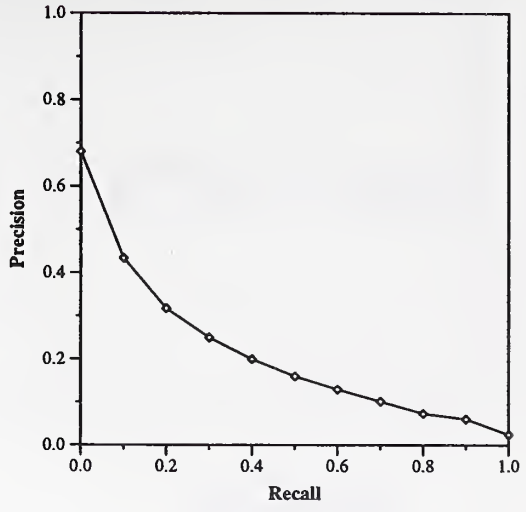
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2493

Recall Level Precision Averages	
Recall	Precision
0.00	0.6803
0.10	0.4340
0.20	0.3174
0.30	0.2499
0.40	0.1996
0.50	0.1598
0.60	0.1296
0.70	0.1012
0.80	0.0733
0.90	0.0607
1.00	0.0246

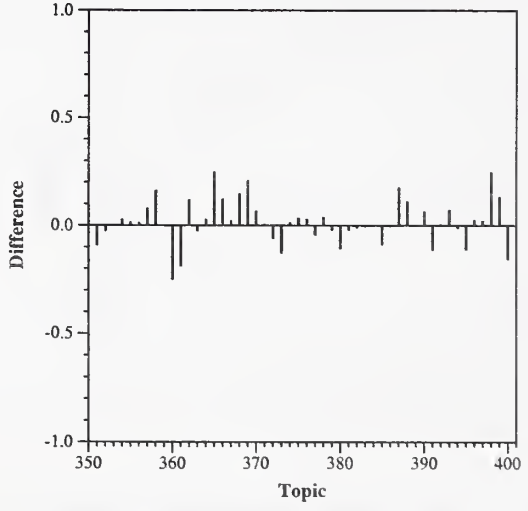
Average precision over all relevant docs	
non-interpolated	0.1973

Document Level Averages	
	Precision
At 5 docs	0.4680
At 10 docs	0.4180
At 15 docs	0.3893
At 20 docs	0.3530
At 30 docs	0.3160
At 100 docs	0.1966
At 200 docs	0.1400
At 500 docs	0.0813
At 1000 docs	0.0499

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2328



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	iowacuhk1
Run Description	Automatic, title + desc + narr
Number of Topics	50

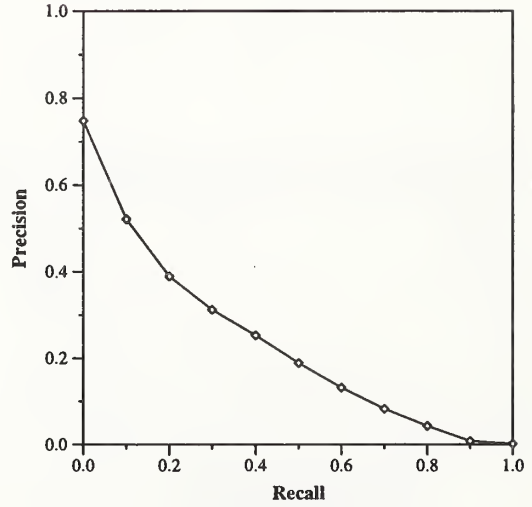
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2649

Recall Level Precision Averages	
Recall	Precision
0.00	0.7485
0.10	0.5216
0.20	0.3895
0.30	0.3126
0.40	0.2530
0.50	0.1890
0.60	0.1320
0.70	0.0828
0.80	0.0431
0.90	0.0079
1.00	0.0013

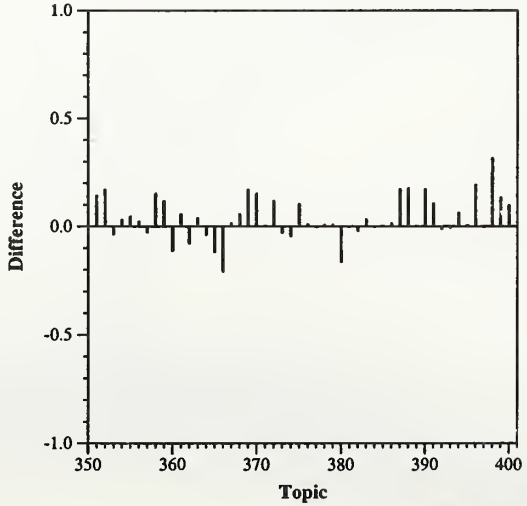
Average precision over all relevant docs	
non-interpolated	0.2221

Document Level Averages	
	Precision
At 5 docs	0.5440
At 10 docs	0.5020
At 15 docs	0.4627
At 20 docs	0.4310
At 30 docs	0.3747
At 100 docs	0.2362
At 200 docs	0.1600
At 500 docs	0.0872
At 1000 docs	0.0530

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2680



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	iowacuhk2
Run Description	Automatic, title + desc + narr
Number of Topics	50

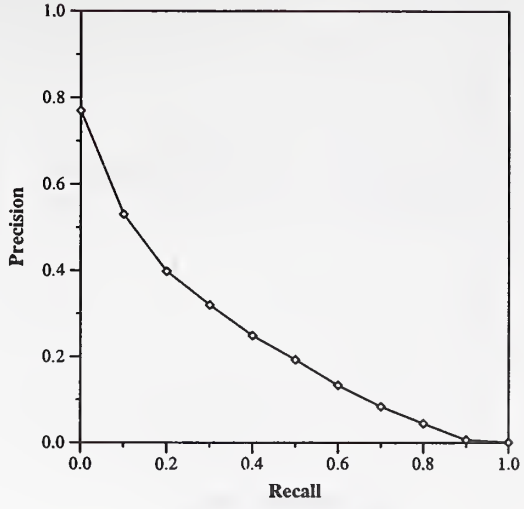
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2674

Recall Level Precision Averages	
Recall	Precision
0.00	0.7703
0.10	0.5304
0.20	0.3985
0.30	0.3207
0.40	0.2493
0.50	0.1932
0.60	0.1338
0.70	0.0842
0.80	0.0449
0.90	0.0069
1.00	0.0011

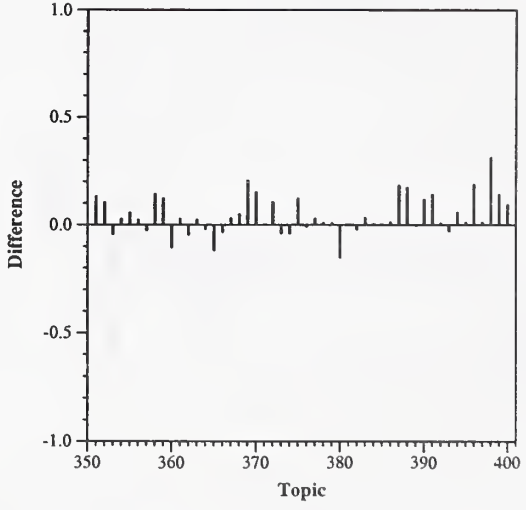
Average precision over all relevant docs	
non-interpolated	0.2260

Document Level Averages	
	Precision
At 5 docs	0.5600
At 10 docs	0.4860
At 15 docs	0.4613
At 20 docs	0.4320
At 30 docs	0.3813
At 100 docs	0.2386
At 200 docs	0.1613
At 500 docs	0.0878
At 1000 docs	0.0535

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2754



Recall-Precision Curve

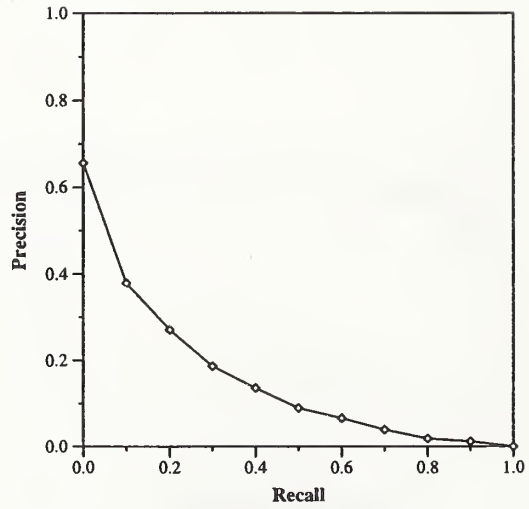


Difference from Median in Average Precision per Topic

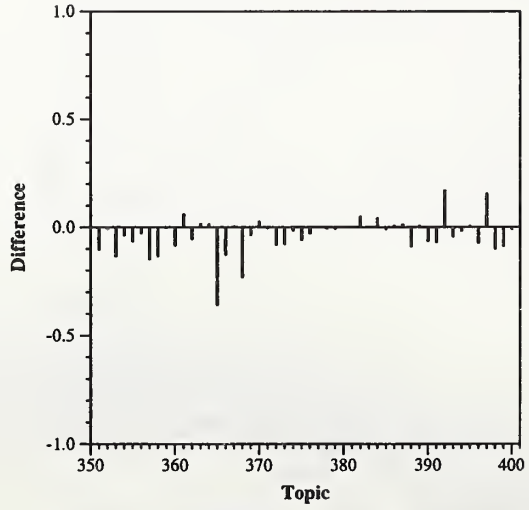
Summary Statistics	
Run Number	umd98a1
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	47396
Relevant:	4674
Rel-ret:	2106

Recall Level Precision Averages	
Recall	Precision
0.00	0.6563
0.10	0.3789
0.20	0.2712
0.30	0.1867
0.40	0.1359
0.50	0.0894
0.60	0.0660
0.70	0.0393
0.80	0.0183
0.90	0.0118
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.1456

Document Level Averages	
	Precision
At 5 docs	0.4040
At 10 docs	0.3560
At 15 docs	0.3413
At 20 docs	0.3120
At 30 docs	0.2567
At 100 docs	0.1562
At 200 docs	0.1096
At 500 docs	0.0670
At 1000 docs	0.0421
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1987



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	umd98a2
Run Description	Automatic, title + desc + narr
Number of Topics	50

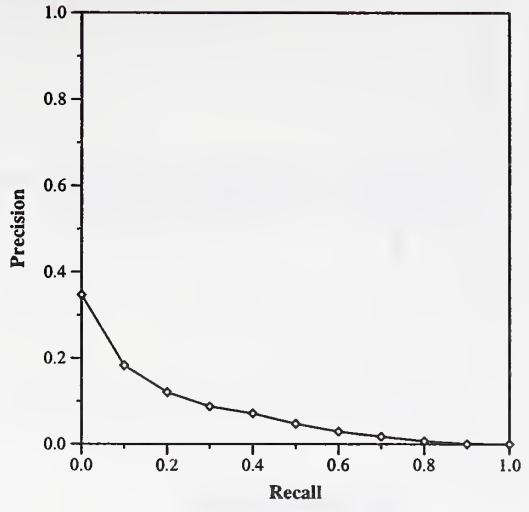
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	1754

Recall Level Precision Averages	
Recall	Precision
0.00	0.3469
0.10	0.1831
0.20	0.1210
0.30	0.0879
0.40	0.0714
0.50	0.0475
0.60	0.0293
0.70	0.0176
0.80	0.0071
0.90	0.0007
1.00	0.0000

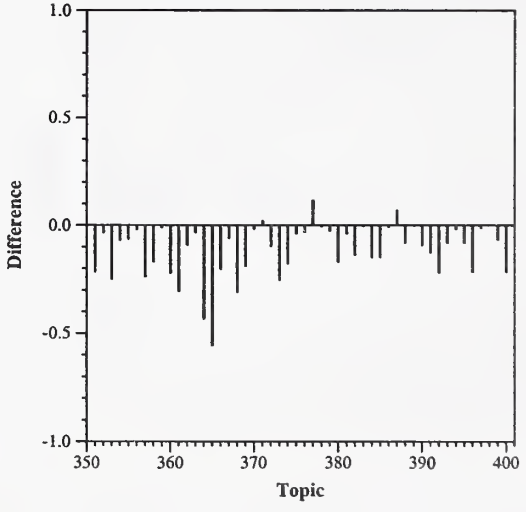
Average precision over all relevant docs	
non-interpolated	0.0668

Document Level Averages	
	Precision
At 5 docs	0.1680
At 10 docs	0.1520
At 15 docs	0.1493
At 20 docs	0.1500
At 30 docs	0.1333
At 100 docs	0.0940
At 200 docs	0.0765
At 500 docs	0.0509
At 1000 docs	0.0351

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1157



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	INQ501
Run Description	Automatic, title + desc + narr
Number of Topics	50

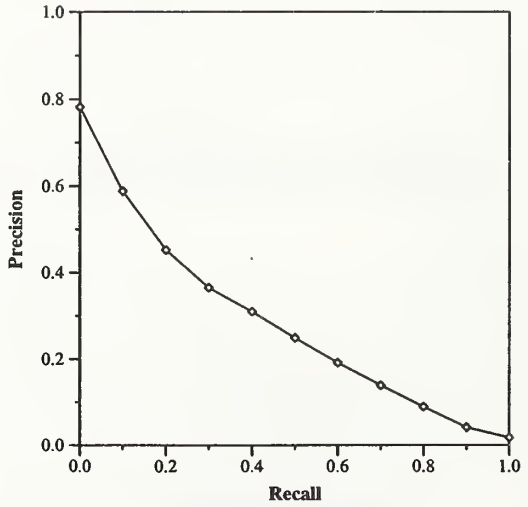
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3206

Recall Level Precision Averages	
Recall	Precision
0.00	0.7823
0.10	0.5883
0.20	0.4526
0.30	0.3656
0.40	0.3101
0.50	0.2490
0.60	0.1919
0.70	0.1393
0.80	0.0893
0.90	0.0417
1.00	0.0175

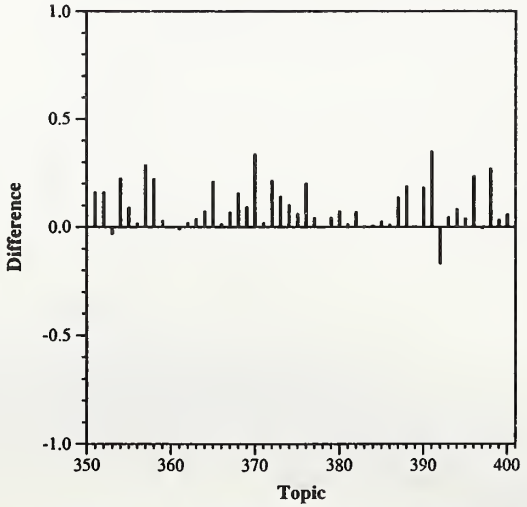
Average precision over all relevant docs	
non-interpolated	0.2739

Document Level Averages	
	Precision
At 5 docs	0.5760
At 10 docs	0.5540
At 15 docs	0.5240
At 20 docs	0.4850
At 30 docs	0.4240
At 100 docs	0.2502
At 200 docs	0.1833
At 500 docs	0.1063
At 1000 docs	0.0641

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3117



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	INQ502
Run Description	Automatic, title + desc + narr
Number of Topics	50

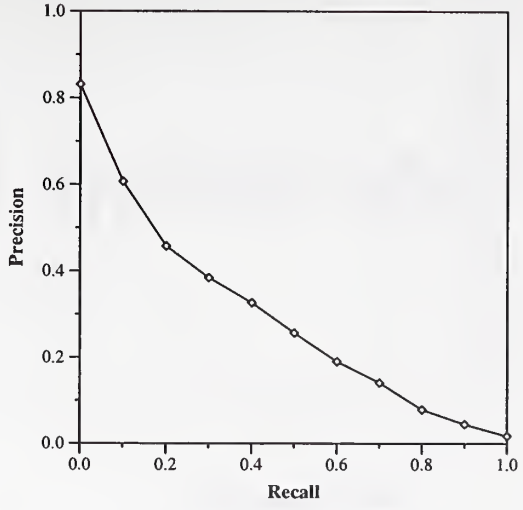
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3215

Recall Level Precision Averages	
Recall	Precision
0.00	0.8314
0.10	0.6070
0.20	0.4579
0.30	0.3848
0.40	0.3269
0.50	0.2571
0.60	0.1906
0.70	0.1414
0.80	0.0790
0.90	0.0449
1.00	0.0175

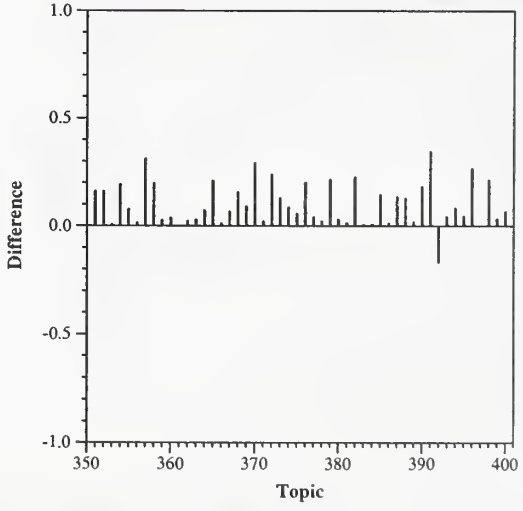
Average precision over all relevant docs	
non-interpolated	0.2815

Document Level Averages	
	Precision
At 5 docs	0.6160
At 10 docs	0.5800
At 15 docs	0.5333
At 20 docs	0.4950
At 30 docs	0.4347
At 100 docs	0.2572
At 200 docs	0.1840
At 500 docs	0.1070
At 1000 docs	0.0643

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3178



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	INQ503
Run Description	Automatic, title + desc
Number of Topics	50

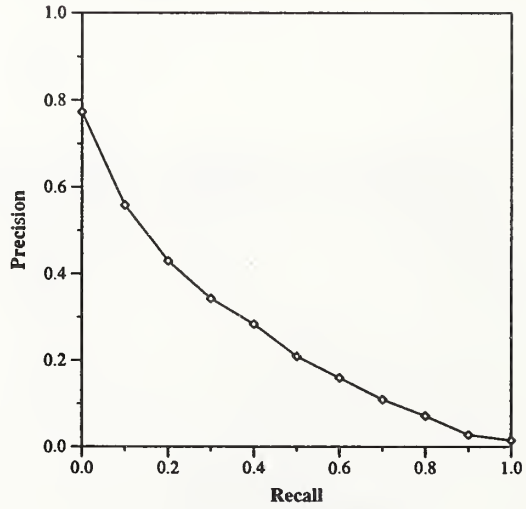
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3017

Recall Level Precision Averages	
Recall	Precision
0.00	0.7730
0.10	0.5586
0.20	0.4301
0.30	0.3434
0.40	0.2843
0.50	0.2097
0.60	0.1597
0.70	0.1095
0.80	0.0715
0.90	0.0279
1.00	0.0151

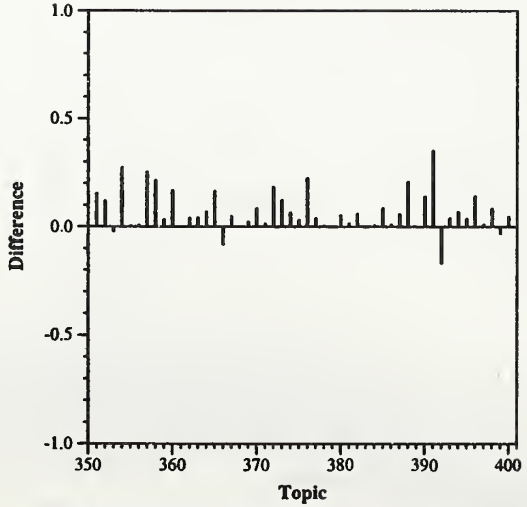
Average precision over all relevant docs	
non-interpolated	0.2521

Document Level Averages	
	Precision
At 5 docs	0.5680
At 10 docs	0.5240
At 15 docs	0.4747
At 20 docs	0.4500
At 30 docs	0.4007
At 100 docs	0.2394
At 200 docs	0.1720
At 500 docs	0.0994
At 1000 docs	0.0603

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2899



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	unc7aal1
Run Description	Automatic, title + desc + narr
Number of Topics	50

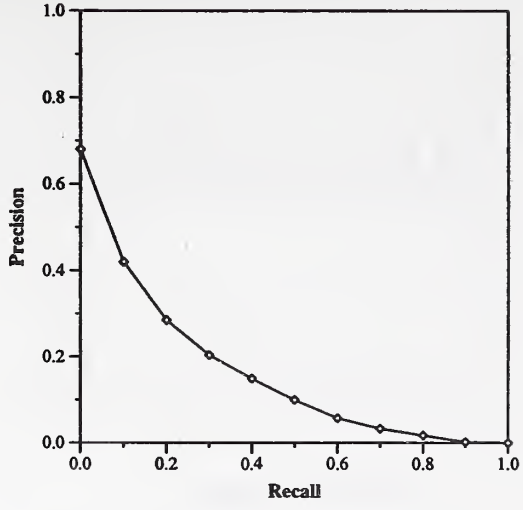
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2188

Recall Level Precision Averages	
Recall	Precision
0.00	0.6803
0.10	0.4201
0.20	0.2855
0.30	0.2043
0.40	0.1494
0.50	0.0996
0.60	0.0575
0.70	0.0330
0.80	0.0173
0.90	0.0019
1.00	0.0000

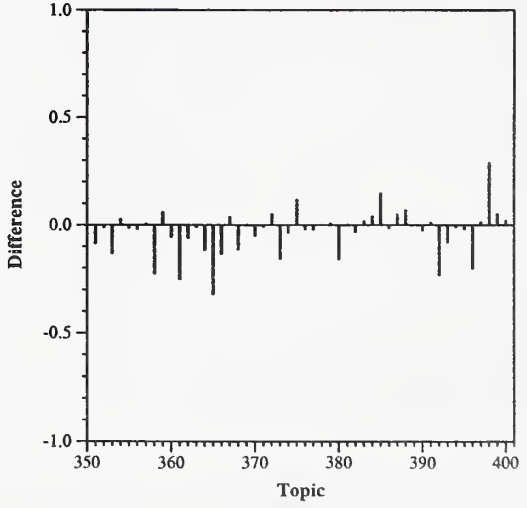
Average precision over all relevant docs	
non-interpolated	0.1506

Document Level Averages	
	Precision
At 5 docs	0.4640
At 10 docs	0.4260
At 15 docs	0.3787
At 20 docs	0.3380
At 30 docs	0.3047
At 100 docs	0.1774
At 200 docs	0.1231
At 500 docs	0.0707
At 1000 docs	0.0438

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1958



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	unc7aal2
Run Description	Automatic, title + desc + narr
Number of Topics	50

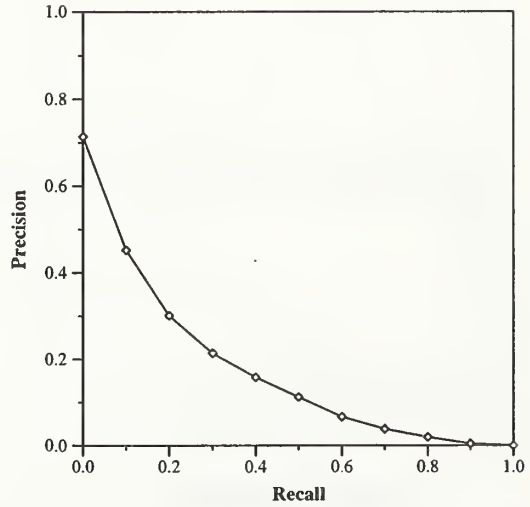
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2264

Recall Level Precision Averages	
Recall	Precision
0.00	0.7137
0.10	0.4522
0.20	0.3012
0.30	0.2142
0.40	0.1584
0.50	0.1124
0.60	0.0668
0.70	0.0386
0.80	0.0196
0.90	0.0043
1.00	0.0000

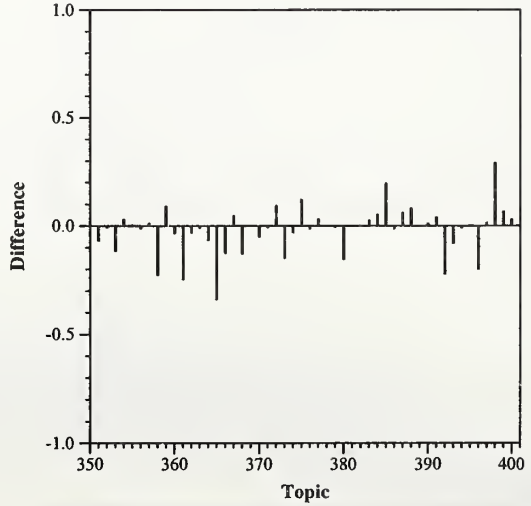
Average precision over all relevant docs	
non-interpolated	0.1618

Document Level Averages	
	Precision
At 5 docs	0.4800
At 10 docs	0.4160
At 15 docs	0.3907
At 20 docs	0.3620
At 30 docs	0.3147
At 100 docs	0.1848
At 200 docs	0.1284
At 500 docs	0.0738
At 1000 docs	0.0453

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2094



Recall-Precision Curve

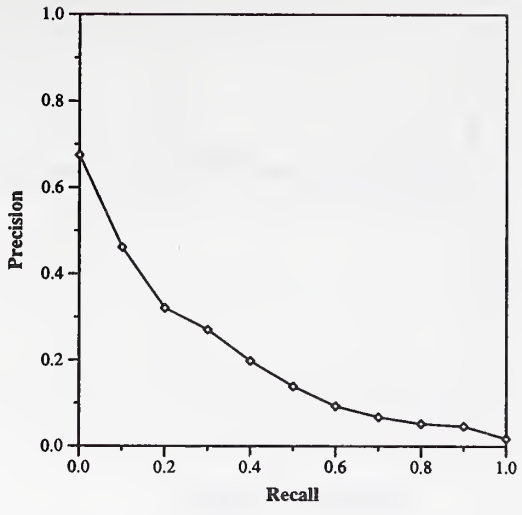


Difference from Median in Average Precision per Topic

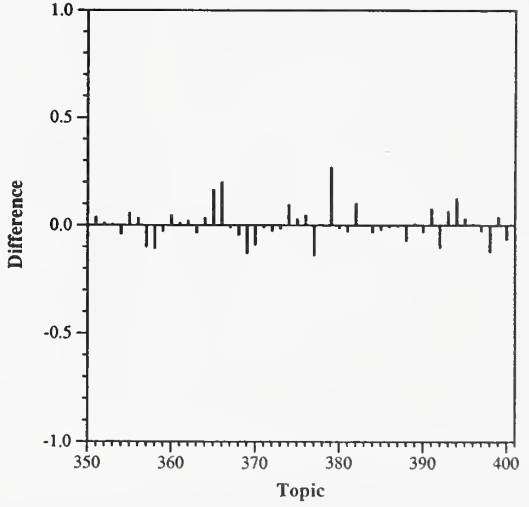
Summary Statistics	
Run Number	uwmt7a0
Run Description	Automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	45904
Relevant:	4674
Rel-ret:	2091

Recall Level Precision Averages	
Recall	Precision
0.00	0.6750
0.10	0.4620
0.20	0.3214
0.30	0.2711
0.40	0.1986
0.50	0.1393
0.60	0.0930
0.70	0.0680
0.80	0.0524
0.90	0.0465
1.00	0.0177
Average precision over all relevant docs	
non-interpolated	0.1866

Document Level Averages	
	Precision
At 5 docs	0.4440
At 10 docs	0.4180
At 15 docs	0.3907
At 20 docs	0.3560
At 30 docs	0.3120
At 100 docs	0.1812
At 200 docs	0.1233
At 500 docs	0.0702
At 1000 docs	0.0418
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2289



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	nsasgrp3
Run Description	Automatic, title + desc + narr
Number of Topics	50

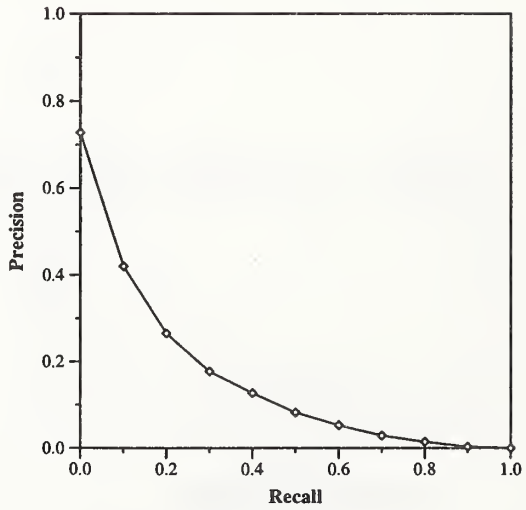
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	1998

Recall Level Precision Averages	
Recall	Precision
0.00	0.7277
0.10	0.4203
0.20	0.2657
0.30	0.1775
0.40	0.1275
0.50	0.0824
0.60	0.0528
0.70	0.0290
0.80	0.0144
0.90	0.0032
1.00	0.0002

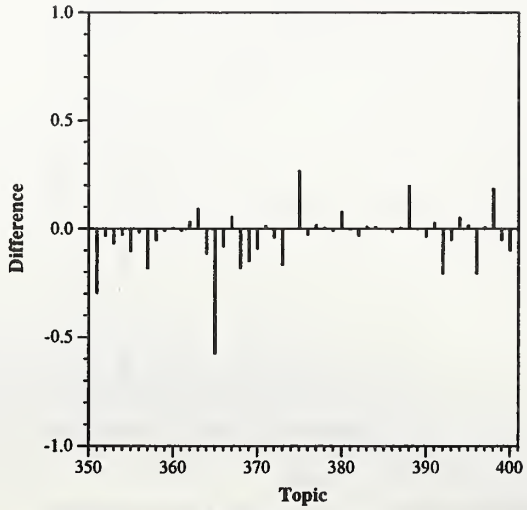
Average precision over all relevant docs	
non-interpolated	0.1449

Document Level Averages	
	Precision
At 5 docs	0.4400
At 10 docs	0.3700
At 15 docs	0.3387
At 20 docs	0.3060
At 30 docs	0.2693
At 100 docs	0.1612
At 200 docs	0.1110
At 500 docs	0.0644
At 1000 docs	0.0400

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1965



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	nsasgrp4
Run Description	Automatic, title + desc + narr
Number of Topics	50

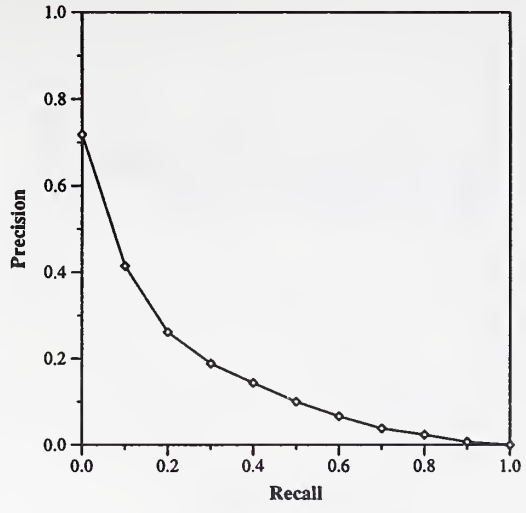
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2063

Recall Level Precision Averages	
Recall	Precision
0.00	0.7184
0.10	0.4150
0.20	0.2620
0.30	0.1891
0.40	0.1441
0.50	0.1005
0.60	0.0665
0.70	0.0383
0.80	0.0238
0.90	0.0070
1.00	0.0001

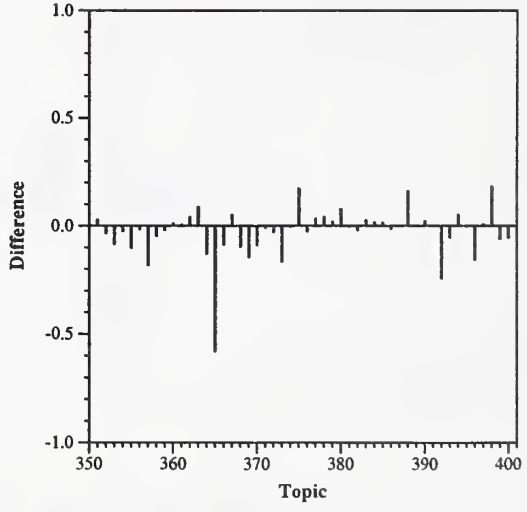
Average precision over all relevant docs	
non-interpolated	0.1537

Document Level Averages	
	Precision
At 5 docs	0.4680
At 10 docs	0.3860
At 15 docs	0.3360
At 20 docs	0.3120
At 30 docs	0.2693
At 100 docs	0.1672
At 200 docs	0.1171
At 500 docs	0.0674
At 1000 docs	0.0413

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2048



Recall-Precision Curve

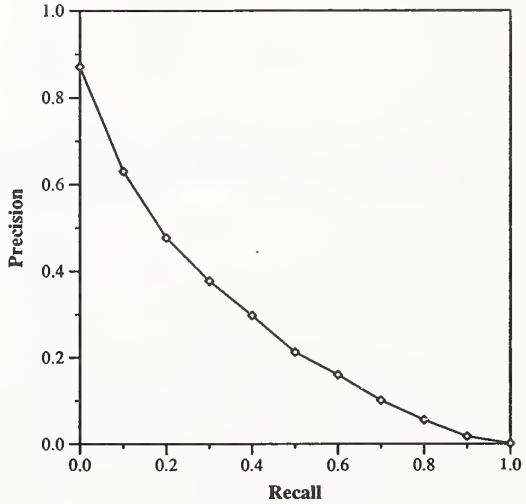


Difference from Median in Average Precision per Topic

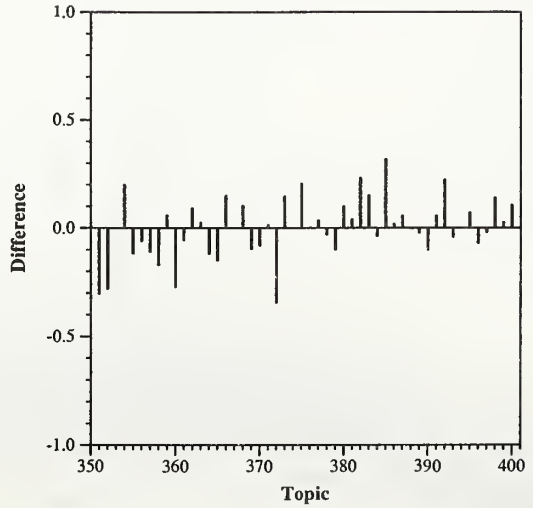
Summary Statistics	
Run Number	acsys7mi
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2552

Recall Level Precision Averages	
Recall	Precision
0.00	0.8716
0.10	0.6309
0.20	0.4777
0.30	0.3780
0.40	0.2982
0.50	0.2132
0.60	0.1607
0.70	0.1014
0.80	0.0559
0.90	0.0174
1.00	0.0013
Average precision over all relevant docs	
non-interpolated	0.2669

Document Level Averages	
	Precision
At 5 docs	0.6720
At 10 docs	0.5960
At 15 docs	0.5253
At 20 docs	0.4860
At 30 docs	0.4173
At 100 docs	0.2524
At 200 docs	0.1701
At 500 docs	0.0886
At 1000 docs	0.0510
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3181



Recall-Precision Curve

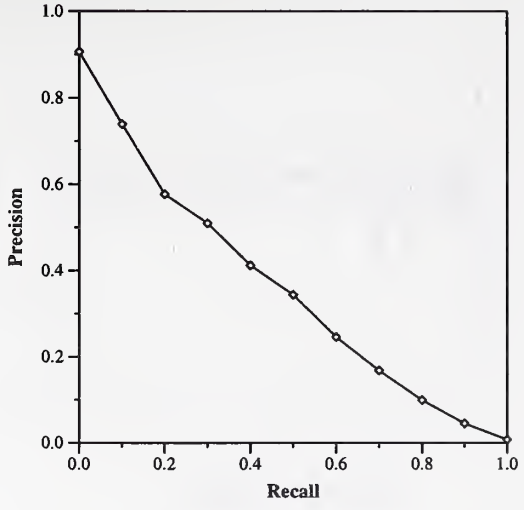


Difference from Median in Average Precision per Topic

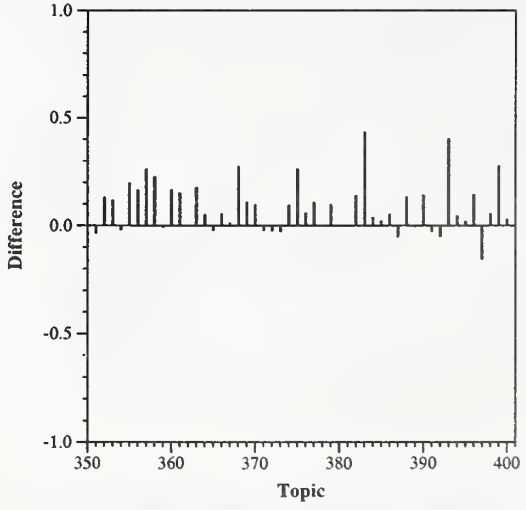
Summary Statistics	
Run Number	CLARIT98CLUS
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3310

Recall Level Precision Averages	
Recall	Precision
0.00	0.9066
0.10	0.7399
0.20	0.5774
0.30	0.5103
0.40	0.4127
0.50	0.3445
0.60	0.2465
0.70	0.1686
0.80	0.0997
0.90	0.0454
1.00	0.0076
Average precision over all relevant docs	
non-interpolated	0.3525

Document Level Averages	
	Precision
At 5 docs	0.7600
At 10 docs	0.6940
At 15 docs	0.6360
At 20 docs	0.5870
At 30 docs	0.5100
At 100 docs	0.2982
At 200 docs	0.1979
At 500 docs	0.1106
At 1000 docs	0.0662
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3730



Recall-Precision Curve

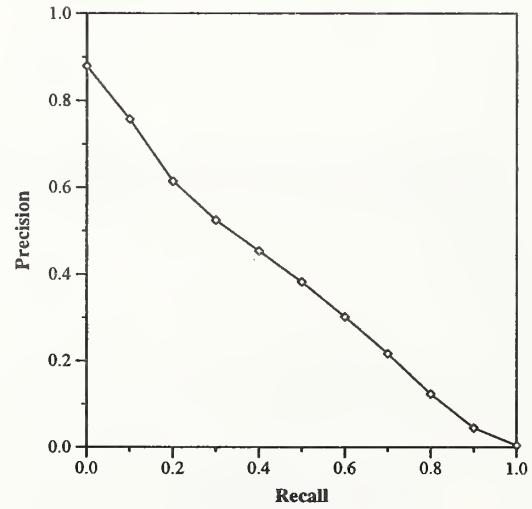


Difference from Median in Average Precision per Topic

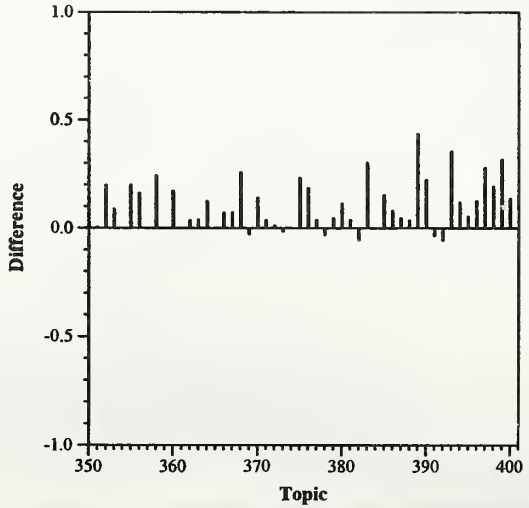
Summary Statistics	
Run Number	CLARIT98COMB
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3417

Recall Level Precision Averages	
Recall	Precision
0.00	0.8796
0.10	0.7575
0.20	0.6148
0.30	0.5250
0.40	0.4538
0.50	0.3826
0.60	0.3018
0.70	0.2168
0.80	0.1235
0.90	0.0451
1.00	0.0045
Average precision over all relevant docs	
non-interpolated	0.3702

Document Level Averages	
	Precision
At 5 docs	0.6920
At 10 docs	0.6940
At 15 docs	0.6613
At 20 docs	0.6180
At 30 docs	0.5653
At 100 docs	0.3178
At 200 docs	0.2142
At 500 docs	0.1147
At 1000 docs	0.0683
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4140



Recall-Precision Curve

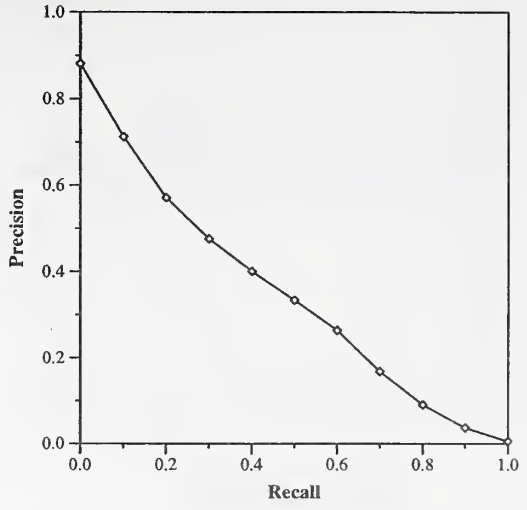


Difference from Median in Average Precision per Topic

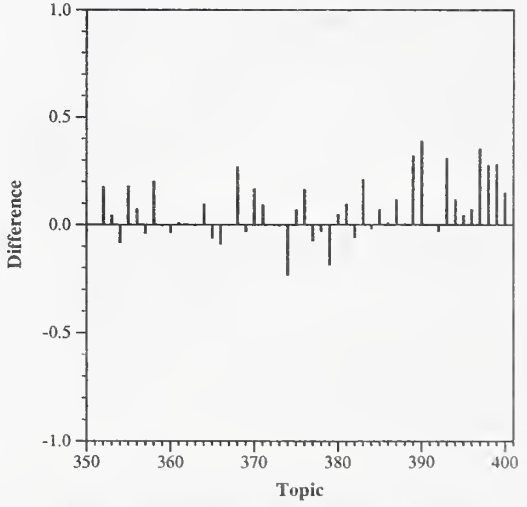
Summary Statistics	
Run Number	CLARIT98RANK
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3192

Recall Level Precision Averages	
Recall	Precision
0.00	0.8814
0.10	0.7124
0.20	0.5708
0.30	0.4759
0.40	0.4008
0.50	0.3338
0.60	0.2638
0.70	0.1684
0.80	0.0908
0.90	0.0372
1.00	0.0055
Average precision over all relevant docs	
non-interpolated	0.3351

Document Level Averages	
	Precision
At 5 docs	0.6720
At 10 docs	0.6440
At 15 docs	0.6320
At 20 docs	0.6050
At 30 docs	0.5327
At 100 docs	0.2864
At 200 docs	0.1975
At 500 docs	0.1074
At 1000 docs	0.0638
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3726



Recall-Precision Curve

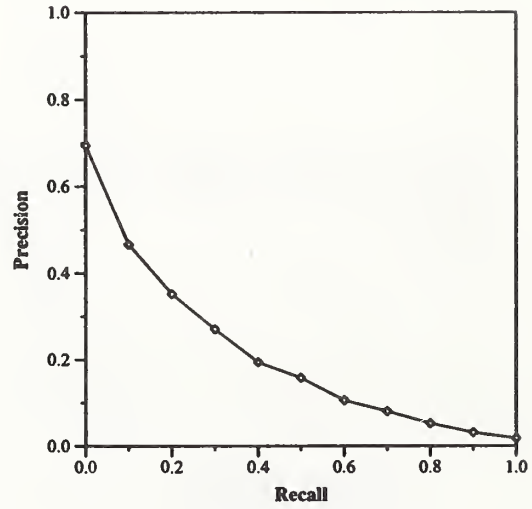


Difference from Median in Average Precision per Topic

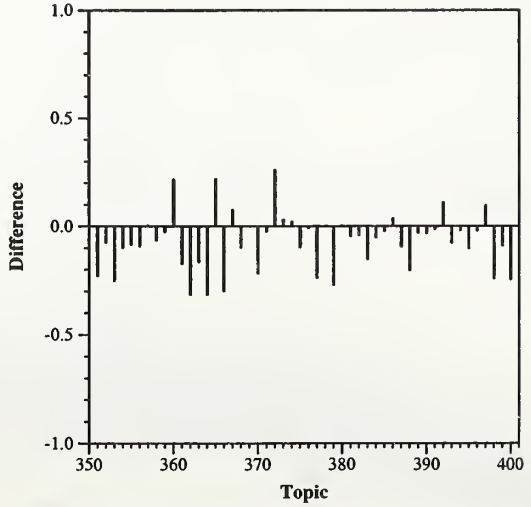
Summary Statistics	
Run Number	fsclt7m
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	18968
Relevant:	4674
Rel-ret:	1849

Recall Level Precision Averages	
Recall	Precision
0.00	0.6951
0.10	0.4667
0.20	0.3526
0.30	0.2709
0.40	0.1944
0.50	0.1582
0.60	0.1061
0.70	0.0806
0.80	0.0527
0.90	0.0314
1.00	0.0179
Average precision over all relevant docs	
non-interpolated	0.1961

Document Level Averages	
	Precision
At 5 docs	0.4600
At 10 docs	0.4160
At 15 docs	0.3947
At 20 docs	0.3810
At 30 docs	0.3333
At 100 docs	0.1962
At 200 docs	0.1270
At 500 docs	0.0648
At 1000 docs	0.0370
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2562



Recall-Precision Curve

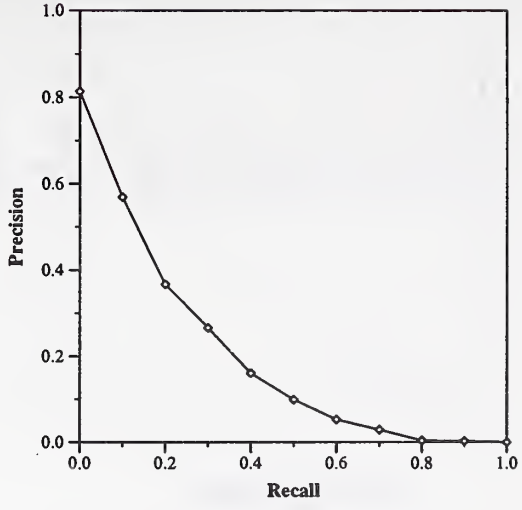


Difference from Median in Average Precision per Topic

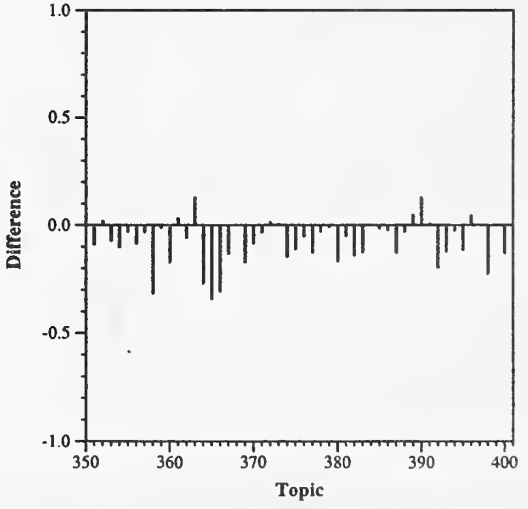
Summary Statistics	
Run Number	gersh1
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	1998

Recall Level Precision Averages	
Recall	Precision
0.00	0.8140
0.10	0.5691
0.20	0.3672
0.30	0.2665
0.40	0.1607
0.50	0.0993
0.60	0.0528
0.70	0.0292
0.80	0.0039
0.90	0.0025
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.1898

Document Level Averages	
	Precision
At 5 docs	0.6480
At 10 docs	0.5620
At 15 docs	0.4987
At 20 docs	0.4450
At 30 docs	0.3680
At 100 docs	0.1818
At 200 docs	0.1148
At 500 docs	0.0642
At 1000 docs	0.0400
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2417



Recall-Precision Curve

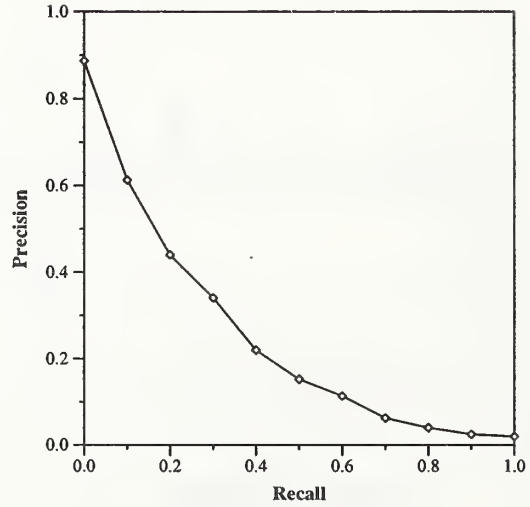


Difference from Median in Average Precision per Topic

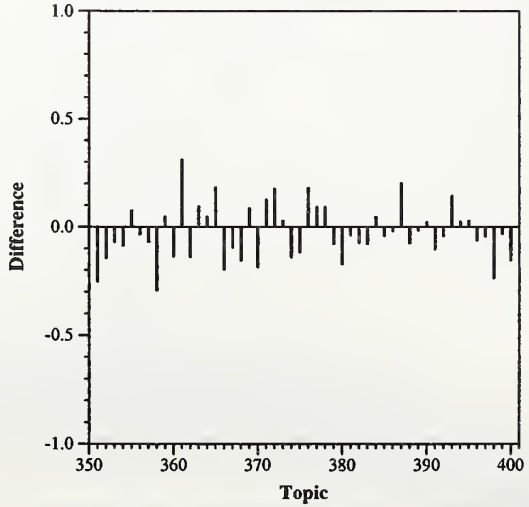
Summary Statistics	
Run Number	harris1
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	40827
Relevant:	4674
Rel-ret:	2233

Recall Level Precision Averages	
Recall	Precision
0.00	0.8872
0.10	0.6131
0.20	0.4404
0.30	0.3412
0.40	0.2203
0.50	0.1527
0.60	0.1140
0.70	0.0628
0.80	0.0403
0.90	0.0252
1.00	0.0199
Average precision over all relevant docs	
non-interpolated	0.2400

Document Level Averages	
	Precision
At 5 docs	0.6880
At 10 docs	0.6160
At 15 docs	0.5533
At 20 docs	0.5170
At 30 docs	0.4293
At 100 docs	0.1992
At 200 docs	0.1235
At 500 docs	0.0702
At 1000 docs	0.0447
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2850



Recall-Precision Curve

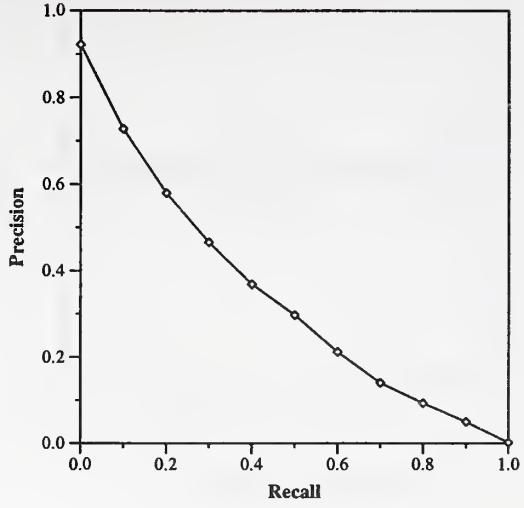


Difference from Median in Average Precision per Topic

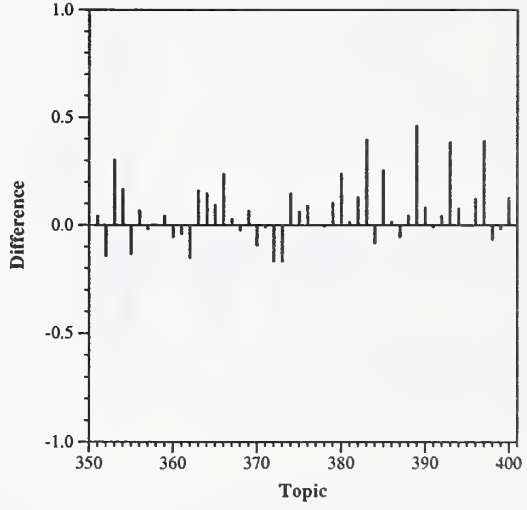
Summary Statistics	
Run Number	iit98ma1
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	26640
Relevant:	4674
Rel-ret:	2793

Recall Level Precision Averages	
Recall	Precision
0.00	0.9220
0.10	0.7280
0.20	0.5796
0.30	0.4661
0.40	0.3692
0.50	0.2976
0.60	0.2121
0.70	0.1405
0.80	0.0933
0.90	0.0500
1.00	0.0019
Average precision over all relevant docs	
non-interpolated	0.3333

Document Level Averages	
	Precision
At 5 docs	0.7320
At 10 docs	0.6400
At 15 docs	0.6027
At 20 docs	0.5490
At 30 docs	0.4887
At 100 docs	0.3006
At 200 docs	0.1957
At 500 docs	0.1002
At 1000 docs	0.0559
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3688



Recall-Precision Curve

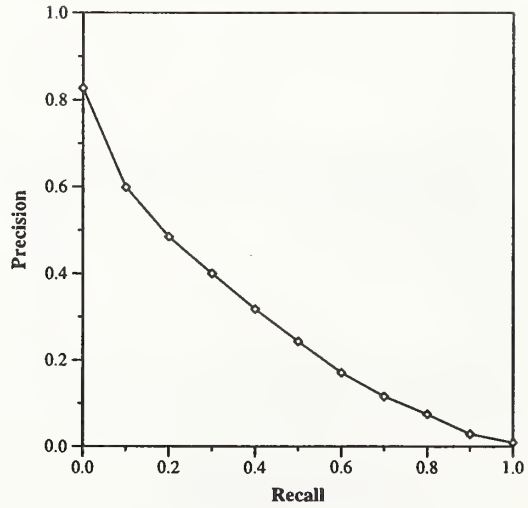


Difference from Median in Average Precision per Topic

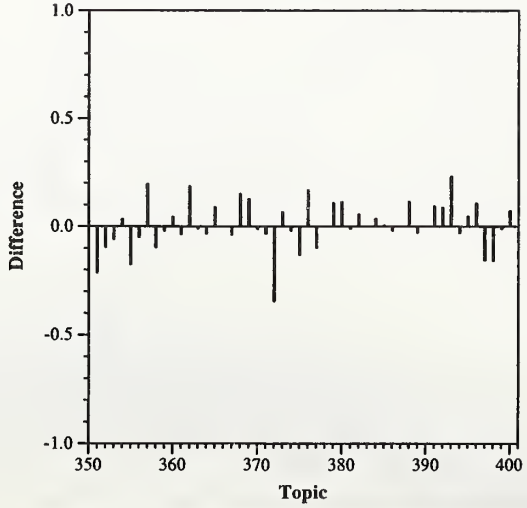
Summary Statistics	
Run Number	LNmanual7
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3005

Recall Level Precision Averages	
Recall	Precision
0.00	0.8272
0.10	0.5992
0.20	0.4848
0.30	0.4004
0.40	0.3182
0.50	0.2433
0.60	0.1709
0.70	0.1163
0.80	0.0750
0.90	0.0290
1.00	0.0092
Average precision over all relevant docs	
non-interpolated	0.2722

Document Level Averages	
	Precision
At 5 docs	0.5720
At 10 docs	0.5320
At 15 docs	0.4880
At 20 docs	0.4490
At 30 docs	0.3927
At 100 docs	0.2578
At 200 docs	0.1821
At 500 docs	0.1008
At 1000 docs	0.0601
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3190



Recall-Precision Curve

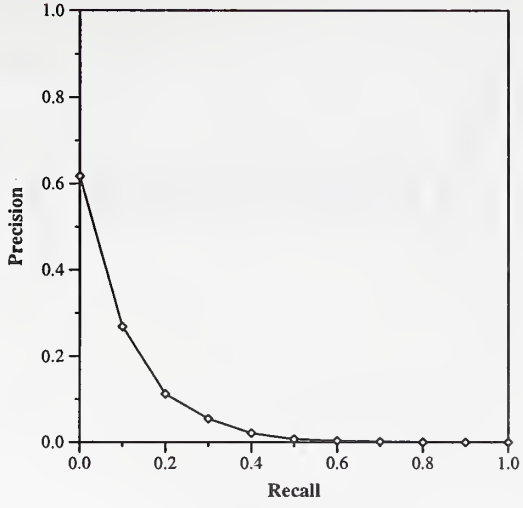


Difference from Median in Average Precision per Topic

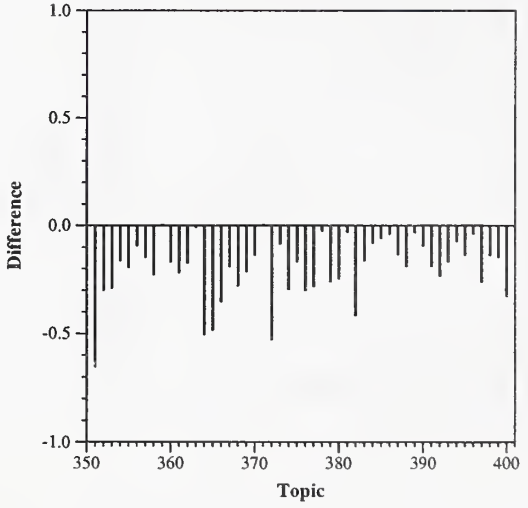
Summary Statistics	
Run Number	lanl981
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	890

Recall Level Precision Averages	
Recall	Precision
0.00	0.6173
0.10	0.2689
0.20	0.1123
0.30	0.0546
0.40	0.0211
0.50	0.0072
0.60	0.0034
0.70	0.0014
0.80	0.0000
0.90	0.0000
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.0691

Document Level Averages	
	Precision
At 5 docs	0.3400
At 10 docs	0.2880
At 15 docs	0.2467
At 20 docs	0.2230
At 30 docs	0.1813
At 100 docs	0.0910
At 200 docs	0.0565
At 500 docs	0.0290
At 1000 docs	0.0178
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1182



Recall-Precision Curve

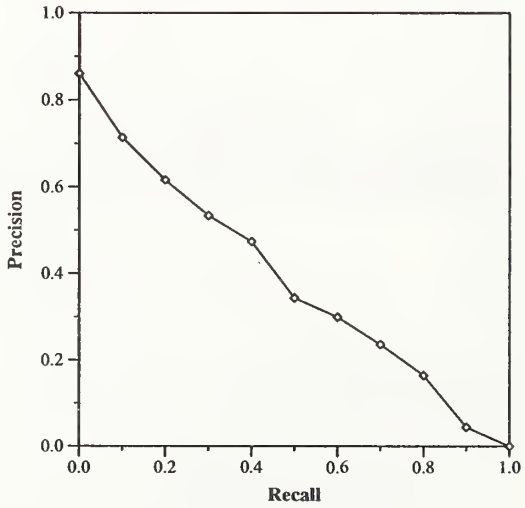


Difference from Median in Average Precision per Topic

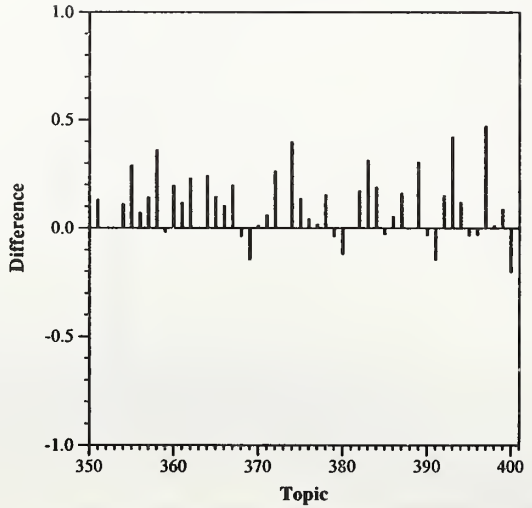
Summary Statistics	
Run Number	t7miti1
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	5898
Relevant:	4674
Rel-ret:	2520

Recall Level Precision Averages	
Recall	Precision
0.00	0.8610
0.10	0.7139
0.20	0.6159
0.30	0.5337
0.40	0.4739
0.50	0.3438
0.60	0.2994
0.70	0.2360
0.80	0.1645
0.90	0.0443
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.3675

Document Level Averages	
	Precision
At 5 docs	0.6640
At 10 docs	0.6400
At 15 docs	0.6213
At 20 docs	0.5780
At 30 docs	0.5433
At 100 docs	0.3512
At 200 docs	0.2218
At 500 docs	0.1008
At 1000 docs	0.0504
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4392



Recall-Precision Curve

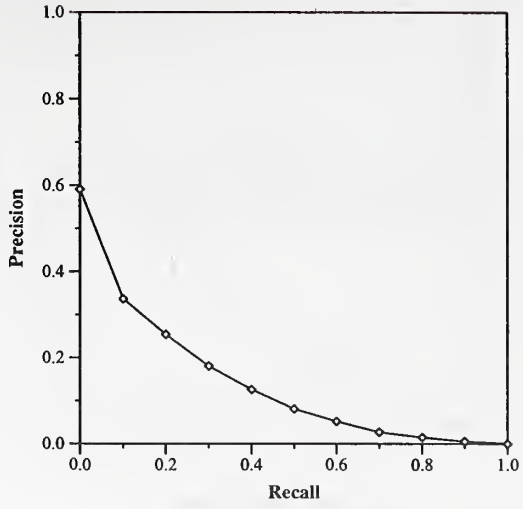


Difference from Median in Average Precision per Topic

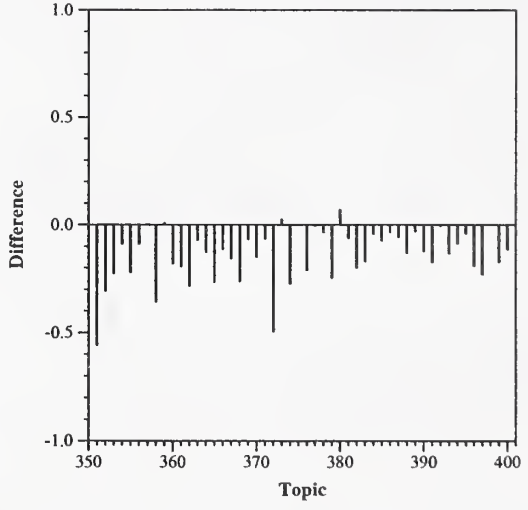
Summary Statistics	
Run Number	nthu1
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2004

Recall Level Precision Averages	
Recall	Precision
0.00	0.5911
0.10	0.3369
0.20	0.2543
0.30	0.1810
0.40	0.1267
0.50	0.0814
0.60	0.0525
0.70	0.0270
0.80	0.0151
0.90	0.0056
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.1272

Document Level Averages	
	Precision
At 5 docs	0.3520
At 10 docs	0.3180
At 15 docs	0.2933
At 20 docs	0.2650
At 30 docs	0.2320
At 100 docs	0.1450
At 200 docs	0.1031
At 500 docs	0.0618
At 1000 docs	0.0401
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1853



Recall-Precision Curve

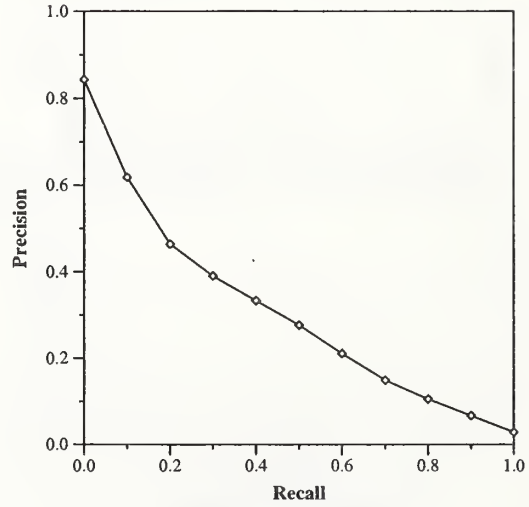


Difference from Median in Average Precision per Topic

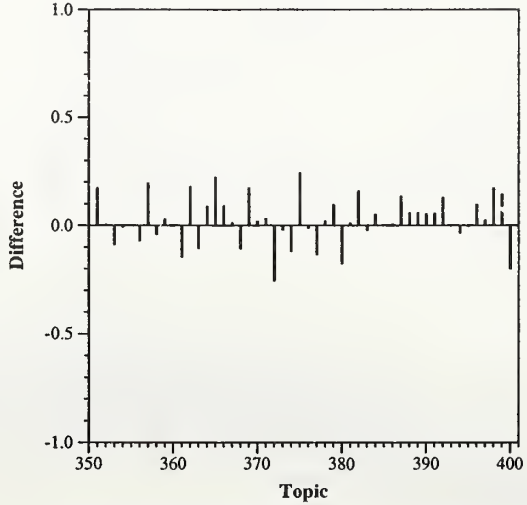
Summary Statistics	
Run Number	Brkly26
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	3174

Recall Level Precision Averages	
Recall	Precision
0.00	0.8434
0.10	0.6183
0.20	0.4643
0.30	0.3907
0.40	0.3336
0.50	0.2772
0.60	0.2112
0.70	0.1494
0.80	0.1055
0.90	0.0670
1.00	0.0284
Average precision over all relevant docs	
non-interpolated	0.2905

Document Level Averages	
	Precision
At 5 docs	0.6000
At 10 docs	0.5440
At 15 docs	0.5173
At 20 docs	0.4860
At 30 docs	0.4287
At 100 docs	0.2756
At 200 docs	0.1893
At 500 docs	0.1066
At 1000 docs	0.0635
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3370



Recall-Precision Curve

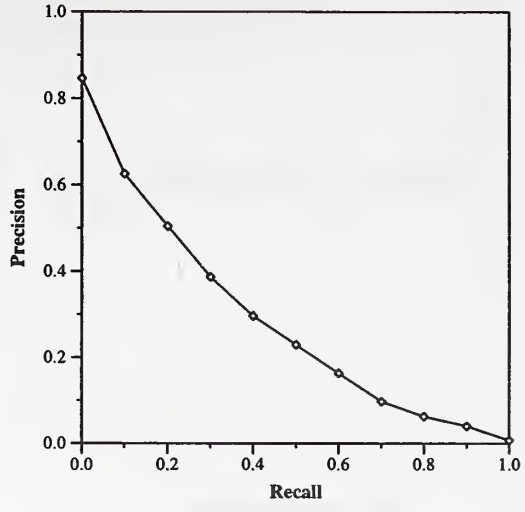


Difference from Median in Average Precision per Topic

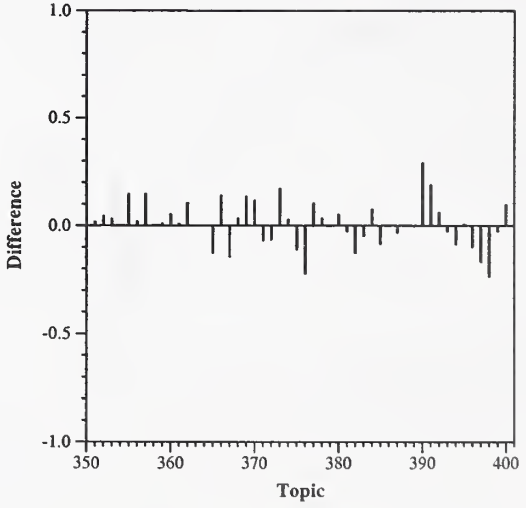
Summary Statistics	
Run Number	uoftimgr
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2817

Recall Level Precision Averages	
Recall	Precision
0.00	0.8468
0.10	0.6256
0.20	0.5046
0.30	0.3870
0.40	0.2965
0.50	0.2294
0.60	0.1633
0.70	0.0978
0.80	0.0626
0.90	0.0407
1.00	0.0074
Average precision over all relevant docs	
non-interpolated	0.2755

Document Level Averages	
	Precision
At 5 docs	0.6600
At 10 docs	0.5980
At 15 docs	0.5200
At 20 docs	0.4950
At 30 docs	0.4207
At 100 docs	0.2650
At 200 docs	0.1792
At 500 docs	0.0952
At 1000 docs	0.0563
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3109



Recall-Precision Curve

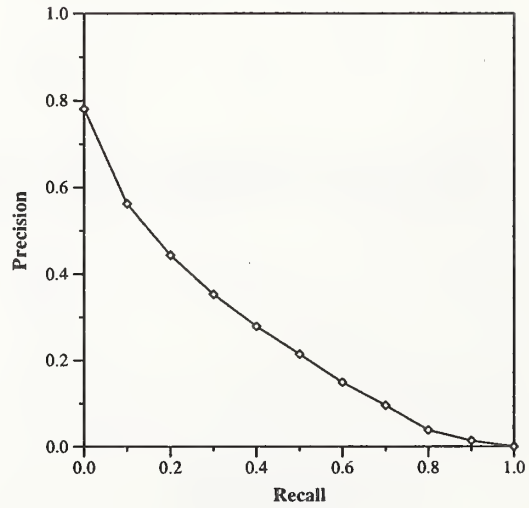


Difference from Median in Average Precision per Topic

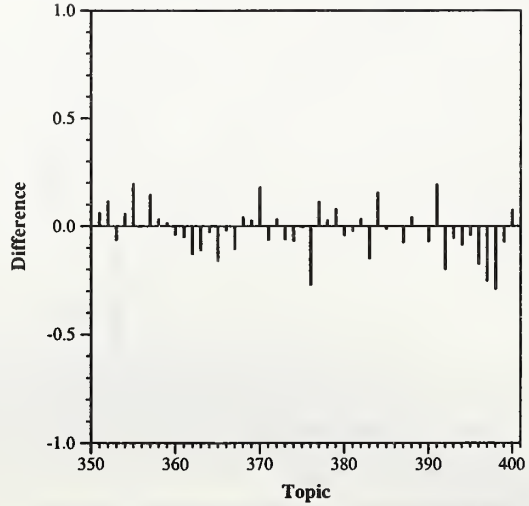
Summary Statistics	
Run Number	uoftimgu
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel-ret:	2585

Recall Level Precision Averages	
Recall	Precision
0.00	0.7810
0.10	0.5625
0.20	0.4439
0.30	0.3535
0.40	0.2790
0.50	0.2149
0.60	0.1494
0.70	0.0959
0.80	0.0383
0.90	0.0136
1.00	0.0000
Average precision over all relevant docs	
non-interpolated	0.2454

Document Level Averages	
	Precision
At 5 docs	0.5760
At 10 docs	0.5240
At 15 docs	0.4813
At 20 docs	0.4500
At 30 docs	0.3947
At 100 docs	0.2404
At 200 docs	0.1637
At 500 docs	0.0885
At 1000 docs	0.0517
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2858



Recall-Precision Curve

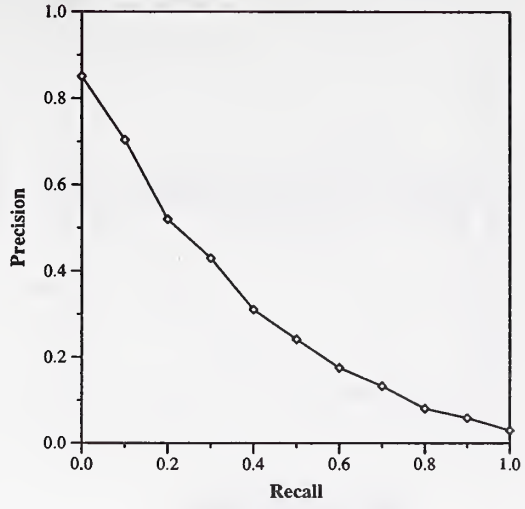


Difference from Median in Average Precision per Topic

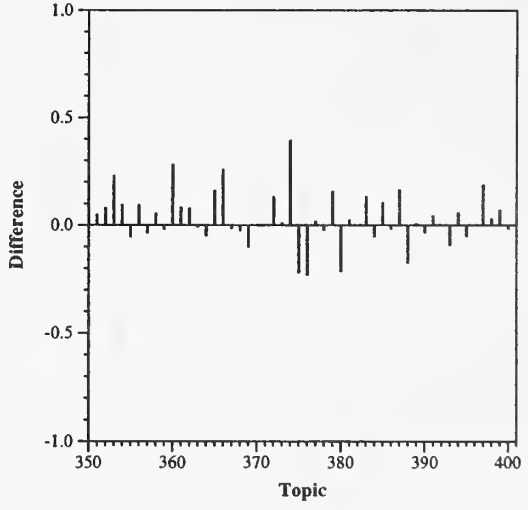
Summary Statistics	
Run Number	uwmt7a1
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	38690
Relevant:	4674
Rel-ret:	2612

Recall Level Precision Averages	
Recall	Precision
0.00	0.8508
0.10	0.7045
0.20	0.5201
0.30	0.4303
0.40	0.3108
0.50	0.2418
0.60	0.1757
0.70	0.1332
0.80	0.0809
0.90	0.0590
1.00	0.0299
Average precision over all relevant docs	
non-interpolated	0.2983

Document Level Averages	
	Precision
At 5 docs	0.6840
At 10 docs	0.6280
At 15 docs	0.5760
At 20 docs	0.5330
At 30 docs	0.4680
At 100 docs	0.2650
At 200 docs	0.1732
At 500 docs	0.0911
At 1000 docs	0.0522
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3402



Recall-Precision Curve

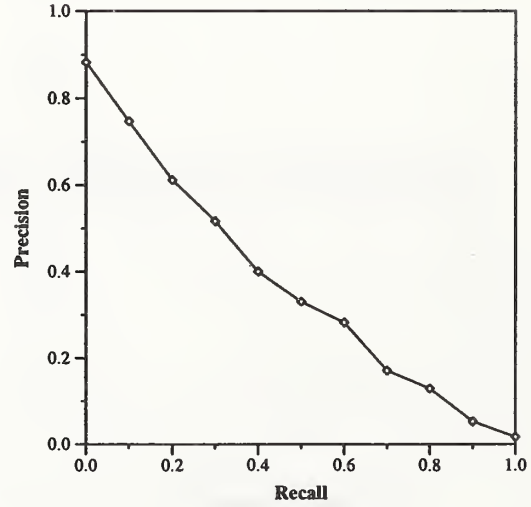


Difference from Median in Average Precision per Topic

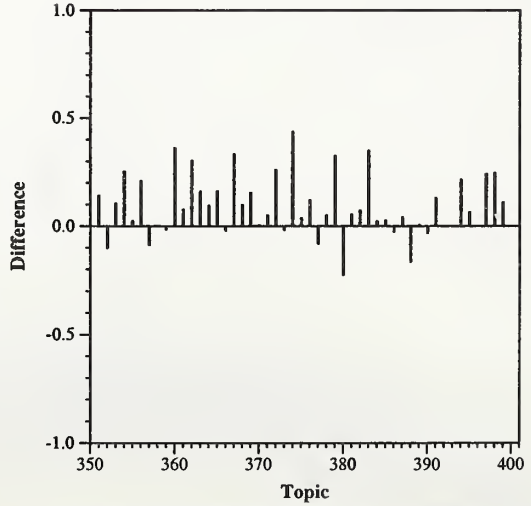
Summary Statistics	
Run Number	uwmt7a2
Run Description	Manual
Number of Topics	50
Total number of documents over all topics	
Retrieved:	16617
Relevant:	4674
Rel-ret:	2720

Recall Level Precision Averages	
Recall	Precision
0.00	0.8829
0.10	0.7471
0.20	0.6118
0.30	0.5171
0.40	0.4003
0.50	0.3307
0.60	0.2825
0.70	0.1710
0.80	0.1295
0.90	0.0529
1.00	0.0171
Average precision over all relevant docs	
non-interpolated	0.3587

Document Level Averages	
	Precision
At 5 docs	0.7240
At 10 docs	0.6540
At 15 docs	0.5960
At 20 docs	0.5610
At 30 docs	0.5173
At 100 docs	0.3160
At 200 docs	0.2081
At 500 docs	0.1021
At 1000 docs	0.0544
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4012



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	98EITdes
Run Description	German topics, EFGI documents automatic, title + desc
Number of Topics	28

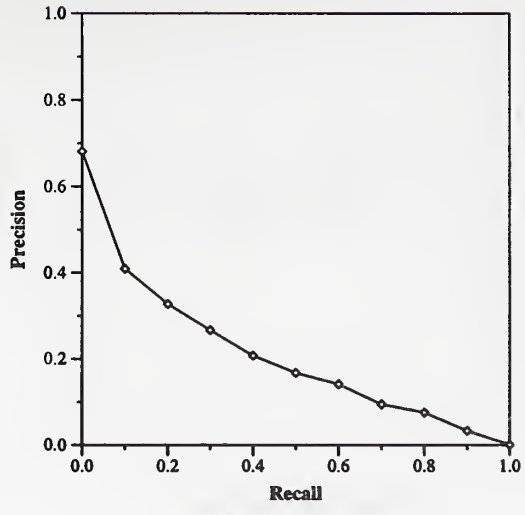
Total number of documents over all topics	
Retrieved:	28000
Relevant:	4098
Rel-ret:	1988

Recall Level Precision Averages	
Recall	Precision
0.00	0.6815
0.10	0.4097
0.20	0.3278
0.30	0.2671
0.40	0.2079
0.50	0.1680
0.60	0.1416
0.70	0.0949
0.80	0.0757
0.90	0.0330
1.00	0.0005

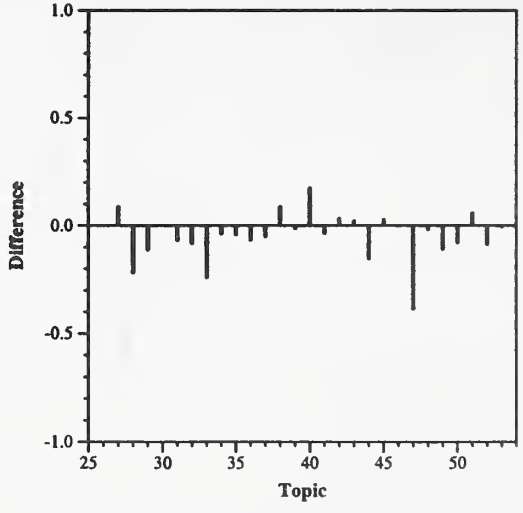
Average precision over all relevant docs	
non-interpolated	0.1962

Document Level Averages	
	Precision
At 5 docs	0.5429
At 10 docs	0.4643
At 15 docs	0.4405
At 20 docs	0.4000
At 30 docs	0.3548
At 100 docs	0.2507
At 200 docs	0.1830
At 500 docs	0.1124
At 1000 docs	0.0710

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2378



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	98EITful
Run Description	German topics, EFGI documents automatic, title + desc + narr
Number of Topics	28

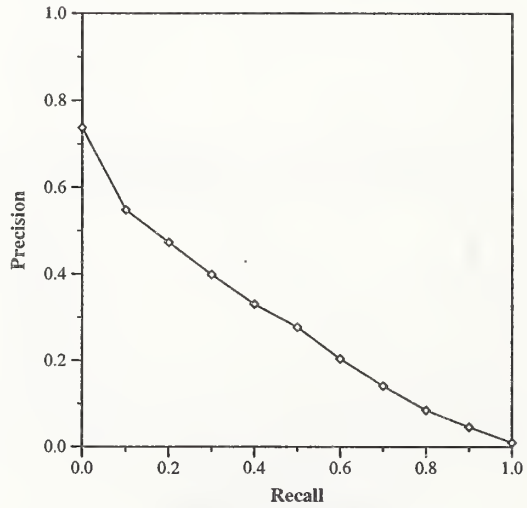
Total number of documents over all topics	
Retrieved:	28000
Relevant:	4098
Rel-ret:	2472

Recall Level Precision Averages	
Recall	Precision
0.00	0.7376
0.10	0.5483
0.20	0.4735
0.30	0.3991
0.40	0.3307
0.50	0.2776
0.60	0.2041
0.70	0.1414
0.80	0.0850
0.90	0.0461
1.00	0.0099

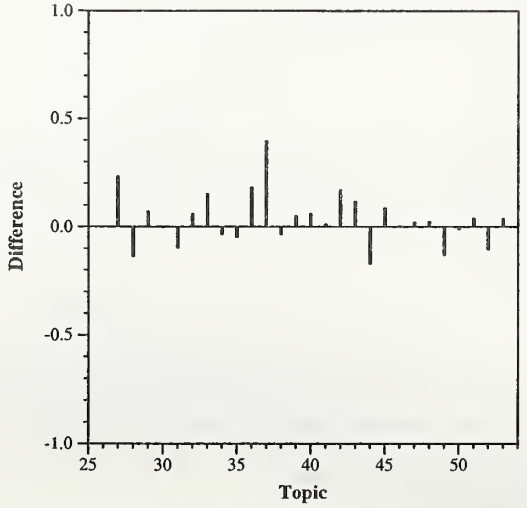
Average precision over all relevant docs	
non-interpolated	0.2767

Document Level Averages	
	Precision
At 5 docs	0.5929
At 10 docs	0.5393
At 15 docs	0.5167
At 20 docs	0.4911
At 30 docs	0.4714
At 100 docs	0.3589
At 200 docs	0.2496
At 500 docs	0.1449
At 1000 docs	0.0883

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3175



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	98EITtit
Run Description	German topics, EFGI documents automatic, title
Number of Topics	28

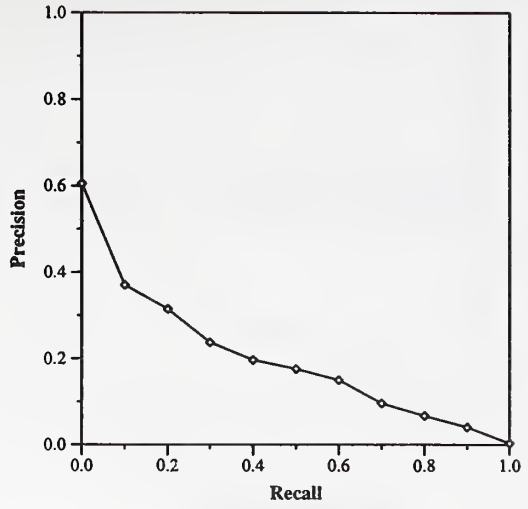
Total number of documents over all topics	
Retrieved:	28000
Relevant:	4098
Rel-ret:	2026

Recall Level Precision Averages	
Recall	Precision
0.00	0.6053
0.10	0.3709
0.20	0.3153
0.30	0.2378
0.40	0.1970
0.50	0.1762
0.60	0.1503
0.70	0.0962
0.80	0.0672
0.90	0.0406
1.00	0.0031

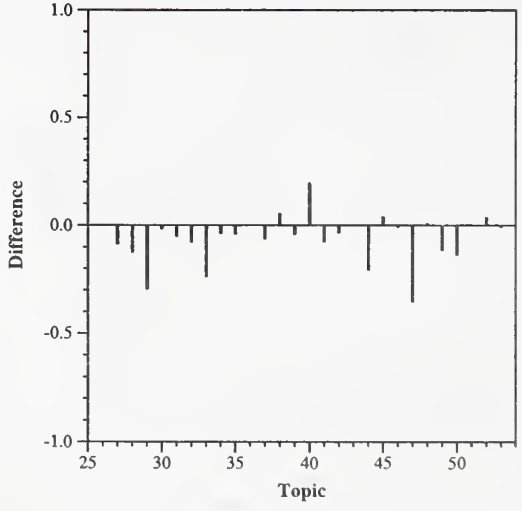
Average precision over all relevant docs	
non-interpolated	0.1841

Document Level Averages	
	Precision
At 5 docs	0.4286
At 10 docs	0.3821
At 15 docs	0.3595
At 20 docs	0.3429
At 30 docs	0.3286
At 100 docs	0.2325
At 200 docs	0.1711
At 500 docs	0.1120
At 1000 docs	0.0724

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2199



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	ibmcl7al
Run Description	English topics, EFGI documents automatic, title + desc + narr
Number of Topics	28

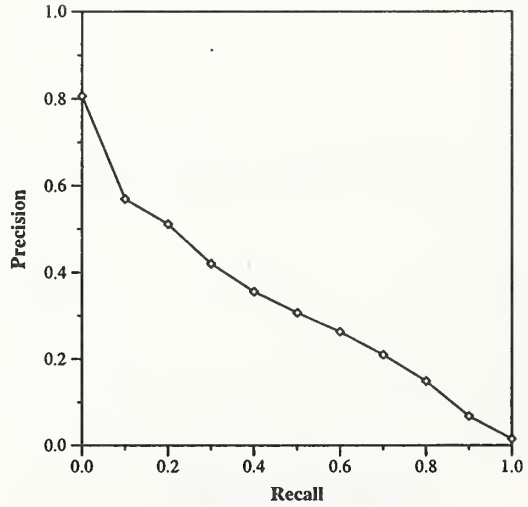
Total number of documents over all topics	
Retrieved:	28000
Relevant:	4098
Rel-ret:	3005

Recall Level Precision Averages	
Recall	Precision
0.00	0.8065
0.10	0.5697
0.20	0.5116
0.30	0.4208
0.40	0.3562
0.50	0.3075
0.60	0.2634
0.70	0.2102
0.80	0.1492
0.90	0.0678
1.00	0.0151

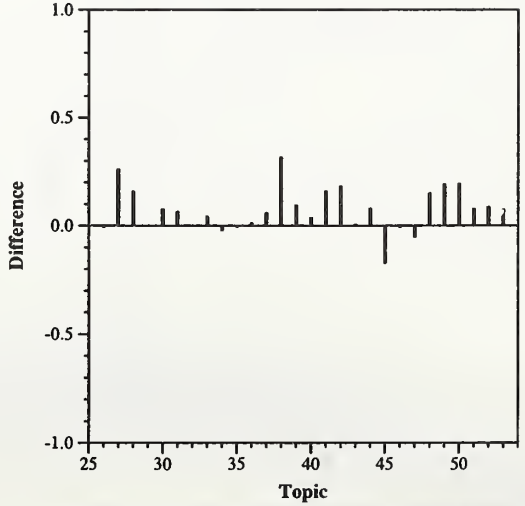
Average precision over all relevant docs	
non-interpolated	0.3168

Document Level Averages	
	Precision
At 5 docs	0.5643
At 10 docs	0.5464
At 15 docs	0.5286
At 20 docs	0.5143
At 30 docs	0.4893
At 100 docs	0.3964
At 200 docs	0.2920
At 500 docs	0.1768
At 1000 docs	0.1073

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3613



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	ibmcl7as
Run Description	English topics, EFGI documents automatic, desc
Number of Topics	28

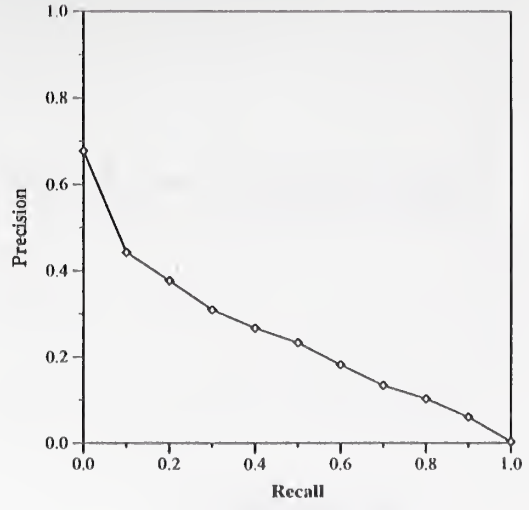
Total number of documents over all topics	
Retrieved:	28000
Relevant:	4098
Rel-ret:	2544

Recall Level Precision Averages	
Recall	Precision
0.00	0.6774
0.10	0.4424
0.20	0.3772
0.30	0.3096
0.40	0.2674
0.50	0.2330
0.60	0.1818
0.70	0.1338
0.80	0.1030
0.90	0.0602
1.00	0.0027

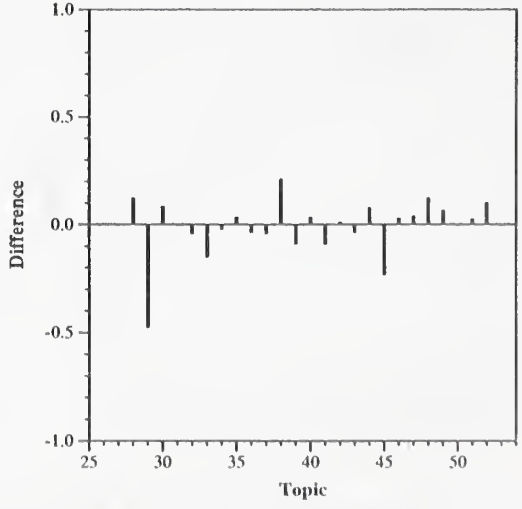
Average precision over all relevant docs	
non-interpolated	0.2337

Document Level Averages	
	Precision
At 5 docs	0.4143
At 10 docs	0.4286
At 15 docs	0.4214
At 20 docs	0.4036
At 30 docs	0.3726
At 100 docs	0.2979
At 200 docs	0.2370
At 500 docs	0.1466
At 1000 docs	0.0909

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2827



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	ibmcl7cl
Run Description	English topics, EFGI documents automatic, title + desc + narr
Number of Topics	28

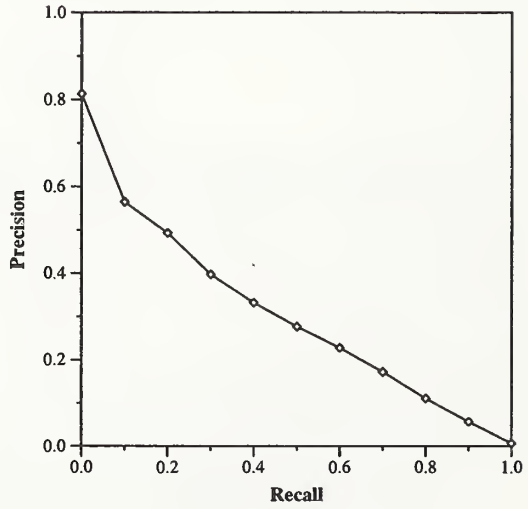
Total number of documents over all topics	
Retrieved:	28000
Relevant:	4098
Rel-ret:	2826

Recall Level Precision Averages	
Recall	Precision
0.00	0.8136
0.10	0.5646
0.20	0.4931
0.30	0.3975
0.40	0.3324
0.50	0.2771
0.60	0.2278
0.70	0.1721
0.80	0.1107
0.90	0.0568
1.00	0.0066

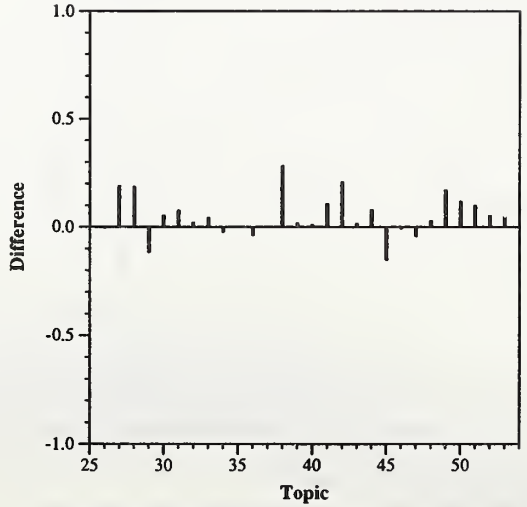
Average precision over all relevant docs	
non-interpolated	0.2942

Document Level Averages	
	Precision
At 5 docs	0.6286
At 10 docs	0.5286
At 15 docs	0.5167
At 20 docs	0.5161
At 30 docs	0.4929
At 100 docs	0.3739
At 200 docs	0.2730
At 500 docs	0.1652
At 1000 docs	0.1009

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3359



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	ibmcl7cs
Run Description	English topics, EFGI documents automatic, desc
Number of Topics	28

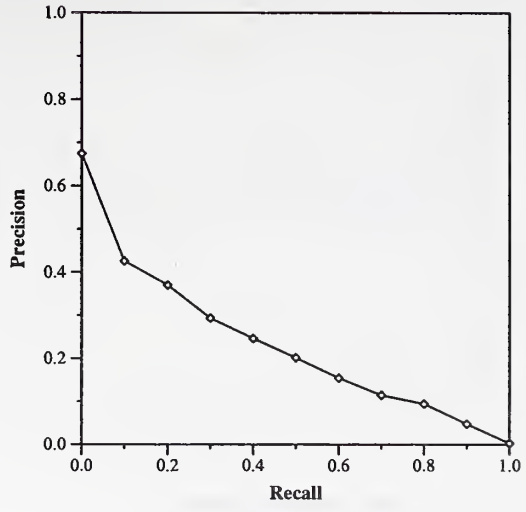
Total number of documents over all topics	
Retrieved:	28000
Relevant:	4098
Rel-ret:	2374

Recall Level Precision Averages	
Recall	Precision
0.00	0.6748
0.10	0.4258
0.20	0.3705
0.30	0.2939
0.40	0.2467
0.50	0.2018
0.60	0.1547
0.70	0.1149
0.80	0.0947
0.90	0.0481
1.00	0.0028

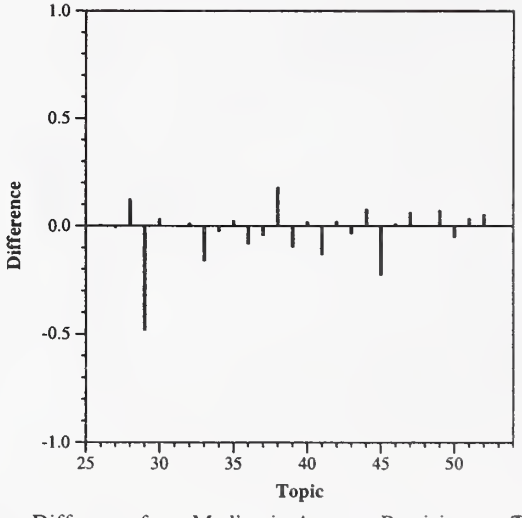
Average precision over all relevant docs	
non-interpolated	0.2212

Document Level Averages	
	Precision
At 5 docs	0.4143
At 10 docs	0.4500
At 15 docs	0.4119
At 20 docs	0.4089
At 30 docs	0.3738
At 100 docs	0.2907
At 200 docs	0.2229
At 500 docs	0.1349
At 1000 docs	0.0848

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2711



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	tno7ddp
Run Description	German topics, EFGI documents automatic, title + desc + narr
Number of Topics	28

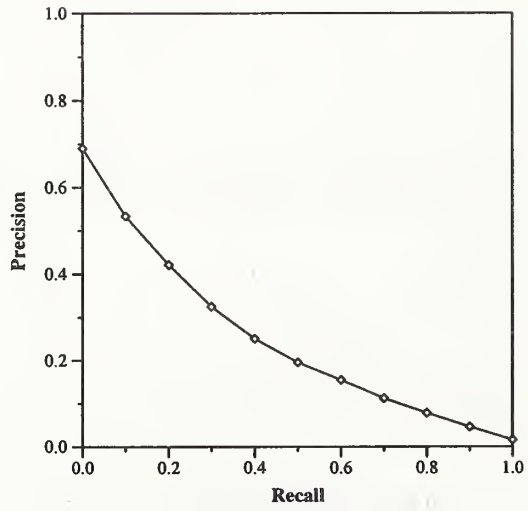
Total number of documents over all topics	
Retrieved:	28000
Relevant:	4098
Rel-ret:	2484

Recall Level Precision Averages	
Recall	Precision
0.00	0.6904
0.10	0.5339
0.20	0.4217
0.30	0.3253
0.40	0.2511
0.50	0.1959
0.60	0.1550
0.70	0.1127
0.80	0.0786
0.90	0.0462
1.00	0.0158

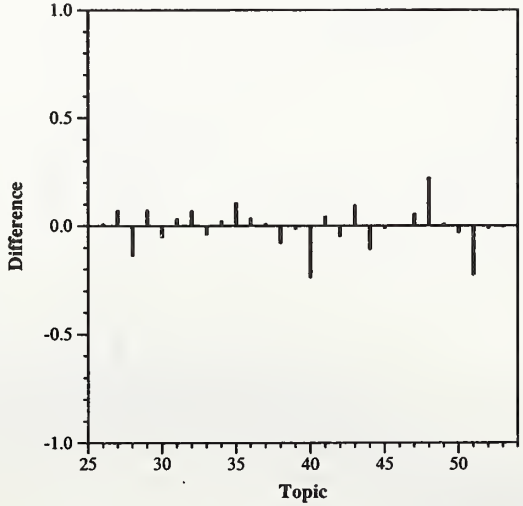
Average precision over all relevant docs	
non-interpolated	0.2382

Document Level Averages	
	Precision
At 5 docs	0.5643
At 10 docs	0.5393
At 15 docs	0.5119
At 20 docs	0.4964
At 30 docs	0.4607
At 100 docs	0.3389
At 200 docs	0.2339
At 500 docs	0.1422
At 1000 docs	0.0887

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2923



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	tno7edp
Run Description	English topics, EFGI documents automatic, title + desc + narr
Number of Topics	28

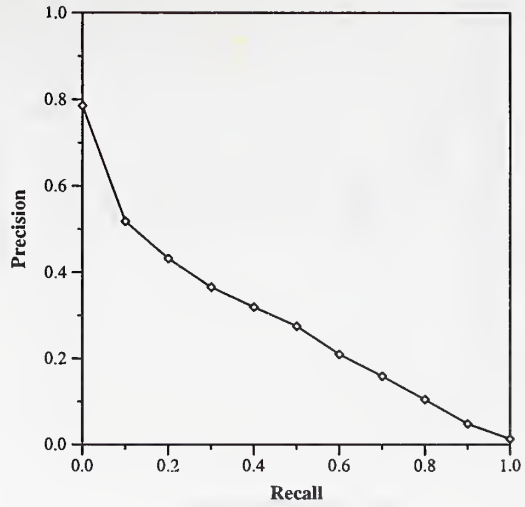
Total number of documents over all topics	
Retrieved:	28000
Relevant:	4098
Rel-ret:	2598

Recall Level Precision Averages	
Recall	Precision
0.00	0.7857
0.10	0.5183
0.20	0.4324
0.30	0.3661
0.40	0.3196
0.50	0.2756
0.60	0.2104
0.70	0.1594
0.80	0.1048
0.90	0.0486
1.00	0.0134

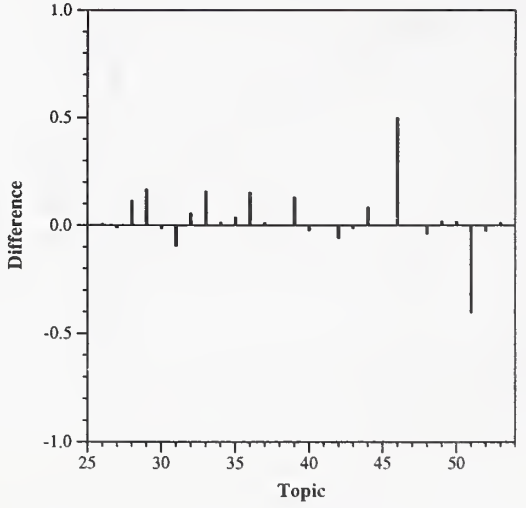
Average precision over all relevant docs	
non-interpolated	0.2716

Document Level Averages	
	Precision
At 5 docs	0.5714
At 10 docs	0.5000
At 15 docs	0.4833
At 20 docs	0.4536
At 30 docs	0.4214
At 100 docs	0.3211
At 200 docs	0.2466
At 500 docs	0.1514
At 1000 docs	0.0928

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3107



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	tno7edpx
Run Description	English topics, EFGI documents automatic, title + desc + narr
Number of Topics	28

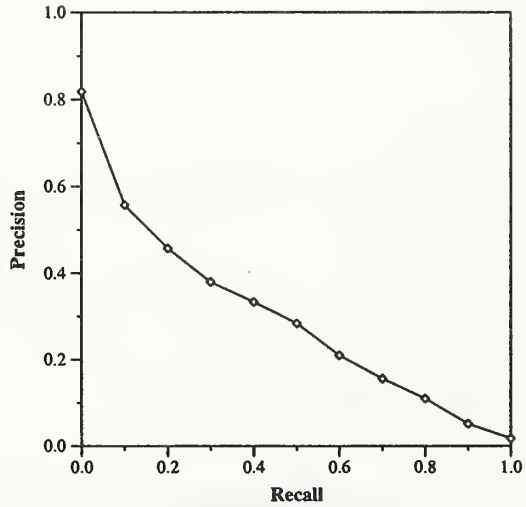
Total number of documents over all topics	
Retrieved:	28000
Relevant:	4098
Rel-ret:	2552

Recall Level Precision Averages	
Recall	Precision
0.00	0.8186
0.10	0.5575
0.20	0.4576
0.30	0.3796
0.40	0.3340
0.50	0.2844
0.60	0.2105
0.70	0.1564
0.80	0.1103
0.90	0.0517
1.00	0.0176

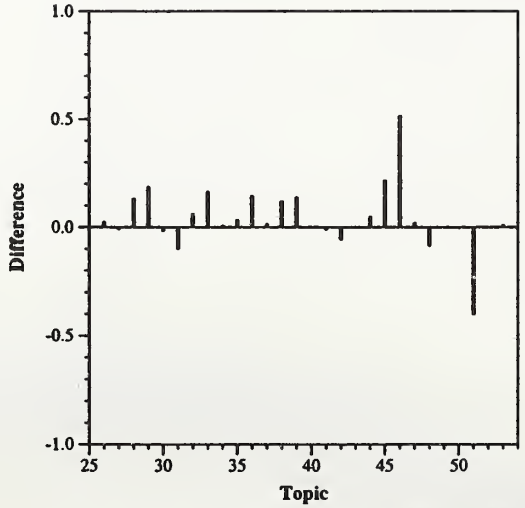
Average precision over all relevant docs	
non-interpolated	0.2846

Document Level Averages	
	Precision
At 5 docs	0.6000
At 10 docs	0.5500
At 15 docs	0.5262
At 20 docs	0.4857
At 30 docs	0.4690
At 100 docs	0.3389
At 200 docs	0.2457
At 500 docs	0.1496
At 1000 docs	0.0911

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3270



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	tno7egr
Run Description	English topics, EFGI documents automatic, title + desc + narr
Number of Topics	28

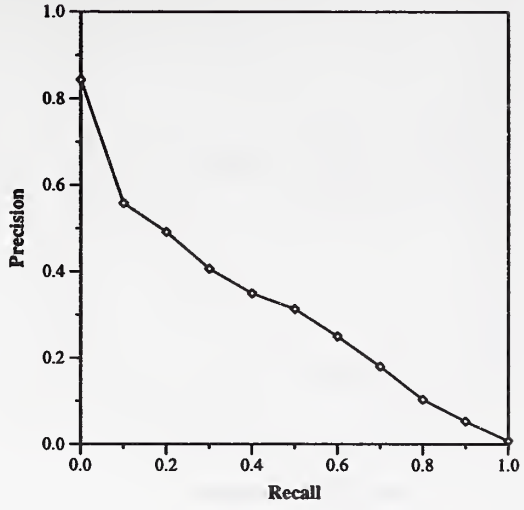
Total number of documents over all topics	
Retrieved:	28000
Relevant:	4098
Rel-ret:	2618

Recall Level Precision Averages	
Recall	Precision
0.00	0.8436
0.10	0.5582
0.20	0.4918
0.30	0.4068
0.40	0.3499
0.50	0.3136
0.60	0.2505
0.70	0.1806
0.80	0.1037
0.90	0.0531
1.00	0.0076

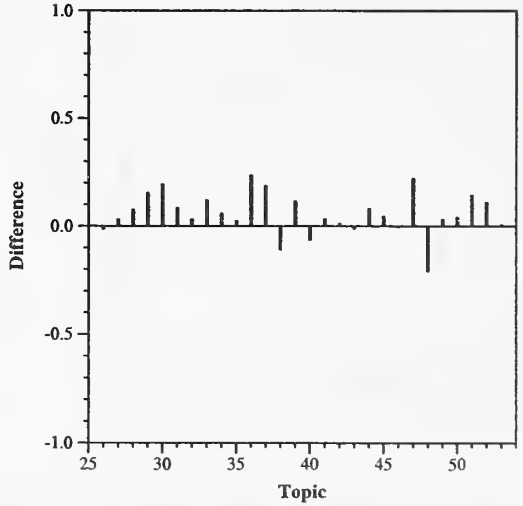
Average precision over all relevant docs	
non-interpolated	0.3009

Document Level Averages	
	Precision
At 5 docs	0.5429
At 10 docs	0.5179
At 15 docs	0.5071
At 20 docs	0.5000
At 30 docs	0.4940
At 100 docs	0.3707
At 200 docs	0.2673
At 500 docs	0.1516
At 1000 docs	0.0935

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3549



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	BKYCL7AF
Run Description	French topics, EFGI documents automatic, title + desc + narr
Number of Topics	28

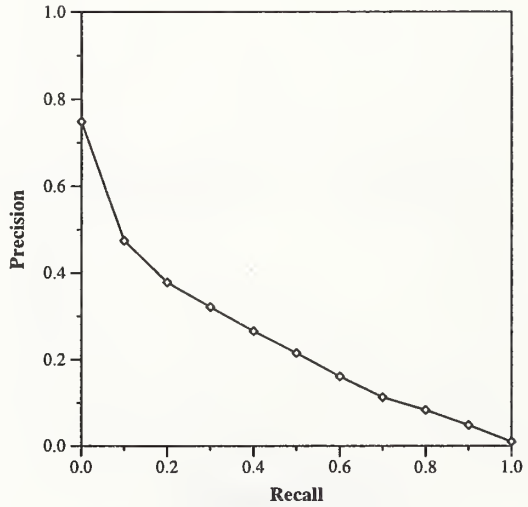
Total number of documents over all topics	
Retrieved:	28000
Relevant:	4098
Rel-ret:	2405

Recall Level Precision Averages	
Recall	Precision
0.00	0.7486
0.10	0.4745
0.20	0.3787
0.30	0.3217
0.40	0.2661
0.50	0.2155
0.60	0.1611
0.70	0.1131
0.80	0.0834
0.90	0.0486
1.00	0.0100

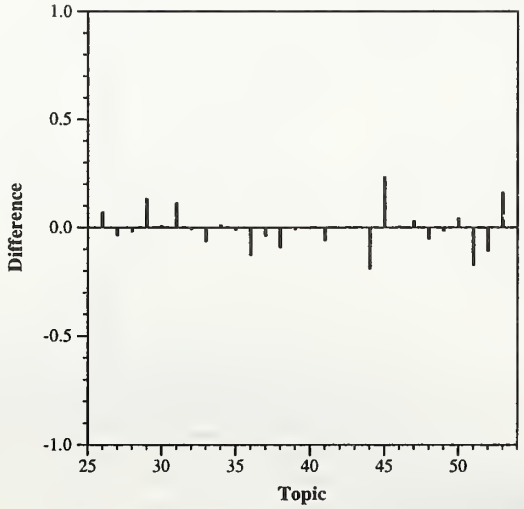
Average precision over all relevant docs	
non-interpolated	0.2369

Document Level Averages	
	Precision
At 5 docs	0.5071
At 10 docs	0.5071
At 15 docs	0.4905
At 20 docs	0.4625
At 30 docs	0.4417
At 100 docs	0.3093
At 200 docs	0.2348
At 500 docs	0.1397
At 1000 docs	0.0859

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2898



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	BKYCL7AG
Run Description	German topics, EFGI documents automatic, title + desc + narr
Number of Topics	28

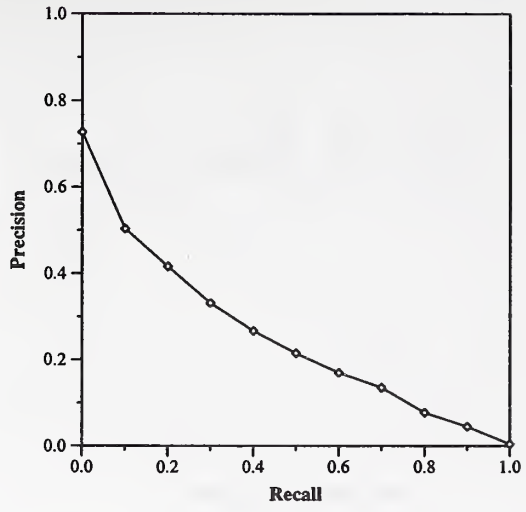
Total number of documents over all topics	
Retrieved:	28000
Relevant:	4098
Rel-ret:	2482

Recall Level Precision Averages	
Recall	Precision
0.00	0.7269
0.10	0.5038
0.20	0.4165
0.30	0.3316
0.40	0.2671
0.50	0.2152
0.60	0.1702
0.70	0.1357
0.80	0.0774
0.90	0.0451
1.00	0.0044

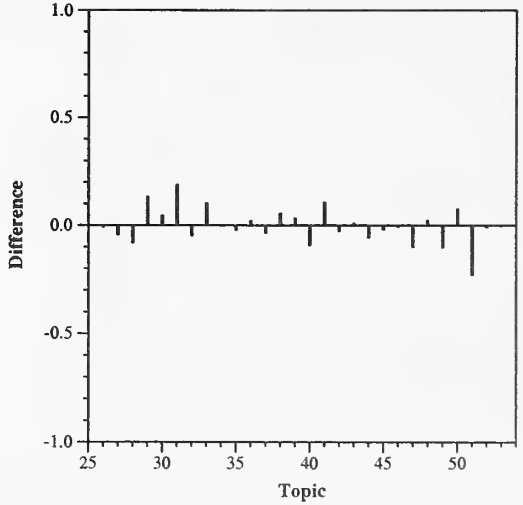
Average precision over all relevant docs	
non-interpolated	0.2406

Document Level Averages	
	Precision
At 5 docs	0.5000
At 10 docs	0.4786
At 15 docs	0.4548
At 20 docs	0.4429
At 30 docs	0.4190
At 100 docs	0.3339
At 200 docs	0.2395
At 500 docs	0.1435
At 1000 docs	0.0886

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2894



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	BKYCL7AI
Run Description	Italian topics, EFGI documents automatic, title + desc + narr
Number of Topics	28

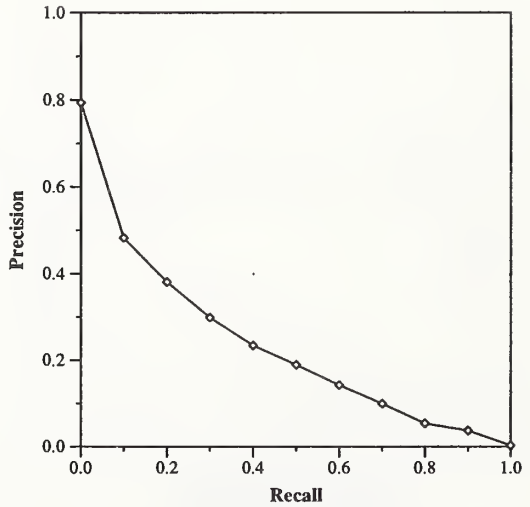
Total number of documents over all topics	
Retrieved:	28000
Relevant:	4098
Rel-ret:	2344

Recall Level Precision Averages	
Recall	Precision
0.00	0.7940
0.10	0.4826
0.20	0.3812
0.30	0.2991
0.40	0.2347
0.50	0.1898
0.60	0.1428
0.70	0.0999
0.80	0.0539
0.90	0.0373
1.00	0.0028

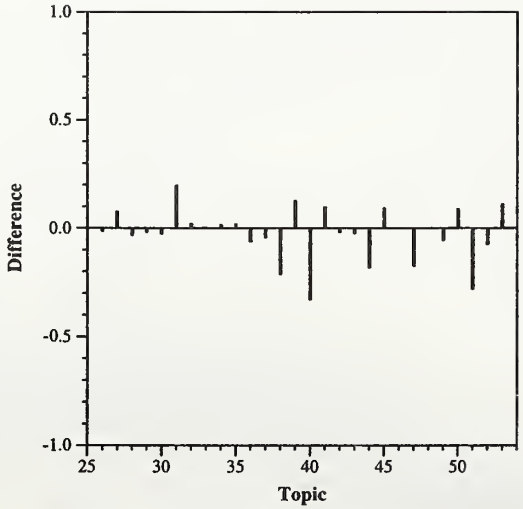
Average precision over all relevant docs	
non-interpolated	0.2184

Document Level Averages	
	Precision
At 5 docs	0.5429
At 10 docs	0.5107
At 15 docs	0.4810
At 20 docs	0.4554
At 30 docs	0.4226
At 100 docs	0.3186
At 200 docs	0.2334
At 500 docs	0.1361
At 1000 docs	0.0837

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2755



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	BKYCL7ME
Run Description	English topics, EFGI documents manual, title + desc + narr
Number of Topics	28

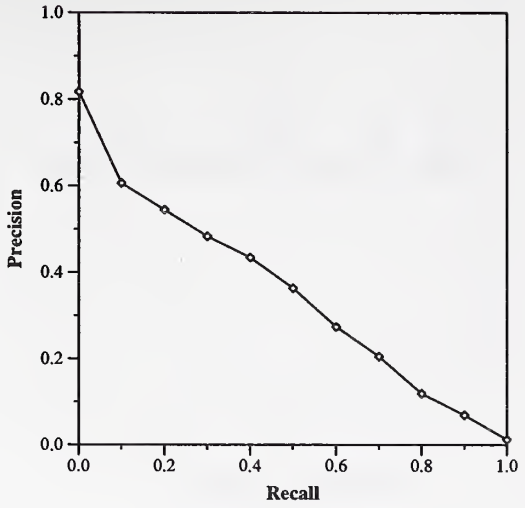
Total number of documents over all topics	
Retrieved:	28000
Relevant:	4098
Rel-ret:	2648

Recall Level Precision Averages	
Recall	Precision
0.00	0.8176
0.10	0.6064
0.20	0.5448
0.30	0.4837
0.40	0.4345
0.50	0.3636
0.60	0.2740
0.70	0.2055
0.80	0.1194
0.90	0.0692
1.00	0.0126

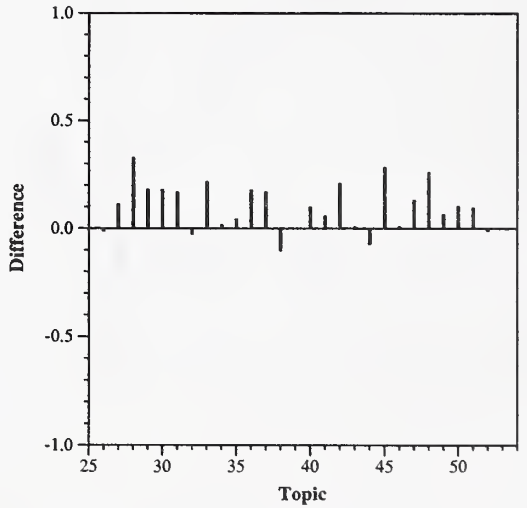
Average precision over all relevant docs	
non-interpolated	0.3390

Document Level Averages	
	Precision
At 5 docs	0.6143
At 10 docs	0.5893
At 15 docs	0.5810
At 20 docs	0.5786
At 30 docs	0.5524
At 100 docs	0.4414
At 200 docs	0.3052
At 500 docs	0.1616
At 1000 docs	0.0946

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3881



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	umdxeof
Run Description	English topics, EFGI documents automatic, title + desc + narr
Number of Topics	28

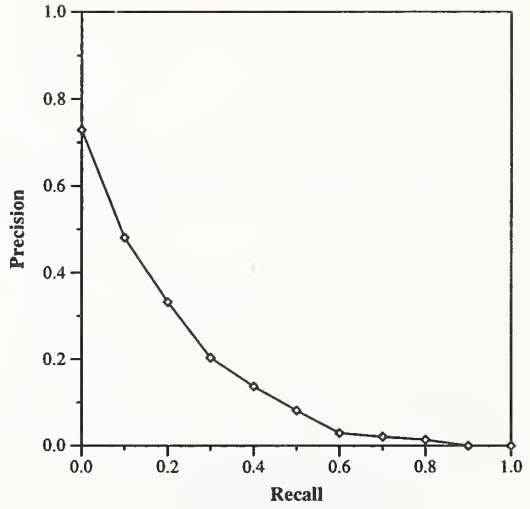
Total number of documents over all topics	
Retrieved:	28000
Relevant:	4098
Rel-ret:	1792

Recall Level Precision Averages	
Recall	Precision
0.00	0.7291
0.10	0.4814
0.20	0.3329
0.30	0.2040
0.40	0.1374
0.50	0.0817
0.60	0.0293
0.70	0.0210
0.80	0.0140
0.90	0.0000
1.00	0.0000

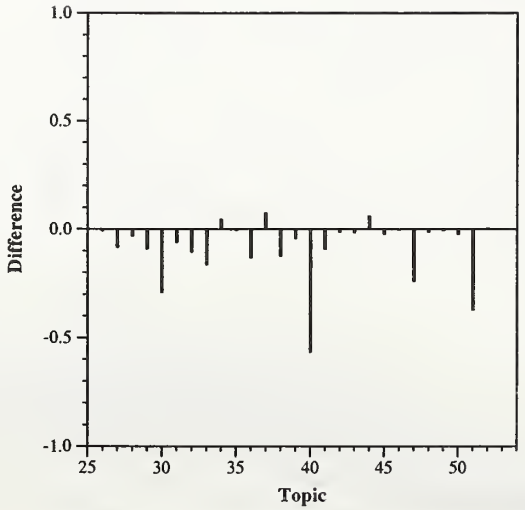
Average precision over all relevant docs	
non-interpolated	0.1610

Document Level Averages	
	Precision
At 5 docs	0.5429
At 10 docs	0.4821
At 15 docs	0.4667
At 20 docs	0.4286
At 30 docs	0.3917
At 100 docs	0.2614
At 200 docs	0.1704
At 500 docs	0.1056
At 1000 docs	0.0640

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2203



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	umdxeot
Run Description	English topics, EFGI documents automatic, title
Number of Topics	28

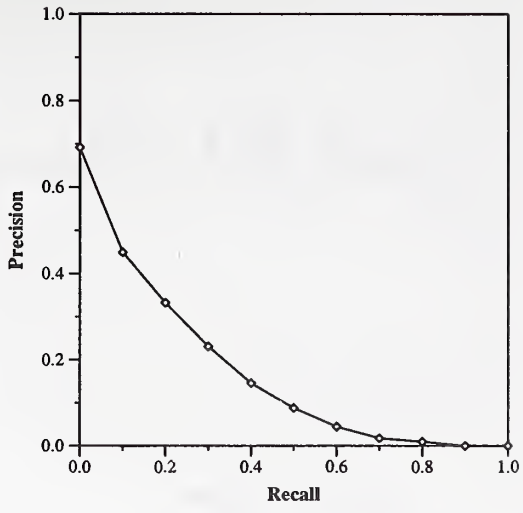
Total number of documents over all topics	
Retrieved:	28000
Relevant:	4098
Rel-ret:	1509

Recall Level Precision Averages	
Recall	Precision
0.00	0.6921
0.10	0.4495
0.20	0.3326
0.30	0.2310
0.40	0.1459
0.50	0.0882
0.60	0.0447
0.70	0.0177
0.80	0.0095
0.90	0.0000
1.00	0.0000

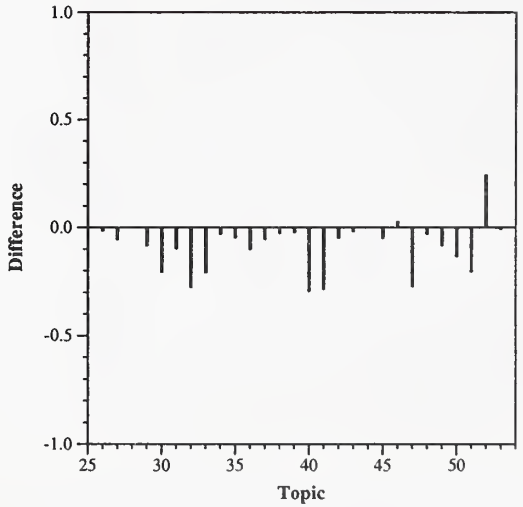
Average precision over all relevant docs	
non-interpolated	0.1588

Document Level Averages	
	Precision
At 5 docs	0.4643
At 10 docs	0.4500
At 15 docs	0.4452
At 20 docs	0.4357
At 30 docs	0.3917
At 100 docs	0.2450
At 200 docs	0.1689
At 500 docs	0.0924
At 1000 docs	0.0539

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2260



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	ceat7e1n
Run Description	English topics, EF documents automatic, desc
Number of Topics	28

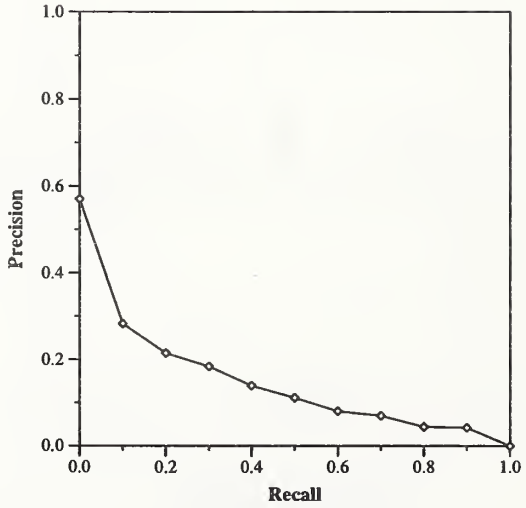
Total number of documents over all topics	
Retrieved:	25108
Relevant:	2680
Rel-ret:	1331

Recall Level Precision Averages	
Recall	Precision
0.00	0.5705
0.10	0.2830
0.20	0.2150
0.30	0.1842
0.40	0.1396
0.50	0.1114
0.60	0.0804
0.70	0.0699
0.80	0.0445
0.90	0.0420
1.00	0.0000

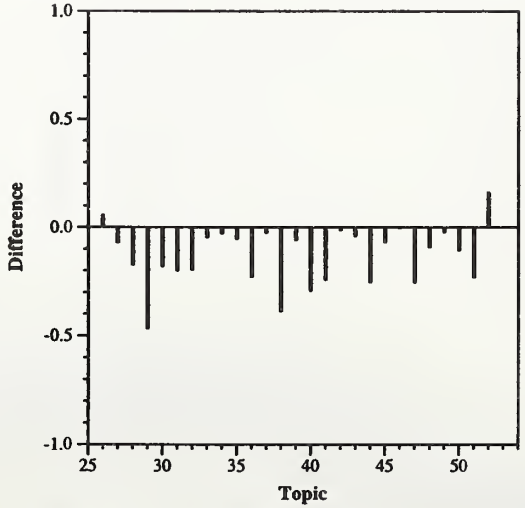
Average precision over all relevant docs	
non-interpolated	0.1271

Document Level Averages	
	Precision
At 5 docs	0.2857
At 10 docs	0.2679
At 15 docs	0.2619
At 20 docs	0.2393
At 30 docs	0.2190
At 100 docs	0.1589
At 200 docs	0.1154
At 500 docs	0.0747
At 1000 docs	0.0475

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1810



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	ceat7e2n
Run Description	English topics, EF documents automatic, desc
Number of Topics	28

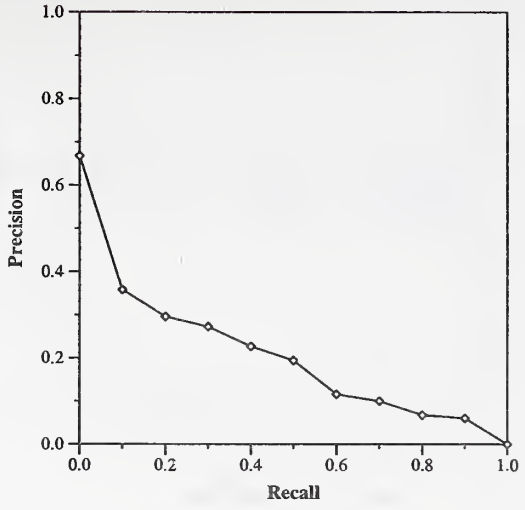
Total number of documents over all topics	
Retrieved:	25108
Relevant:	2680
Rel-ret:	1331

Recall Level Precision Averages	
Recall	Precision
0.00	0.6679
0.10	0.3583
0.20	0.2964
0.30	0.2729
0.40	0.2272
0.50	0.1946
0.60	0.1163
0.70	0.1002
0.80	0.0681
0.90	0.0607
1.00	0.0000

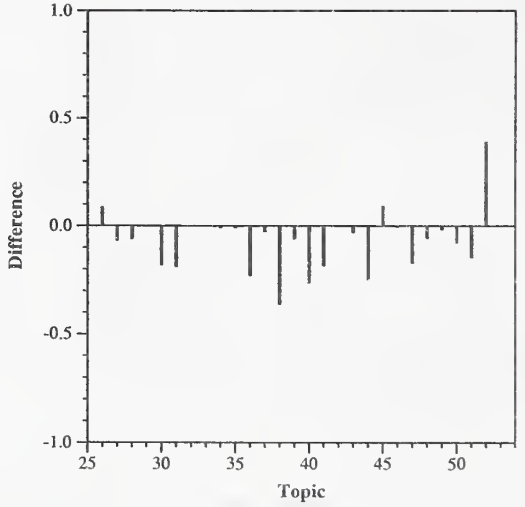
Average precision over all relevant docs	
non-interpolated	0.1879

Document Level Averages	
	Precision
At 5 docs	0.4071
At 10 docs	0.3750
At 15 docs	0.3405
At 20 docs	0.3143
At 30 docs	0.2940
At 100 docs	0.2054
At 200 docs	0.1502
At 500 docs	0.0794
At 1000 docs	0.0475

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2375



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	ibmcl7ef
Run Description	English topics, EF documents automatic, title + desc + narr
Number of Topics	28

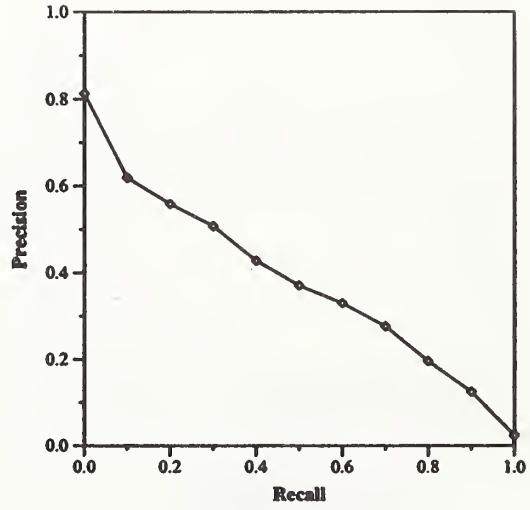
Total number of documents over all topics	
Retrieved:	28000
Relevant:	2680
Rel-ret:	2220

Recall Level Precision Averages	
Recall	Precision
0.00	0.8134
0.10	0.6195
0.20	0.5592
0.30	0.5077
0.40	0.4281
0.50	0.3708
0.60	0.3302
0.70	0.2765
0.80	0.1960
0.90	0.1248
1.00	0.0239

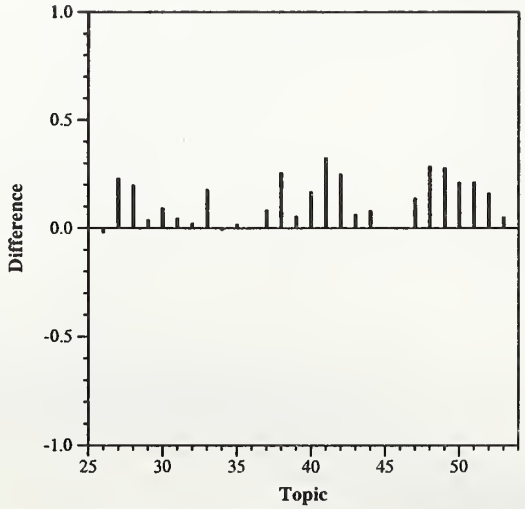
Average precision over all relevant docs	
non-interpolated	0.3732

Document Level Averages	
	Precision
At 5 docs	0.6286
At 10 docs	0.5643
At 15 docs	0.5405
At 20 docs	0.5268
At 30 docs	0.4976
At 100 docs	0.3550
At 200 docs	0.2548
At 500 docs	0.1416
At 1000 docs	0.0793

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3994



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	lanl982
Run Description	English topics, EF documents manual, title + desc + narr
Number of Topics	28

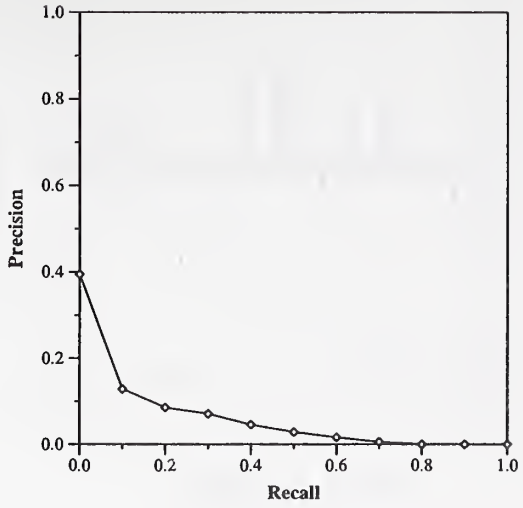
Total number of documents over all topics	
Retrieved:	28000
Relevant:	2680
Rel-ret:	690

Recall Level Precision Averages	
Recall	Precision
0.00	0.3949
0.10	0.1288
0.20	0.0858
0.30	0.0711
0.40	0.0458
0.50	0.0288
0.60	0.0164
0.70	0.0056
0.80	0.0000
0.90	0.0000
1.00	0.0000

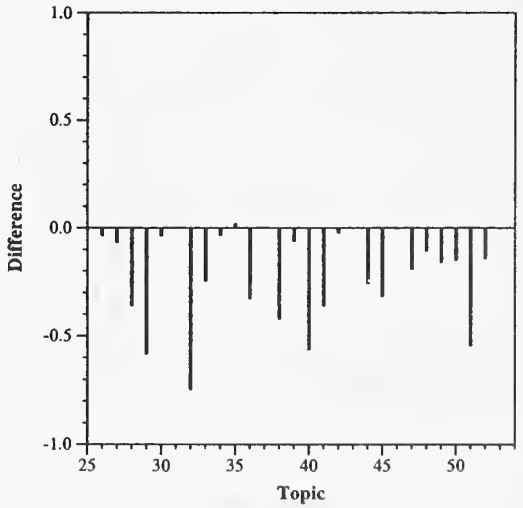
Average precision over all relevant docs	
non-interpolated	0.0487

Document Level Averages	
	Precision
At 5 docs	0.1643
At 10 docs	0.1536
At 15 docs	0.1381
At 20 docs	0.1446
At 30 docs	0.1393
At 100 docs	0.1175
At 200 docs	0.0771
At 500 docs	0.0414
At 1000 docs	0.0246

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0947



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	TW1E2EF
Run Description	English topics, EF documents automatic, title
Number of Topics	28

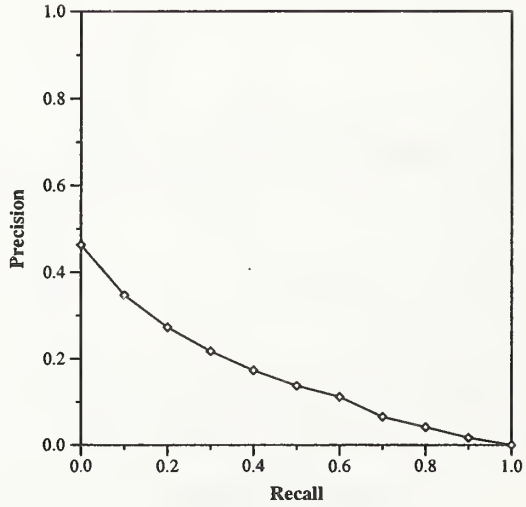
Total number of documents over all topics	
Retrieved:	28000
Relevant:	2680
Rel-ret:	1045

Recall Level Precision Averages	
Recall	Precision
0.00	0.4632
0.10	0.3471
0.20	0.2729
0.30	0.2174
0.40	0.1731
0.50	0.1376
0.60	0.1119
0.70	0.0654
0.80	0.0416
0.90	0.0171
1.00	0.0000

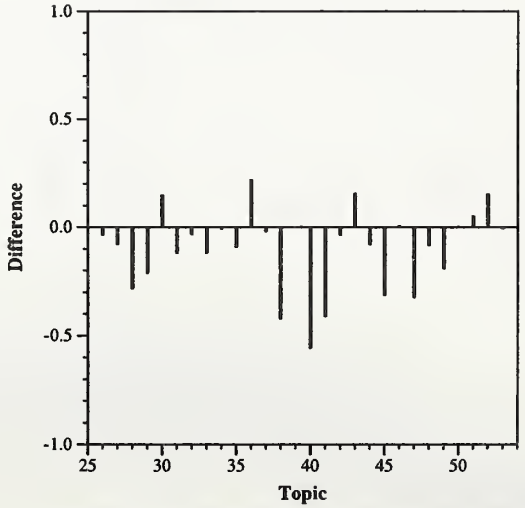
Average precision over all relevant docs	
non-interpolated	0.1570

Document Level Averages	
	Precision
At 5 docs	0.2929
At 10 docs	0.2964
At 15 docs	0.3000
At 20 docs	0.2804
At 30 docs	0.2655
At 100 docs	0.1779
At 200 docs	0.1138
At 500 docs	0.0617
At 1000 docs	0.0373

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1927



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	tno7eef
Run Description	English topics, EF documents automatic, title + desc + narr
Number of Topics	28

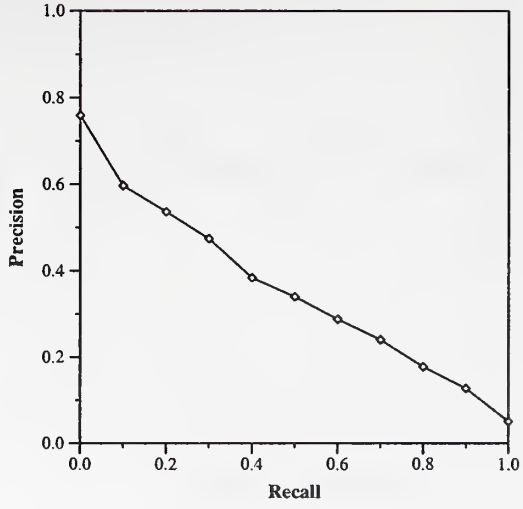
Total number of documents over all topics	
Retrieved:	28000
Relevant:	2680
Rel-ret:	2038

Recall Level Precision Averages	
Recall	Precision
0.00	0.7595
0.10	0.5971
0.20	0.5368
0.30	0.4745
0.40	0.3841
0.50	0.3404
0.60	0.2883
0.70	0.2408
0.80	0.1781
0.90	0.1279
1.00	0.0510

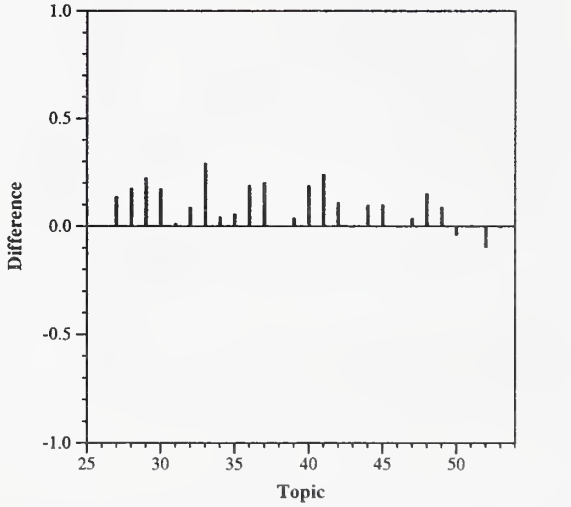
Average precision over all relevant docs	
non-interpolated	0.3404

Document Level Averages	
	Precision
At 5 docs	0.5286
At 10 docs	0.5071
At 15 docs	0.5024
At 20 docs	0.4679
At 30 docs	0.4440
At 100 docs	0.3193
At 200 docs	0.2282
At 500 docs	0.1266
At 1000 docs	0.0728

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3583



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	RaliDicE2EF
Run Description	English topics, EF documents automatic, title + desc + narr
Number of Topics	28

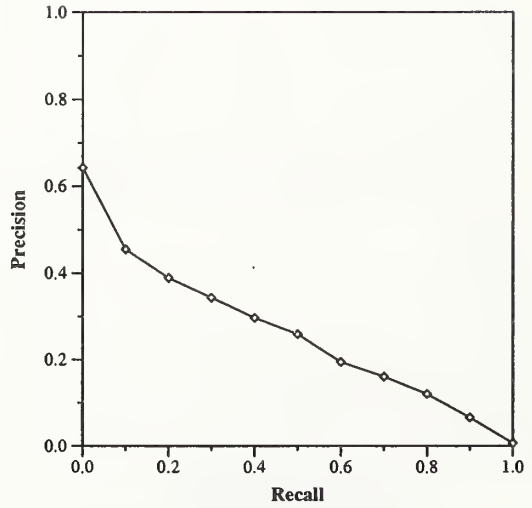
Total number of documents over all topics	
Retrieved:	28000
Relevant:	2680
Rel-ret:	1798

Recall Level Precision Averages	
Recall	Precision
0.00	0.6432
0.10	0.4551
0.20	0.3892
0.30	0.3436
0.40	0.2968
0.50	0.2589
0.60	0.1945
0.70	0.1608
0.80	0.1204
0.90	0.0659
1.00	0.0063

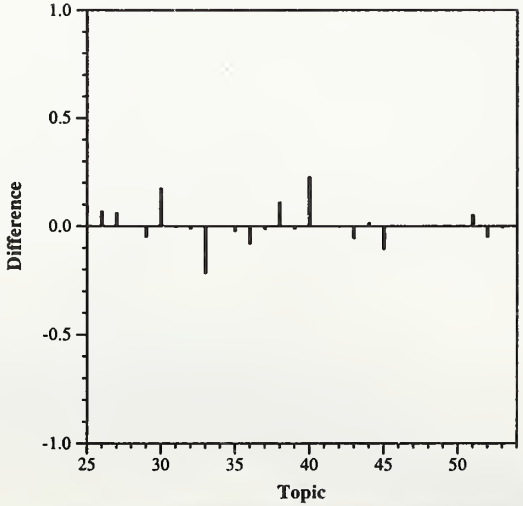
Average precision over all relevant docs	
non-interpolated	0.2554

Document Level Averages	
	Precision
At 5 docs	0.4571
At 10 docs	0.4179
At 15 docs	0.3905
At 20 docs	0.3786
At 30 docs	0.3560
At 100 docs	0.2446
At 200 docs	0.1846
At 500 docs	0.1061
At 1000 docs	0.0642

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2955



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	RaliDicSDAef
Run Description	English topics, EF documents automatic, title + desc
Number of Topics	28

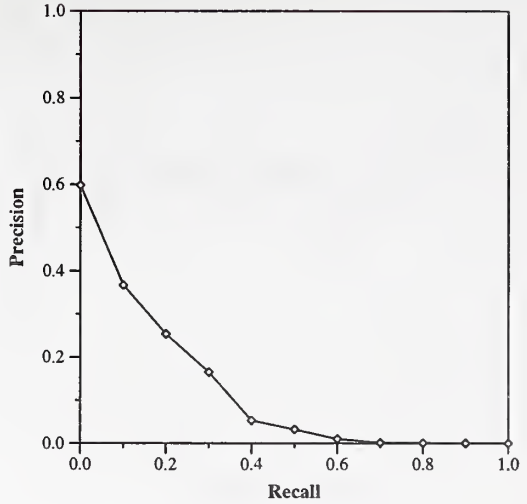
Total number of documents over all topics	
Retrieved:	28000
Relevant:	2680
Rel-ret:	792

Recall Level Precision Averages	
Recall	Precision
0.00	0.5983
0.10	0.3673
0.20	0.2542
0.30	0.1655
0.40	0.0531
0.50	0.0322
0.60	0.0100
0.70	0.0011
0.80	0.0005
0.90	0.0001
1.00	0.0001

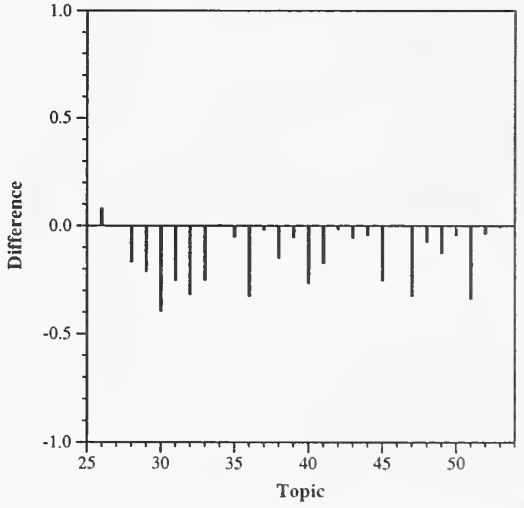
Average precision over all relevant docs	
non-interpolated	0.1150

Document Level Averages	
	Precision
At 5 docs	0.3714
At 10 docs	0.3179
At 15 docs	0.3167
At 20 docs	0.3071
At 30 docs	0.2476
At 100 docs	0.1525
At 200 docs	0.1018
At 500 docs	0.0514
At 1000 docs	0.0283

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1698



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	BKYCL7AEF
Run Description	English topics, EF documents automatic, title + desc + narr
Number of Topics	28

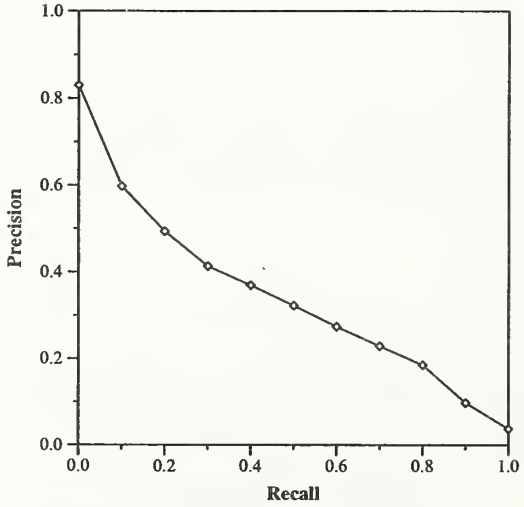
Total number of documents over all topics	
Retrieved:	28000
Relevant:	2680
Rel-ret:	2007

Recall Level Precision Averages	
Recall	Precision
0.00	0.8299
0.10	0.5977
0.20	0.4940
0.30	0.4136
0.40	0.3699
0.50	0.3227
0.60	0.2742
0.70	0.2292
0.80	0.1854
0.90	0.0979
1.00	0.0375

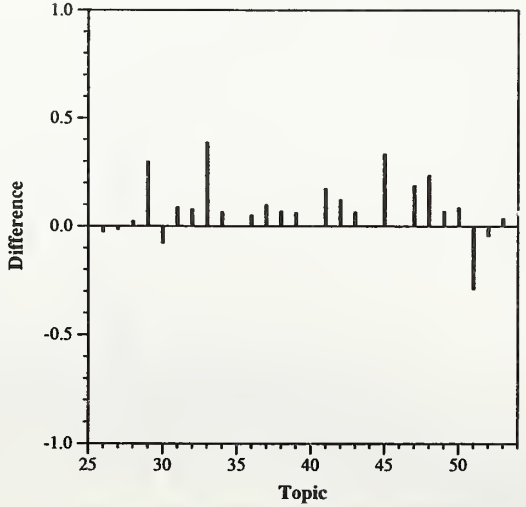
Average precision over all relevant docs	
non-interpolated	0.3261

Document Level Averages	
	Precision
At 5 docs	0.5857
At 10 docs	0.5536
At 15 docs	0.5333
At 20 docs	0.5107
At 30 docs	0.4560
At 100 docs	0.3121
At 200 docs	0.2229
At 500 docs	0.1246
At 1000 docs	0.0717

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3540



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	BKYCL7MEF
Run Description	English topics, EF documents manual, title + desc + narr
Number of Topics	28

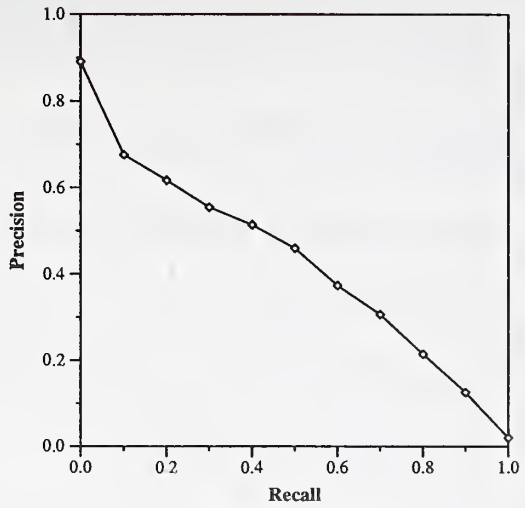
Total number of documents over all topics	
Retrieved:	28000
Relevant:	2680
Rel-ret:	2106

Recall Level Precision Averages	
Recall	Precision
0.00	0.8908
0.10	0.6763
0.20	0.6172
0.30	0.5548
0.40	0.5142
0.50	0.4597
0.60	0.3738
0.70	0.3067
0.80	0.2145
0.90	0.1259
1.00	0.0201

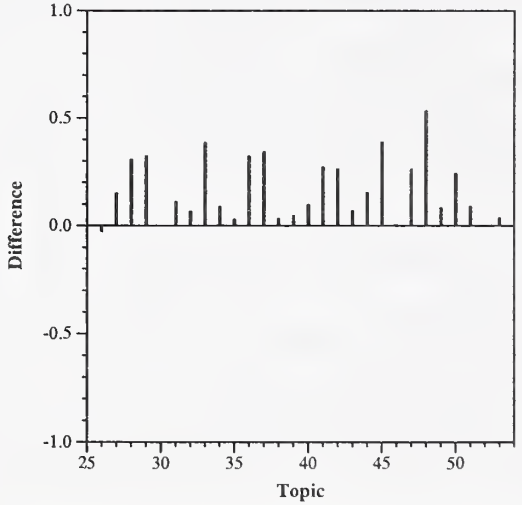
Average precision over all relevant docs	
non-interpolated	0.4185

Document Level Averages	
	Precision
At 5 docs	0.6929
At 10 docs	0.6393
At 15 docs	0.6000
At 20 docs	0.5875
At 30 docs	0.5464
At 100 docs	0.3807
At 200 docs	0.2564
At 500 docs	0.1339
At 1000 docs	0.0752

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4306



Recall-Precision Curve



Difference from Median in Average Precision per Topic

TREC-7 Adaptive Filtering Results, F1 Measure

Topic	A	B	C	D	E	F	G	H	I	J	K	L	M	Maximal
1	1	1	-57	-17	-103	14	260	-19	9	-8	-535	-84	-5	1032
2	-13	-13	-53	7	-13	58	-23	-19	-17	-15	-645	-2067	-2049	1377
3	13	7	-49	-26	-22	-15	-5	-22	-16	18	-484	-24804	-25043	660
4	-15	-18	-436	-4	-139	34	34	-11	-12	-50	-477	-4264	-4274	282
5	27	27	-108	-4	-47	114	59	62	53	-6	-827	-70	-534	204
6	-20	-21	-185	-2	-42	179	148	-15	31	79	-1708	73	71	810
7	110	110	-1133	-18	-49	-34	52	16	54	32	-1096	-35	-54	933
8	-38	-38	-77	-28	-636	6	4	-16	-16	-446	-529	-14	-96	198
9	34	34	-48	-26	-142	47	-35	-26	-24	21	-407	-9	-6	321
10	233	233	32	-4	-6	207	179	148	278	118	-897	-2560	-2563	795
11	96	96	-552	-4	-60	-39	-42	-20	129	-14	-1187	-1	-28	1281
12	250	250	103	-2	-16	-124	-124	238	465	5	-826	6	15	1872
13	172	172	70	-41	-18	175	159	12	102	249	-123	-2	-4	258
14	-11	-11	-155	-17	-85	5	2	-19	20	-14	-1667	-22	-330	348
15	-84	-6	-539	-1	-109	-243	-208	-60	-23	-164	-1475	-7	-42	321
16	-5	-8	-29	-5	-49	44	31	4	24	-55	-581	-61	-183	426
17	135	135	55	7	10	-87	-87	77	239	11	-151	0	-113	672
18	9	9	-351	-13	-1650	-7	7	-17	-23	-169	-652	0	-1860	390
19	-3	0	-368	-17	-95	-458	69	10	-6	56	-199	17	-53	807
20	93	93	142	-8	45	19	-168	161	95	204	-146	3	4	474
21	-13	-13	29	2	-18	55	44	51	53	37	-93	9	6	87
22	698	698	225	0	483	1070	964	931	980	148	-844	-2	-2	3714
23	360	360	144	8	-67	379	281	156	198	137	266	-2	-2	711
24	67	67	-93	-9	-47	132	-63	-4	1	7	-904	-15	-617	900
25	-62	-37	-68	-6	-90	14	2	-26	-33	-14	-163	-6	-10	195
26	-9	-19	-6	-19	-45	18	9	-21	-34	-20	-12	0	3	183
27	5	6	0	-20	-1995	15	9	0	-8	13	-55	0	0	60
28	-41	-32	-2	-42	-58	6	6	-62	-79	-82	-222	0	0	198
29	-34	-34	-28	-24	-222	6	6	-116	-116	-110	-138	0	0	21
30	-49	-28	-16	-80	-1249	-195	-195	-69	-69	-105	-101	-25	-71	3
31	-51	-25	-10	-28	-71	-3	3	-60	-60	-20	-116	-11	-11	3
32	-30	-13	-12	-86	-2000	9	5	-21	-83	-16	-79	0	0	18
33	-61	-23	-10	-20	-45	-10	7	-41	-51	-81	-384	1	-3	39
34	-38	-41	0	-71	-293	-200	-200	-23	-23	-16	-19	0	0	6
35	-77	-45	0	-27	-19	3	3	-47	-95	-39	-130	0	0	3
36	-30	-28	-10	-39	-65	8	12	-42	-59	3	-133	-38	-95	30
37	-90	-40	6	-32	-1189	17	5	-42	-52	-50	-114	-2	-2	21
38	34	34	-44	-1	-120	-928	-928	63	28	69	-1864	0	-1	828
39	-37	-38	-2	-165	-29	-180	-180	-47	-47	-19	-10	0	0	12
40	33	33	-164	-21	-2000	-824	-824	125	67	105	-1599	0	-2	708
41	-34	-30	-1	-57	-1980	24	26	-33	-61	-75	-527	0	-6	222
42	26	26	-79	-64	-1586	-90	-221	4	5	-476	-294	4	4	453
43	-23	-21	-49	-17	-146	31	-1379	-8	-1	-153	-552	0	-2	306
44	-9	-12	-118	-77	-79	111	37	-17	-30	-40	-746	0	-26	456
45	-21	-19	-28	-13	-1995	-25	-70	-12	-12	-73	-754	-10	-69	156
46	2	-14	-105	-35	-97	68	71	-22	9	27	-479	0	0	222
47	-42	-42	-98	-44	-32	-16	21	5	-15	-5	-868	0	-13	267
48	-26	-24	-4	-38	-774	1	-4	-43	-50	-29	-161	-4	-4	90
49	-15	-15	5	-15	-27	40	33	34	9	4	-208	23	23	165
50	-30	-30	0	-44	-350	3	3	-34	-34	-137	-14	0	0	18

KEY

A	CLARITafF1a	B	CLARITafF1b	C	IAHKaf11	D	IAHKaf12
E	INQ510	F	Mer7AGbF1	G	Mer7ARbF1	H	ok7ff12
I	ok7ff13	J	pirc8FA1	K	sigmaTrec7F1	L	TNOAF103
M	TNOAF102						

TREC-7 Adaptive Filtering Results, F3 Measure

Topic	A	B	C	D	E	F	G	H	I	J	K	L	Maximal
1	128	128	-1	-6	-34	279	363	7	57	482	-73	50	1376
2	140	140	4	11	21	158	11	-7	-6	131	-82	-1224	1836
3	139	139	4	-8	9	110	-135	61	39	215	23	-24408	880
4	71	71	-8	3	-37	77	69	33	82	9	-269	-3944	376
5	50	50	0	3	29	159	132	115	115	-92	-317	-504	272
6	166	166	32	4	4	260	140	503	286	399	-611	286	1080
7	398	398	-9	-4	-2	299	320	313	440	206	-396	-34	1244
8	-19	-19	-10	-9	-303	9	7	-8	-8	-225	-260	-96	264
9	152	152	-7	-8	-31	152	116	-5	3	119	-231	24	428
10	607	607	24	3	22	323	126	226	503	283	97	-1968	1060
11	581	581	28	3	-10	759	759	414	548	581	3	-8	1708
12	901	901	-3	4	12	868	868	636	1041	189	256	40	2496
13	283	283	30	-18	-9	239	116	36	193	303	55	-4	344
14	5	5	-8	-6	-25	61	41	23	48	-19	-722	-300	464
15	-67	-67	7	7	-12	-369	-239	27	14	16	-440	138	428
16	28	28	-1	0	-12	186	98	32	99	38	-310	-178	568
17	333	333	0	11	35	64	64	264	446	234	194	22	896
18	34	34	18	-4	-650	-30	-369	8	1	-678	-476	-1570	520
19	20	18	0	-6	25	-55	84	88	17	160	29	152	1076
20	328	328	16	6	165	297	12	385	201	456	193	14	632
21	9	9	15	6	11	85	82	88	90	19	-29	16	116
22	2078	2078	2	5	749	2060	1962	2049	2244	1270	1055	-2	4952
23	775	775	0	14	-11	235	356	263	402	373	478	-2	948
24	387	387	0	-2	4	330	218	201	54	272	-288	-312	1200
25	21	21	3	2	-30	72	61	-3	7	-9	-51	30	260
26	24	-3	-1	-7	-5	24	28	-1	8	12	24	8	244
27	21	21	0	-5	-995	20	17	15	11	22	-15	0	80
28	-18	-16	-1	-21	-14	8	16	-26	-27	-32	-102	0	264
29	-17	-17	-3	-12	-106	13	8	-58	-58	-164	-63	0	28
30	-22	-14	-3	-40	-622	-95	-95	-32	-32	-61	-48	-66	4
31	-24	-15	-3	-14	-33	1	4	-30	-30	-15	-58	-6	4
32	-9	-9	0	-43	-1000	12	10	-3	-34	-34	-37	0	24
33	-10	-10	0	-10	-10	17	11	-10	-13	-46	-154	2	52
34	-14	-18	0	-33	-144	-100	-100	-9	-9	-7	-7	0	8
35	-31	-21	0	-11	-7	4	4	-21	-45	-66	-65	0	4
36	-7	-13	7	-17	-20	17	16	-11	-12	13	-55	-80	40
37	-46	-12	0	-11	-577	26	15	-12	-16	-22	-52	-2	28
38	108	108	0	2	45	71	71	199	328	195	-779	14	1104
39	-11	-19	0	-80	-7	-80	-80	-16	-16	-6	5	0	16
40	442	442	312	-8	-1000	153	153	220	373	298	-366	-2	944
41	-28	-10	4	-26	-980	32	43	-5	-18	-22	-208	-6	296
42	82	82	40	-27	-613	-323	-125	49	60	-121	-164	14	604
43	26	7	-4	-6	-23	-498	-638	18	32	-68	-275	-2	408
44	34	34	-31	-31	-22	132	124	-7	0	-23	-159	-26	608
45	-10	-9	0	-4	-995	-565	-211	-6	-6	-41	-318	-64	208
46	143	143	31	-10	-31	170	159	78	155	163	-232	0	296
47	-20	-27	-46	-17	4	-174	32	29	33	18	-441	-8	356
48	-13	-7	-1	-19	-372	-4	-3	-14	-10	-16	-73	-4	120
49	-18	-18	51	-5	4	91	84	86	62	38	-12	98	220
50	-15	-15	0	-22	-175	4	-581	-17	-17	-109	-7	0	24

KEY

A	CLARITafF3a	B	CLARITafF3b	C	IAHKaf31	D	IAHKaf32
E	INQ511	F	Mer7AGbF3	G	Mer7ARbF3	H	ok7ff32
I	ok7ff33	J	pirc8FA3	K	sigmaTrec7F3	L	TNOAF102

TREC-7 Batch Filtering Results, F1 Measure

Topic	att98fb5	CLARITbfF1	IAHKbf11	MerBF1	nttd7bf1	pirc8FB1	Maximal
1	87	27	14	-147	-91	101	666
2	-8	-7	1	-185	5	11	597
3	136	44	17	-54	220	204	480
4	-1	2	-14	3	12	4	192
5	47	5	18	64	50	38	120
6	193	58	4	98	166	183	597
7	25	0	-4	-140	-14	3	585
8	-2	-8	-14	-2	2	6	114
9	0	40	38	18	3	0	237
10	89	153	46	132	363	361	540
11	130	32	33	148	117	72	906
12	86	48	36	65	-90	419	1332
13	36	200	28	104	0	180	234
14	-11	-10	-34	-36	12	6	219
15	-38	-53	19	-46	-143	1	144
16	1	6	-4	12	26	2	342
17	110	32	20	63	5	108	405
18	0	-14	0	-43	-56	0	204
19	0	33	9	85	83	0	630
20	17	96	6	25	-10	49	333
21	2	-17	-7	0	0	6	24
22	786	39	197	-918	655	745	2529
23	207	314	168	197	192	353	480
24	78	15	49	103	70	117	567
25	-5	-12	-13	-22	11	25	126
26	0	4	0	-3	0	3	162
27	0	-9	-7	0	0	0	39
28	0	-17	0	0	0	0	177
29	-2	-34	-14	-2	0	0	9
30	0	-13	-2	-195	0	0	3
31	0	-32	-16	0	0	0	0
32	0	-22	0	-190	0	0	18
33	0	-31	-6	0	0	0	30
34	0	-12	0	-190	0	0	6
35	0	-33	1	-195	0	0	3
36	0	-28	-12	0	0	0	18
37	0	-11	0	-180	0	-4	21
38	50	3	3	-271	44	13	699
39	0	-17	9	-180	0	-4	12
40	3	33	52	-392	-61	17	486
41	0	-2	3	-8	0	0	186
42	-116	-35	-119	-206	-139	-129	255
43	0	11	-1	1	0	1	231
44	75	35	-2	93	63	43	372
45	-175	-21	-104	-1073	-167	-178	3
46	0	6	14	14	-2	28	144
47	3	-6	2	-105	2	5	189
48	-4	-9	-4	-8	-2	-4	72
49	2	-1	30	-9	9	8	120
50	0	-22	0	-10	0	0	3

TREC-7 Batch Filtering Results, F3 Measure

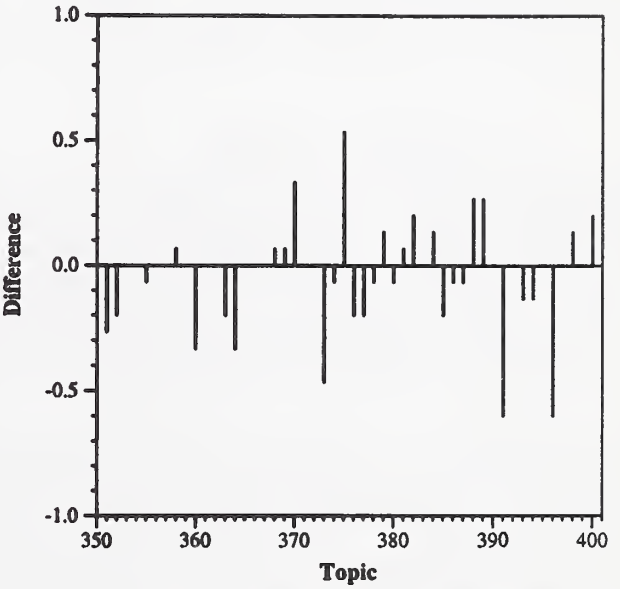
Topic	att98fb6	CLARITbfF3	IAHKbf32	MerBF3	nttd7bf3	pirc8FB3	Maximal
1	386	68	16	262	288	391	888
2	-31	-2	20	-1300	193	37	796
3	316	80	35	95	359	333	640
4	24	15	-5	21	63	36	256
5	79	32	24	92	80	81	160
6	385	201	26	206	360	364	796
7	130	91	1	-1251	235	231	780
8	1	-4	-1	-10	21	7	152
9	0	118	-6	24	8	19	316
10	152	368	59	-97	576	576	720
11	421	280	176	235	457	413	1208
12	213	257	113	165	910	788	1776
13	48	284	128	182	0	260	312
14	47	0	6	-355	79	76	292
15	-19	-110	-10	-69	-39	6	192
16	11	22	12	90	43	18	456
17	247	190	40	-1221	300	276	540
18	-21	-2	-1	-330	-47	-14	272
19	4	61	51	208	119	3	840
20	46	221	12	-162	5	133	444
21	15	-13	5	24	21	19	32
22	1807	477	91	1201	1655	1528	3372
23	391	469	156	-281	446	523	640
24	311	53	46	-622	353	346	756
25	42	19	-1	7	22	58	168
26	0	29	-5	6	0	8	216
27	0	4	-1	0	0	0	52
28	0	-1	0	-1	0	4	236
29	-1	-17	-10	-1	0	0	12
30	0	-4	-1	-95	0	0	4
31	0	-16	-5	0	0	0	0
32	0	-7	0	-90	0	0	24
33	0	-11	-3	0	0	0	40
34	0	-1	0	-90	0	0	8
35	0	-23	0	-95	0	0	4
36	0	-6	-5	0	0	0	24
37	0	2	0	-80	0	-10	28
38	196	65	16	26	262	95	932
39	0	-1	0	-80	0	-7	16
40	33	128	117	164	262	74	648
41	0	4	0	1	0	0	248
42	-33	-17	4	-161	-44	-68	340
43	0	53	2	8	0	19	308
44	146	112	1	29	89	148	496
45	-143	-8	-24	-1277	-123	-136	4
46	18	34	9	92	-1	67	192
47	7	2	2	-5	25	14	252
48	-5	-4	-3	-4	-1	-8	96
49	21	14	24	38	67	51	160
50	0	-11	0	-5	0	0	4

TREC-7 Routing Filtering Results, Average Precision

Topic	AntRout1	AntRout2	arc98cs	att98fr4	att98fr5	MerRou	nttd7rt1	nttd7rt2	pirc8R1	pirc8R2
1	0.293	0.487	0.059	0.516	0.616	0.409	0.655	0.655	0.180	0.633
2	0.083	0.093	0.029	0.041	0.099	0.048	0.440	0.440	0.051	0.167
3	0.340	0.542	0.090	0.597	0.735	0.342	0.664	0.664	0.292	0.704
4	0.370	0.263	0.084	0.282	0.422	0.205	0.563	0.563	0.342	0.376
5	0.510	0.735	0.149	0.627	0.781	0.673	0.701	0.701	0.211	0.842
6	0.326	0.316	0.071	0.582	0.589	0.481	0.675	0.675	0.506	0.553
7	0.261	0.103	0.086	0.377	0.453	0.361	0.521	0.521	0.269	0.422
8	0.000	0.049	0.000	0.074	0.213	0.067	0.214	0.214	0.005	0.154
9	0.255	0.079	0.031	0.555	0.561	0.610	0.455	0.455	0.490	0.536
10	0.816	0.685	0.079	0.522	0.852	0.586	0.918	0.918	0.431	0.855
11	0.153	0.247	0.019	0.433	0.402	0.438	0.409	0.409	0.369	0.363
12	0.195	0.051	0.006	0.297	0.619	0.314	0.660	0.660	0.286	0.510
13	0.799	0.710	0.002	0.679	0.991	0.735	0.979	0.979	1.000	0.996
14	0.056	0.139	0.010	0.307	0.394	0.242	0.479	0.479	0.063	0.341
15	0.072	0.125	0.024	0.099	0.130	0.124	0.161	0.161	0.117	0.157
16	0.080	0.065	0.012	0.299	0.381	0.281	0.517	0.517	0.232	0.362
17	0.468	0.370	0.001	0.568	0.625	0.536	0.702	0.702	0.351	0.601
18	0.002	0.082	0.003	0.079	0.101	0.139	0.103	0.103	0.104	0.068
19	0.241	0.236	0.089	0.387	0.447	0.346	0.602	0.602	0.251	0.377
20	0.705	0.755	0.387	0.185	0.821	0.298	0.862	0.862	0.854	0.684
21	0.465	0.238	0.131	0.495	0.479	0.429	0.378	0.378	0.426	0.550
22	0.299	0.323	0.239	0.477	0.538	0.464	0.503	0.503	0.343	0.487
23	0.617	0.714	0.026	0.758	0.923	0.625	0.851	0.851	0.519	0.948
24	0.321	0.350	0.055	0.483	0.544	0.478	0.606	0.606	0.240	0.517
25	0.175	0.161	0.004	0.375	0.443	0.413	0.426	0.426	0.307	0.445
26	0.238	0.174	0.073	0.242	0.347	0.239	0.388	0.388	0.225	0.394
27	0.159	0.154	0.095	0.049	0.260	0.024	0.503	0.503	0.504	0.629
28	0.004	0.037	0.024	0.211	0.333	0.117	0.538	0.538	0.173	0.329
29	0.000	0.000	0.010	0.003	0.007	0.001	0.339	0.339	0.006	0.011
30	0.500	1.000	0.167	0.000	0.500	0.333	0.200	0.200	0.001	0.001
31	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
32	0.067	0.183	0.094	0.000	0.036	0.050	0.545	0.545	0.000	0.000
33	0.143	0.028	0.001	0.015	0.011	0.013	0.375	0.375	0.192	0.029
34	0.500	0.500	0.050	0.000	1.000	0.016	0.504	0.504	0.000	0.000
35	1.000	1.000	1.000	0.000	1.000	0.333	1.000	1.000	0.002	0.002
36	0.039	0.595	0.182	0.081	0.184	0.417	0.756	0.756	0.662	0.423
37	0.143	0.169	0.151	0.000	0.137	0.134	0.617	0.617	0.000	0.000
38	0.149	0.216	0.001	0.315	0.456	0.154	0.491	0.491	0.308	0.304
39	0.750	0.750	0.001	0.000	0.644	0.887	0.779	0.779	0.009	0.009
40	0.332	0.218	0.272	0.546	0.516	0.446	0.458	0.458	0.428	0.506
41	0.000	0.000	0.037	0.241	0.173	0.116	0.354	0.354	0.064	0.130
42	0.097	0.046	0.019	0.069	0.073	0.057	0.090	0.090	0.112	0.073
43	0.305	0.239	0.000	0.298	0.343	0.322	0.401	0.401	0.298	0.374
44	0.164	0.181	0.195	0.509	0.500	0.451	0.718	0.718	0.122	0.519
45	0.000	0.002	0.500	0.031	0.019	0.040	1.000	1.000	1.000	0.011
46	0.251	0.334	0.229	0.648	0.587	0.651	0.610	0.610	0.492	0.669
47	0.003	0.039	0.070	0.145	0.288	0.168	0.341	0.341	0.080	0.264
48	0.000	0.000	0.153	0.023	0.030	0.029	0.196	0.196	0.150	0.075
49	0.257	0.265	0.079	0.248	0.364	0.256	0.447	0.447	0.402	0.395
50	0.016	0.091	0.000	0.001	0.000	0.000	0.003	0.003	0.067	0.002
Mean	0.260	0.283	0.102	0.275	0.419	0.298	0.514	0.514	0.271	0.356

Summary Statistics	
Run Number	acsys7hp
Number of Topics	50
Total number of documents over all topics	
Retrieved:	648
Relevant:	4674
Rel-ret:	384

Means over 50 topics	
Precision at 15 Docs:	0.5120
Relative-Precision at 15 Docs:	0.5205
Unranked-Avg-Precision at 15 Docs:	0.0886



Precision(15) Difference from Median per Topic

Evaluation Measures

Precision at 15: The percentage of documents retrieved in the top fifteen that are relevant. If fewer than 15 documents are retrieved, then all missing documents are assumed to be non-relevant. Precision considers each retrieved relevant document to be equally important, no matter if is retrieved for a query with 500 relevant documents or a query with two relevant documents.

Relative Precision at 15: The precision after fifteen documents relative to the maximum precision possible at that point. For example, if there are 5 relevant documents and 3 of those are retrieved, then Precision at 15 is .2, but the Relative-Precision is .2/.33 or .06. Relative-Precision considers each query to be equally important. Thus a query with only 5 relevant documents has the same maximum score of 1.0 as a query with fifteen or more relevant documents.

Unranked-Average Precision at 15: Similar to the standard TREC "average precision" measure, with all retrieved relevant documents getting a precision value of the retrieved set (i.e., $r/15$ where r is the number of relevant documents retrieved). All non-retrieved relevant documents get a precision value of 0. This measure is actually directed at unranked evaluation where the size of the retrieved set is under the control of the user, and is not completely appropriate for the High Precision track. It is included to gain more operational experience with the measure, and to see if it offers insights into the results.

Summary Statistics	
Run Number	Cor7HP1
Number of Topics	50
Total number of documents over all topics	
Retrieved:	750
Relevant:	4674
Rel-ret:	434

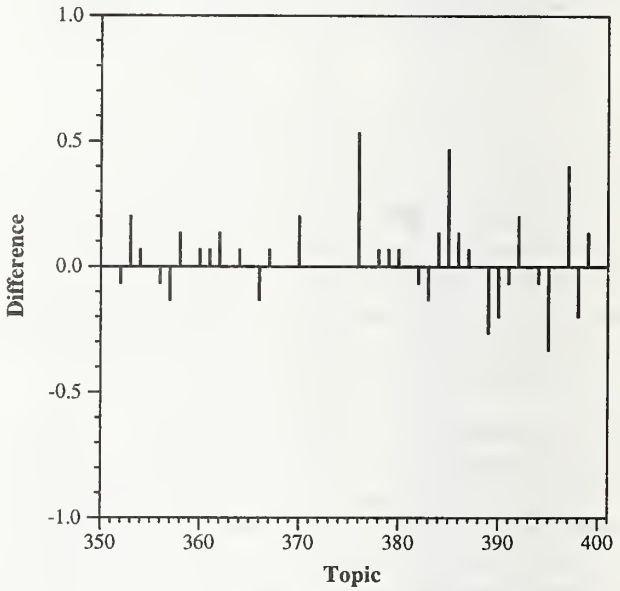
Means over 50 topics	
Precision at 15 Docs:	0.5787
Relative-Precision at 15 Docs:	0.5909
Unranked-Avg-Precision at 15 Docs:	0.1125

Evaluation Measures

Precision at 15: The percentage of documents retrieved in the top fifteen that are relevant. If fewer than 15 documents are retrieved, then all missing documents are assumed to be non-relevant. Precision considers each retrieved relevant document to be equally important, no matter if is retrieved for a query with 500 relevant documents or a query with two relevant documents.

Relative Precision at 15: The precision after fifteen documents relative to the maximum precision possible at that point. For example, if there are 5 relevant documents and 3 of those are retrieved, then Precision at 15 is .2, but the Relative-Precision is .2/.33 or .06. Relative-Precision considers each query to be equally important. Thus a query with only 5 relevant documents has the same maximum score of 1.0 as a query with fifteen or more relevant documents.

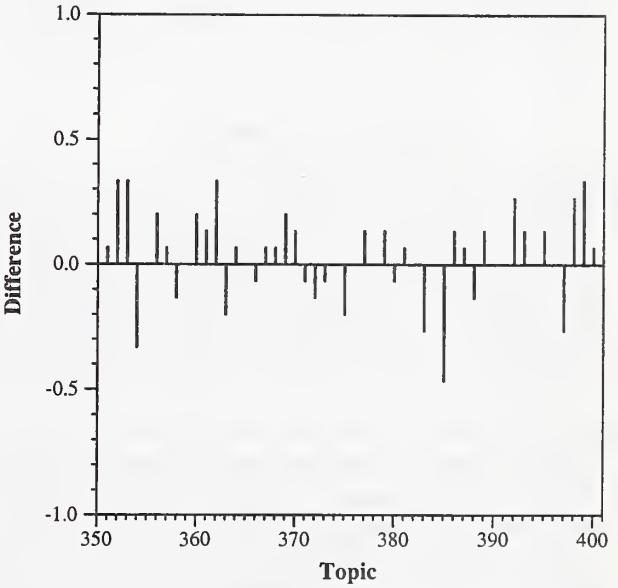
Unranked-Average Precision at 15: Similar to the standard TREC "average precision" measure, with all retrieved relevant documents getting a precision value of the retrieved set (i.e., $r/15$ where r is the number of relevant documents retrieved). All non-retrieved relevant documents get a precision value of 0. This measure is actually directed at unranked evaluation where the size of the retrieved set is under the control of the user, and is not completely appropriate for the High Precision track. It is included to gain more operational experience with the measure, and to see if it offers insights into the results.



Precision(15) Difference from Median per Topic

Summary Statistics	
Run Number	Cor7HP2
Number of Topics	50
Total number of documents over all topics	
Retrieved:	750
Relevant:	4674
Rel-ret:	436

Means over 50 topics	
Precision at 15 Docs:	0.5813
Relative-Precision at 15 Docs:	0.5920
Unranked-Avg-Precision at 15 Docs:	0.1108



Precision(15) Difference from Median per Topic

Evaluation Measures

Precision at 15: The percentage of documents retrieved in the top fifteen that are relevant. If fewer than 15 documents are retrieved, then all missing documents are assumed to be non-relevant. Precision considers each retrieved relevant document to be equally important, no matter if is retrieved for a query with 500 relevant documents or a query with two relevant documents.

Relative Precision at 15: The precision after fifteen documents relative to the maximum precision possible at that point. For example, if there are 5 relevant documents and 3 of those are retrieved, then Precision at 15 is .2, but the Relative-Precision is .2/.33 or .06. Relative-Precision considers each query to be equally important. Thus a query with only 5 relevant documents has the same maximum score of 1.0 as a query with fifteen or more relevant documents.

Unranked-Average Precision at 15: Similar to the standard TREC "average precision" measure, with all retrieved relevant documents getting a precision value of the retrieved set (i.e., $r/15$ where r is the number of relevant documents retrieved). All non-retrieved relevant documents get a precision value of 0. This measure is actually directed at unranked evaluation where the size of the retrieved set is under the control of the user, and is not completely appropriate for the High Precision track. It is included to gain more operational experience with the measure, and to see if it offers insights into the results.

Summary Statistics	
Run Number	Cor7HP3
Number of Topics	50
Total number of documents over all topics	
Retrieved:	750
Relevant:	4674
Rel-ret:	439

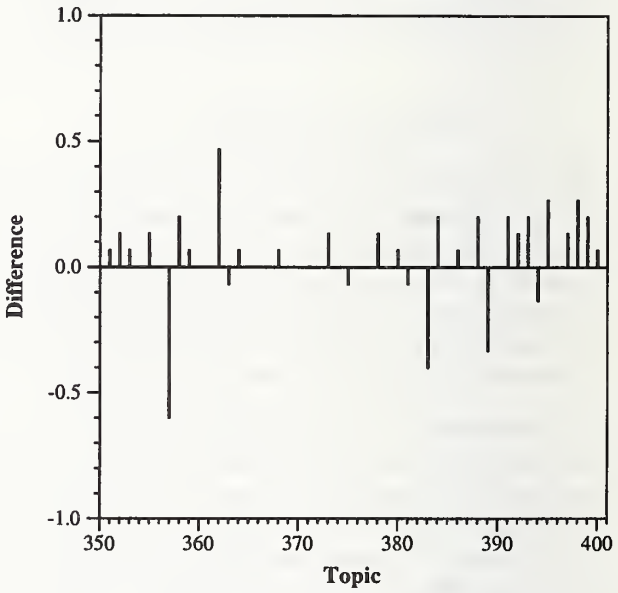
Means over 50 topics	
Precision at 15 Docs:	0.5853
Relative-Precision at 15 Docs:	0.5967
Unranked-Avg-Precision at 15 Docs:	0.1140

Evaluation Measures

Precision at 15: The percentage of documents retrieved in the top fifteen that are relevant. If fewer than 15 documents are retrieved, then all missing documents are assumed to be non-relevant. Precision considers each retrieved relevant document to be equally important, no matter if is retrieved for a query with 500 relevant documents or a query with two relevant documents.

Relative Precision at 15: The precision after fifteen documents relative to the maximum precision possible at that point. For example, if there are 5 relevant documents and 3 of those are retrieved, then Precision at 15 is .2, but the Relative-Precision is .2/.33 or .06. Relative-Precision considers each query to be equally important. Thus a query with only 5 relevant documents has the same maximum score of 1.0 as a query with fifteen or more relevant documents.

Unranked-Average Precision at 15: Similar to the standard TREC "average precision" measure, with all retrieved relevant documents getting a precision value of the retrieved set (i.e., $r/15$ where r is the number of relevant documents retrieved). All non-retrieved relevant documents get a precision value of 0. This measure is actually directed at unranked evaluation where the size of the retrieved set is under the control of the user, and is not completely appropriate for the High Precision track. It is included to gain more operational experience with the measure, and to see if it offers insights into the results.



Precision(15) Difference from Median per Topic

Summary Statistics	
Run Number	pirc8Ha
Number of Topics	50
Total number of documents over all topics	
Retrieved:	750
Relevant:	4674
Rel-ret:	358

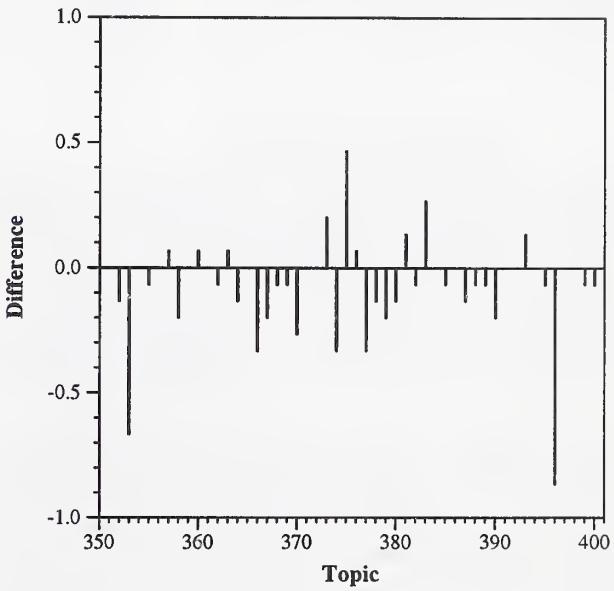
Means over 50 topics	
Precision at 15 Docs:	0.4773
Relative-Precision at 15 Docs:	0.4839
Unranked-Avg-Precision at 15 Docs:	0.0764

Evaluation Measures

Precision at 15: The percentage of documents retrieved in the top fifteen that are relevant. If fewer than 15 documents are retrieved, then all missing documents are assumed to be non-relevant. Precision considers each retrieved relevant document to be equally important, no matter if is retrieved for a query with 500 relevant documents or a query with two relevant documents.

Relative Precision at 15: The precision after fifteen documents relative to the maximum precision possible at that point. For example, if there are 5 relevant documents and 3 of those are retrieved, then Precision at 15 is .2, but the Relative-Precision is .2/.33 or .06. Relative-Precision considers each query to be equally important. Thus a query with only 5 relevant documents has the same maximum score of 1.0 as a query with fifteen or more relevant documents.

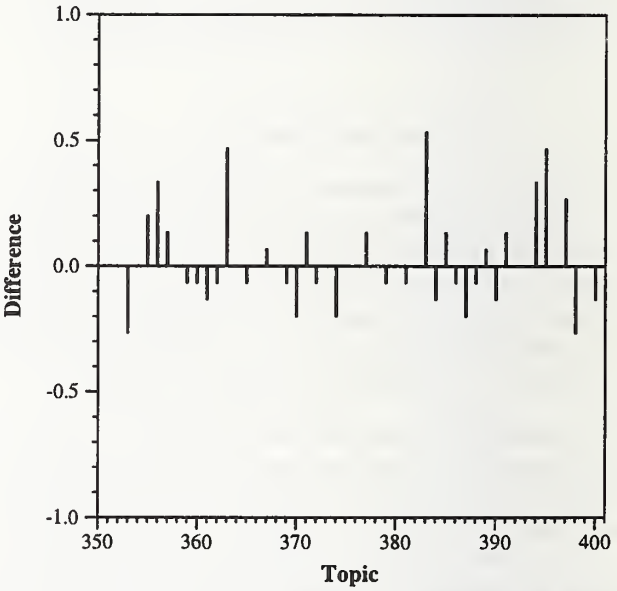
Unranked-Average Precision at 15: Similar to the standard TREC "average precision" measure, with all retrieved relevant documents getting a precision value of the retrieved set (i.e., $r/15$ where r is the number of relevant documents retrieved). All non-retrieved relevant documents get a precision value of 0. This measure is actually directed at unranked evaluation where the size of the retrieved set is under the control of the user, and is not completely appropriate for the High Precision track. It is included to gain more operational experience with the measure, and to see if it offers insights into the results.



Precision(15) Difference from Median per Topic

Summary Statistics	
Run Number	uwmt7h1
Number of Topics	50
Total number of documents over all topics	
Retrieved:	734
Relevant:	4674
Rel-ret:	427

Means over 50 topics	
Precision at 15 Docs:	0.5693
Relative-Precision at 15 Docs:	0.5772
Unranked-Avg-Precision at 15 Docs:	0.1078



Precision(15) Difference from Median per Topic

Evaluation Measures

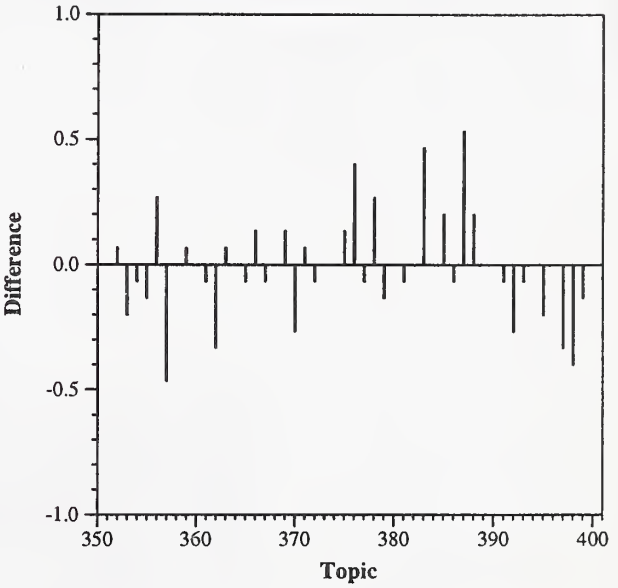
Precision at 15: The percentage of documents retrieved in the top fifteen that are relevant. If fewer than 15 documents are retrieved, then all missing documents are assumed to be non-relevant. Precision considers each retrieved relevant document to be equally important, no matter if is retrieved for a query with 500 relevant documents or a query with two relevant documents.

Relative Precision at 15: The precision after fifteen documents relative to the maximum precision possible at that point. For example, if there are 5 relevant documents and 3 of those are retrieved, then Precision at 15 is .2, but the Relative-Precision is .2/.33 or .06. Relative-Precision considers each query to be equally important. Thus a query with only 5 relevant documents has the same maximum score of 1.0 as a query with fifteen or more relevant documents.

Unranked-Average Precision at 15: Similar to the standard TREC "average precision" measure, with all retrieved relevant documents getting a precision value of the retrieved set (i.e., $r/15$ where r is the number of relevant documents retrieved). All non-retrieved relevant documents get a precision value of 0. This measure is actually directed at unranked evaluation where the size of the retrieved set is under the control of the user, and is not completely appropriate for the High Precision track. It is included to gain more operational experience with the measure, and to see if it offers insights into the results.

Summary Statistics	
Run Number	uwmt7h2
Number of Topics	50
Total number of documents over all topics	
Retrieved:	711
Relevant:	4674
Rel-ret:	403

Means over 50 topics	
Precision at 15 Docs:	0.5373
Relative-Precision at 15 Docs:	0.5467
Unranked-Avg-Precision at 15 Docs:	0.0915



Precision(15) Difference from Median per Topic

Evaluation Measures

Precision at 15: The percentage of documents retrieved in the top fifteen that are relevant. If fewer than 15 documents are retrieved, then all missing documents are assumed to be non-relevant. Precision considers each retrieved relevant document to be equally important, no matter if is retrieved for a query with 500 relevant documents or a query with two relevant documents.

Relative Precision at 15: The precision after fifteen documents relative to the maximum precision possible at that point. For example, if there are 5 relevant documents and 3 of those are retrieved, then Precision at 15 is .2, but the Relative-Precision is .2/.33 or .06. Relative-Precision considers each query to be equally important. Thus a query with only 5 relevant documents has the same maximum score of 1.0 as a query with fifteen or more relevant documents.

Unranked-Average Precision at 15: Similar to the standard TREC "average precision" measure, with all retrieved relevant documents getting a precision value of the retrieved set (i.e., $r/15$ where r is the number of relevant documents retrieved). All non-retrieved relevant documents get a precision value of 0. This measure is actually directed at unranked evaluation where the size of the retrieved set is under the control of the user, and is not completely appropriate for the High Precision track. It is included to gain more operational experience with the measure, and to see if it offers insights into the results.

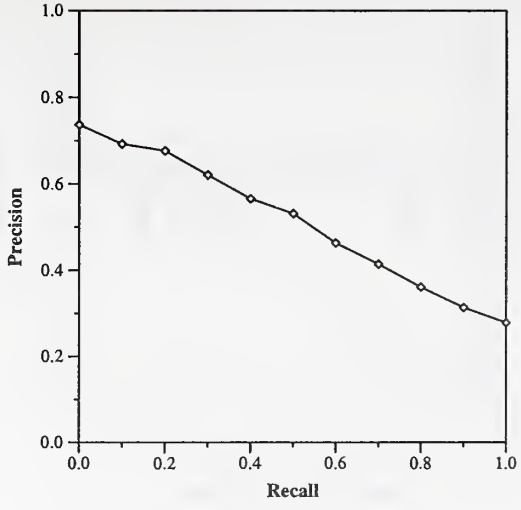
TREC-7 Query Track Results

	APL			Cornell		
Query Set	P(20)	R-Prec	Ave Prec	P(20)	R-Prec	Ave Prec
APL1a	0.1460	0.1017	0.0559	0.2350	0.1583	0.1051
APL2a	0.1230	0.0923	0.0477	0.2710	0.1633	0.1142
APL5a	0.3010	0.2272	0.1627	0.4010	0.2600	0.1971
APL5b	0.5480	0.3289	0.2577	0.6450	0.3727	0.3219
Cor1	0.2730	0.1743	0.1055	0.5030	0.3066	0.2457
Cor2	0.4290	0.2661	0.1846	0.6040	0.3901	0.3367
Cor3	0.2330	0.1670	0.0917	0.4560	0.2774	0.2020
Cor4	—	—	—	0.6500	0.3743	0.3282
Cor5	0.4540	0.3055	0.2296	0.7760	0.4861	0.4586

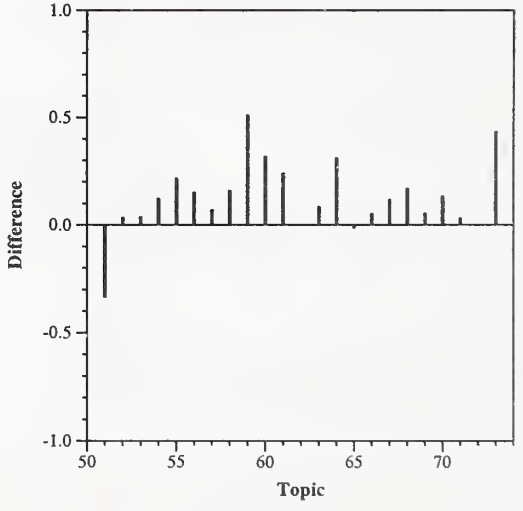
Summary Statistics	
Run Number	att-r1
Run Description	reference
Number of Topics	23
Total number of documents over all topics	
Retrieved:	22675
Relevant:	390
Rel-ret:	384

Recall Level Precision Averages	
Recall	Precision
0.00	0.7372
0.10	0.6927
0.20	0.6770
0.30	0.6209
0.40	0.5659
0.50	0.5314
0.60	0.4636
0.70	0.4147
0.80	0.3612
0.90	0.3139
1.00	0.2786
Average precision over all relevant docs	
non-interpolated	0.4992

Document Level Averages	
	Precision
At 5 docs	0.5478
At 10 docs	0.4522
At 15 docs	0.3884
At 20 docs	0.3413
At 30 docs	0.2768
At 100 docs	0.1265
At 200 docs	0.0720
At 500 docs	0.0323
At 1000 docs	0.0167
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4336



Recall-Precision Curve

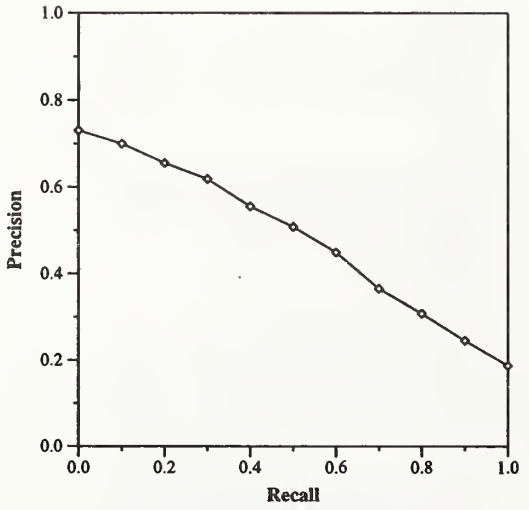


Difference from Median in Average Precision per Topic

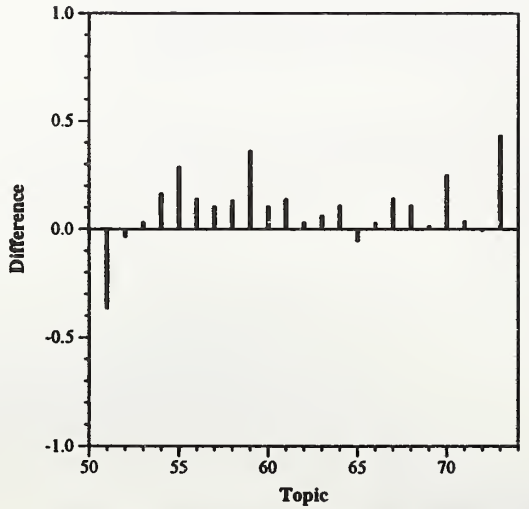
Summary Statistics	
Run Number	att-b1
Run Description	baseline1
Number of Topics	23
Total number of documents over all topics	
Retrieved:	22925
Relevant:	390
Rel-ret:	382

Recall Level Precision Averages	
Recall	Precision
0.00	0.7304
0.10	0.7001
0.20	0.6556
0.30	0.6186
0.40	0.5552
0.50	0.5082
0.60	0.4496
0.70	0.3656
0.80	0.3078
0.90	0.2454
1.00	0.1875
Average precision over all relevant docs	
non-interpolated	0.4700

Document Level Averages	
	Precision
At 5 docs	0.5130
At 10 docs	0.4391
At 15 docs	0.3681
At 20 docs	0.3239
At 30 docs	0.2652
At 100 docs	0.1196
At 200 docs	0.0678
At 500 docs	0.0311
At 1000 docs	0.0166
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4684



Recall-Precision Curve

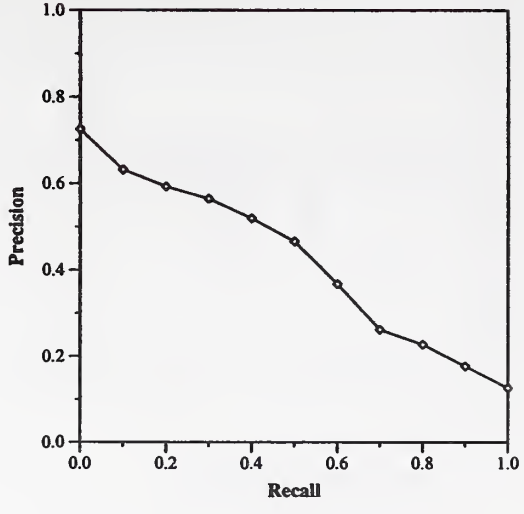


Difference from Median in Average Precision per Topic

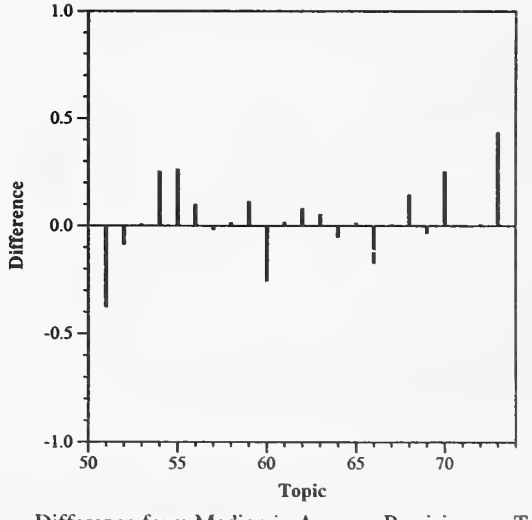
Summary Statistics	
Run Number	att-b2
Run Description	baseline2
Number of Topics	23
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	368

Recall Level Precision Averages	
Recall	Precision
0.00	0.7254
0.10	0.6321
0.20	0.5932
0.30	0.5649
0.40	0.5197
0.50	0.4662
0.60	0.3678
0.70	0.2620
0.80	0.2274
0.90	0.1770
1.00	0.1266
Average precision over all relevant docs	
non-interpolated	0.4065

Document Level Averages	
	Precision
At 5 docs	0.4348
At 10 docs	0.3652
At 15 docs	0.3188
At 20 docs	0.2870
At 30 docs	0.2377
At 100 docs	0.1087
At 200 docs	0.0615
At 500 docs	0.0298
At 1000 docs	0.0160
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4070



Recall-Precision Curve

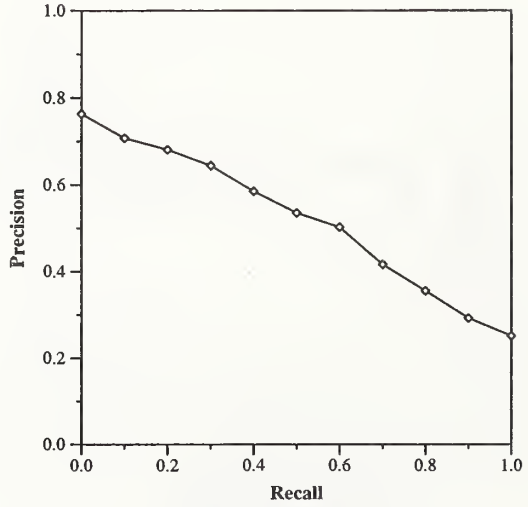


Difference from Median in Average Precision per Topic

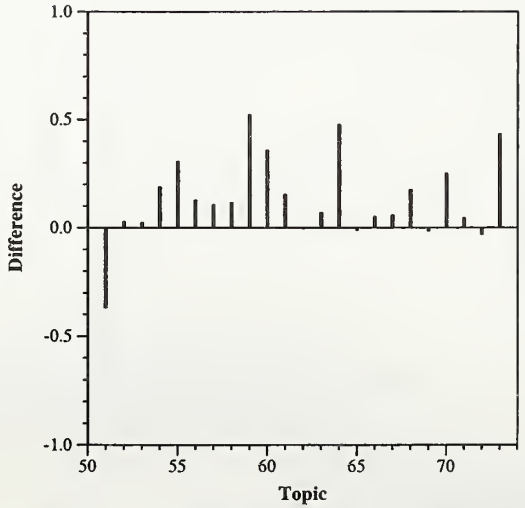
Summary Statistics	
Run Number	att-s1
Run Description	recognizer
Number of Topics	23
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	383

Recall Level Precision Averages	
Recall	Precision
0.00	0.7636
0.10	0.7080
0.20	0.6814
0.30	0.6447
0.40	0.5854
0.50	0.5353
0.60	0.5026
0.70	0.4165
0.80	0.3556
0.90	0.2926
1.00	0.2513
Average precision over all relevant docs	
non-interpolated	0.5065

Document Level Averages	
	Precision
At 5 docs	0.5217
At 10 docs	0.4391
At 15 docs	0.3768
At 20 docs	0.3261
At 30 docs	0.2681
At 100 docs	0.1213
At 200 docs	0.0683
At 500 docs	0.0317
At 1000 docs	0.0167
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4798



Recall-Precision Curve

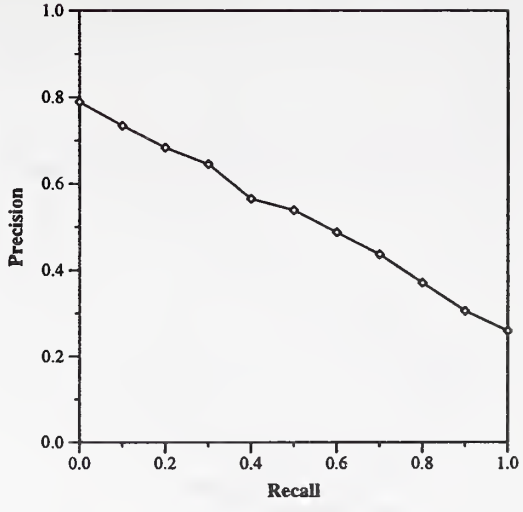


Difference from Median in Average Precision per Topic

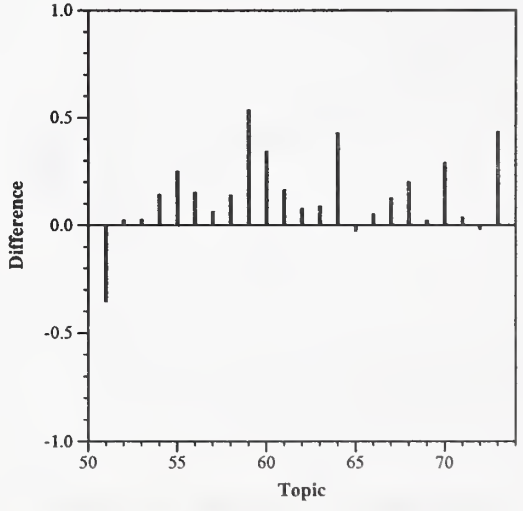
Summary Statistics	
Run Number	att-s2
Run Description	recognizer
Number of Topics	23
Total number of documents over all topics	
Retrieved:	22925
Relevant:	390
Rel-ret:	385

Recall Level Precision Averages	
Recall	Precision
0.00	0.7894
0.10	0.7348
0.20	0.6841
0.30	0.6461
0.40	0.5658
0.50	0.5391
0.60	0.4878
0.70	0.4368
0.80	0.3709
0.90	0.3059
1.00	0.2594
Average precision over all relevant docs	
non-interpolated	0.5120

Document Level Averages	
	Precision
At 5 docs	0.5652
At 10 docs	0.4435
At 15 docs	0.3768
At 20 docs	0.3457
At 30 docs	0.2783
At 100 docs	0.1230
At 200 docs	0.0691
At 500 docs	0.0317
At 1000 docs	0.0167
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4573



Recall-Precision Curve

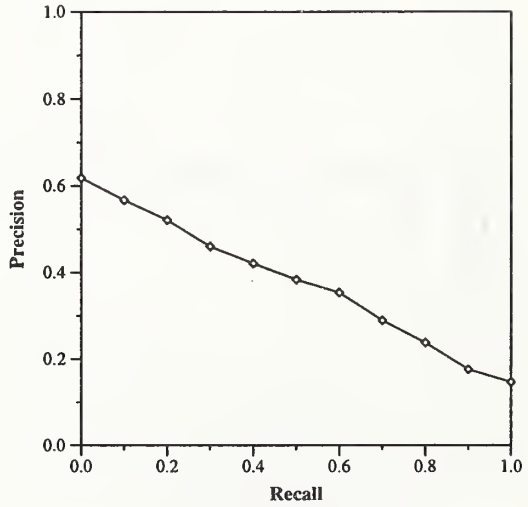


Difference from Median in Average Precision per Topic

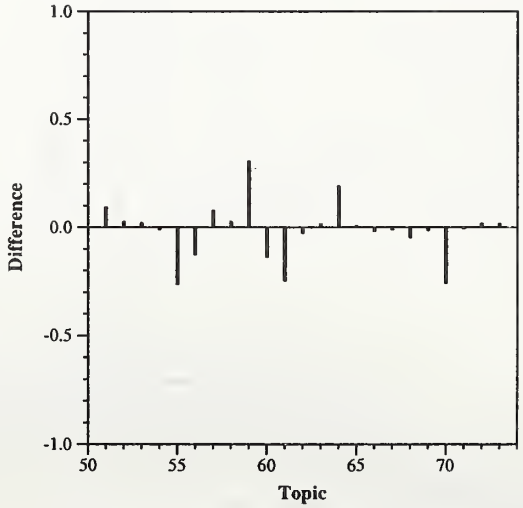
Summary Statistics	
Run Number	cmu1-r.ret
Run Description	reference
Number of Topics	23
Total number of documents over all topics	
Retrieved:	16405
Relevant:	390
Rel-ret:	354

Recall Level Precision Averages	
Recall	Precision
0.00	0.6182
0.10	0.5672
0.20	0.5212
0.30	0.4602
0.40	0.4211
0.50	0.3839
0.60	0.3537
0.70	0.2897
0.80	0.2380
0.90	0.1762
1.00	0.1468
Average precision over all relevant docs	
non-interpolated	0.3577

Document Level Averages	
	Precision
At 5 docs	0.4174
At 10 docs	0.3826
At 15 docs	0.3275
At 20 docs	0.3000
At 30 docs	0.2406
At 100 docs	0.1109
At 200 docs	0.0643
At 500 docs	0.0303
At 1000 docs	0.0154
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3321



Recall-Precision Curve

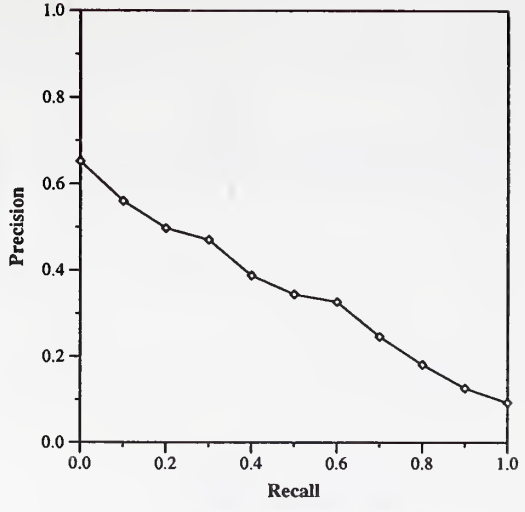


Difference from Median in Average Precision per Topic

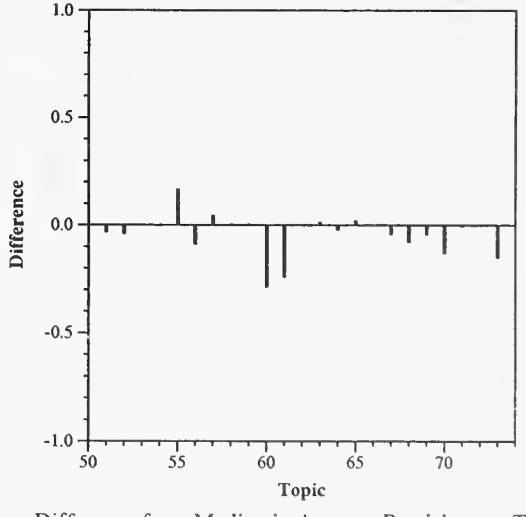
Summary Statistics	
Run Number	cmu1-b1.ret
Run Description	baseline1
Number of Topics	23
Total number of documents over all topics	
Retrieved:	16890
Relevant:	390
Rel-ret:	345

Recall Level Precision Averages	
Recall	Precision
0.00	0.6521
0.10	0.5603
0.20	0.4978
0.30	0.4707
0.40	0.3883
0.50	0.3446
0.60	0.3270
0.70	0.2463
0.80	0.1816
0.90	0.1266
1.00	0.0929
Average precision over all relevant docs	
non-interpolated	0.3345

Document Level Averages	
	Precision
At 5 docs	0.3913
At 10 docs	0.3478
At 15 docs	0.3043
At 20 docs	0.2761
At 30 docs	0.2217
At 100 docs	0.1074
At 200 docs	0.0622
At 500 docs	0.0292
At 1000 docs	0.0150
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3302



Recall-Precision Curve

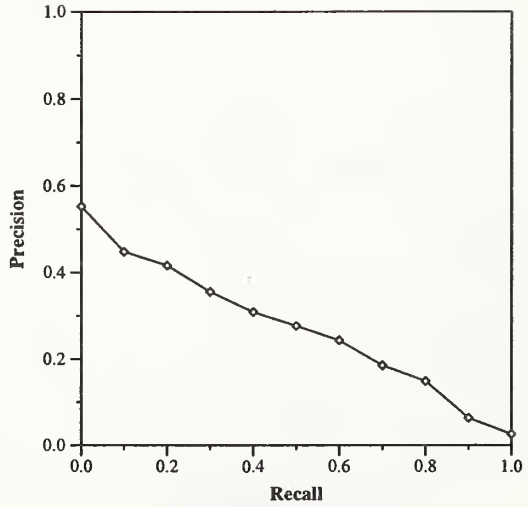


Difference from Median in Average Precision per Topic

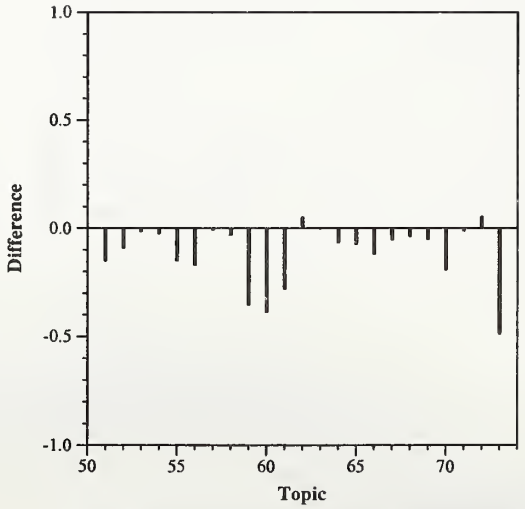
Summary Statistics	
Run Number	cmu1-b2.ret
Run Description	baseline2
Number of Topics	23
Total number of documents over all topics	
Retrieved:	17277
Relevant:	390
Rel-ret:	338

Recall Level Precision Averages	
Recall	Precision
0.00	0.5527
0.10	0.4481
0.20	0.4162
0.30	0.3557
0.40	0.3088
0.50	0.2769
0.60	0.2433
0.70	0.1854
0.80	0.1489
0.90	0.0633
1.00	0.0257
Average precision over all relevant docs	
non-interpolated	0.2590

Document Level Averages	
	Precision
At 5 docs	0.3130
At 10 docs	0.3043
At 15 docs	0.2580
At 20 docs	0.2326
At 30 docs	0.1942
At 100 docs	0.0965
At 200 docs	0.0574
At 500 docs	0.0283
At 1000 docs	0.0147
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2780



Recall-Precision Curve

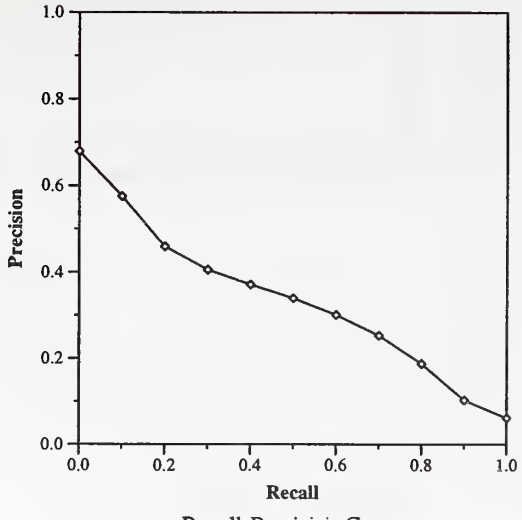


Difference from Median in Average Precision per Topic

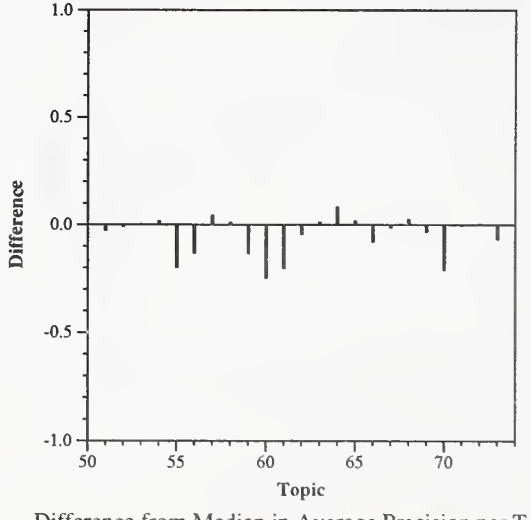
Summary Statistics	
Run Number	cmu1-s1.ret
Run Description	recognizer
Number of Topics	23
Total number of documents over all topics	
Retrieved:	16487
Relevant:	390
Rel-ret:	350

Recall Level Precision Averages	
Recall	Precision
0.00	0.6795
0.10	0.5756
0.20	0.4594
0.30	0.4060
0.40	0.3717
0.50	0.3399
0.60	0.3012
0.70	0.2532
0.80	0.1877
0.90	0.1031
1.00	0.0616
Average precision over all relevant docs	
non-interpolated	0.3224

Document Level Averages	
	Precision
At 5 docs	0.3739
At 10 docs	0.3478
At 15 docs	0.3130
At 20 docs	0.2674
At 30 docs	0.2290
At 100 docs	0.1078
At 200 docs	0.0626
At 500 docs	0.0300
At 1000 docs	0.0152
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3328



Recall-Precision Curve

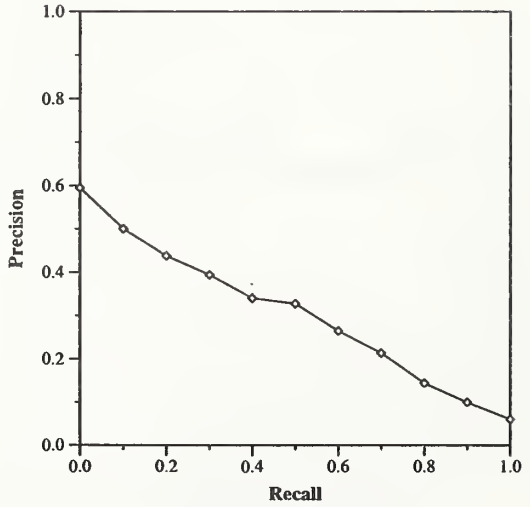


Difference from Median in Average Precision per Topic

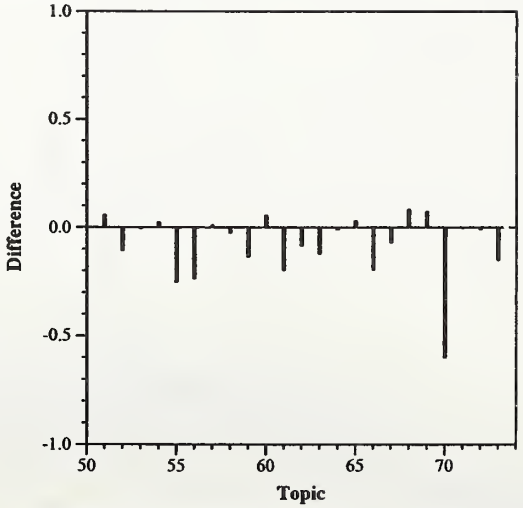
Summary Statistics	
Run Number	cmu1-s2.ret
Run Description	recognizer
Number of Topics	23
Total number of documents over all topics	
Retrieved:	16514
Relevant:	390
Rel-ret:	350

Recall Level Precision Averages	
Recall	Precision
0.00	0.5947
0.10	0.5007
0.20	0.4385
0.30	0.3948
0.40	0.3408
0.50	0.3281
0.60	0.2651
0.70	0.2141
0.80	0.1442
0.90	0.0997
1.00	0.0606
Average precision over all relevant docs	
non-interpolated	0.2926

Document Level Averages	
	Precision
At 5 docs	0.3304
At 10 docs	0.3130
At 15 docs	0.2783
At 20 docs	0.2413
At 30 docs	0.2130
At 100 docs	0.1052
At 200 docs	0.0635
At 500 docs	0.0293
At 1000 docs	0.0152
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3017



Recall-Precision Curve

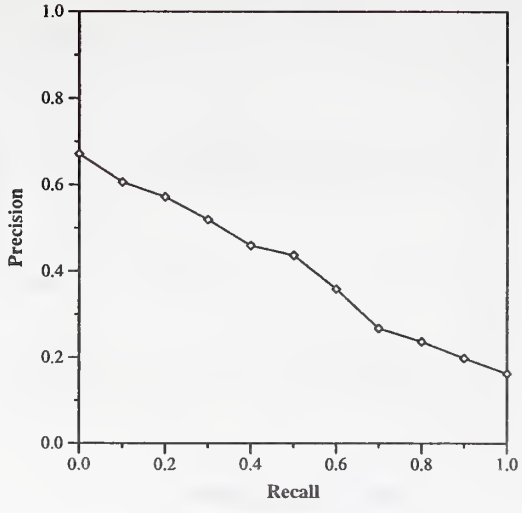


Difference from Median in Average Precision per Topic

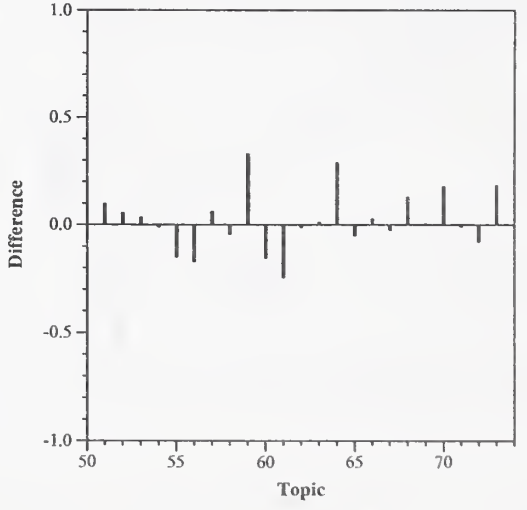
Summary Statistics	
Run Number	cmu2-r1.ret
Run Description	reference
Number of Topics	23
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	345

Recall Level Precision Averages	
Recall	Precision
0.00	0.6716
0.10	0.6065
0.20	0.5723
0.30	0.5197
0.40	0.4598
0.50	0.4372
0.60	0.3593
0.70	0.2680
0.80	0.2369
0.90	0.1985
1.00	0.1625
Average precision over all relevant docs	
non-interpolated	0.3936

Document Level Averages	
	Precision
At 5 docs	0.4609
At 10 docs	0.3870
At 15 docs	0.3362
At 20 docs	0.3000
At 30 docs	0.2435
At 100 docs	0.1026
At 200 docs	0.0598
At 500 docs	0.0283
At 1000 docs	0.0150
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3638



Recall-Precision Curve

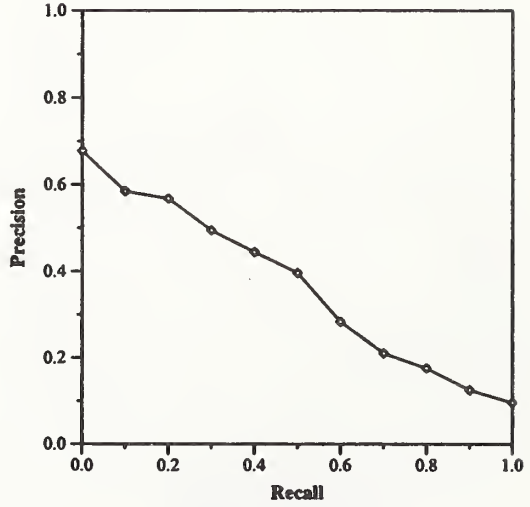


Difference from Median in Average Precision per Topic

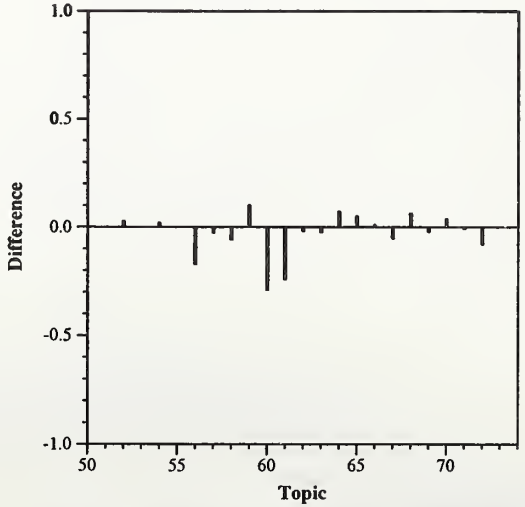
Summary Statistics	
Run Number	cmu2-b1.ret
Run Description	baseline1
Number of Topics	23
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	328

Recall Level Precision Averages	
Recall	Precision
0.00	0.6779
0.10	0.5847
0.20	0.5679
0.30	0.4946
0.40	0.4446
0.50	0.3966
0.60	0.2836
0.70	0.2106
0.80	0.1759
0.90	0.1254
1.00	0.0960
Average precision over all relevant docs	
non-interpolated	0.3472

Document Level Averages	
	Precision
At 5 docs	0.4348
At 10 docs	0.3348
At 15 docs	0.3217
At 20 docs	0.2826
At 30 docs	0.2232
At 100 docs	0.0974
At 200 docs	0.0572
At 500 docs	0.0272
At 1000 docs	0.0143
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3284



Recall-Precision Curve

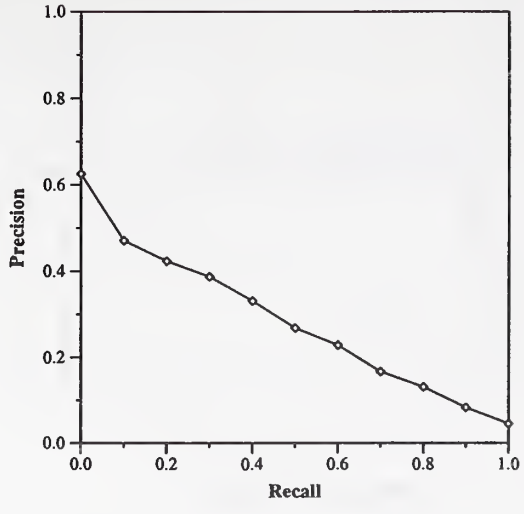


Difference from Median in Average Precision per Topic

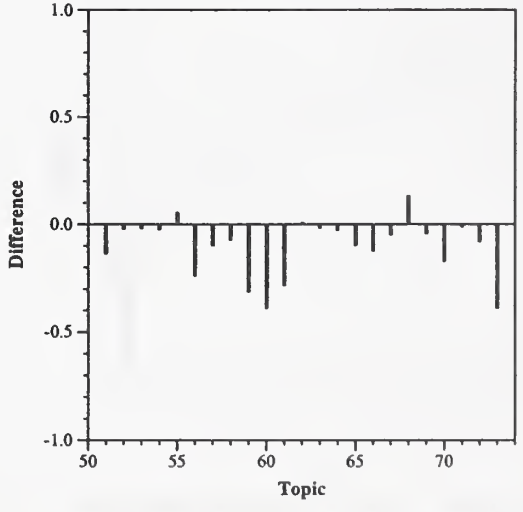
Summary Statistics	
Run Number	cmu2-b2.ret
Run Description	baseline2
Number of Topics	23
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	317

Recall Level Precision Averages	
Recall	Precision
0.00	0.6255
0.10	0.4711
0.20	0.4236
0.30	0.3874
0.40	0.3311
0.50	0.2680
0.60	0.2283
0.70	0.1667
0.80	0.1309
0.90	0.0830
1.00	0.0452
Average precision over all relevant docs	
non-interpolated	0.2693

Document Level Averages	
	Precision
At 5 docs	0.3217
At 10 docs	0.2870
At 15 docs	0.2522
At 20 docs	0.2413
At 30 docs	0.1942
At 100 docs	0.0904
At 200 docs	0.0537
At 500 docs	0.0258
At 1000 docs	0.0138
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2438



Recall-Precision Curve

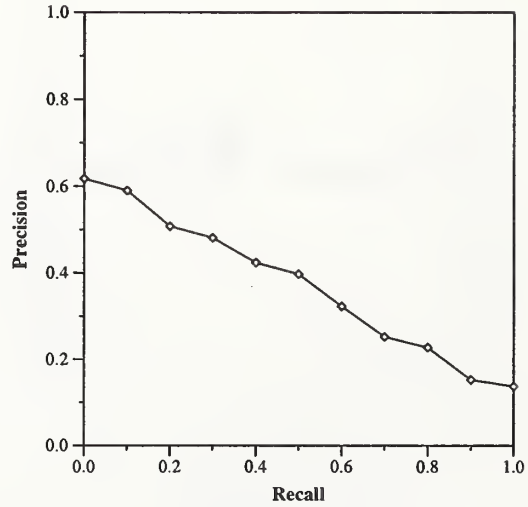


Difference from Median in Average Precision per Topic

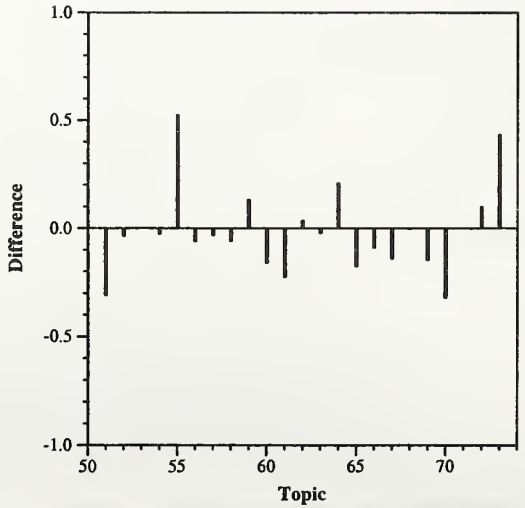
Summary Statistics	
Run Number	derasru1-r1.ret
Run Description	reference
Number of Topics	23
Total number of documents over all topics	
Retrieved:	21650
Relevant:	390
Rel-ret:	370

Recall Level Precision Averages	
Recall	Precision
0.00	0.6175
0.10	0.5904
0.20	0.5080
0.30	0.4814
0.40	0.4241
0.50	0.3977
0.60	0.3231
0.70	0.2526
0.80	0.2279
0.90	0.1529
1.00	0.1377
Average precision over all relevant docs	
non-interpolated	0.3579

Document Level Averages	
	Precision
At 5 docs	0.4087
At 10 docs	0.3304
At 15 docs	0.2696
At 20 docs	0.2391
At 30 docs	0.1942
At 100 docs	0.0978
At 200 docs	0.0587
At 500 docs	0.0290
At 1000 docs	0.0161
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3752



Recall-Precision Curve

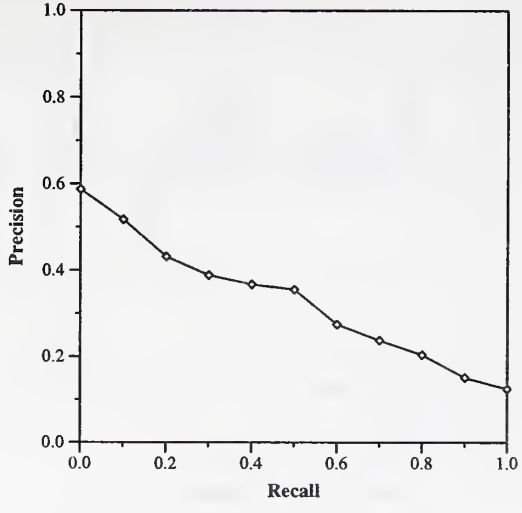


Difference from Median in Average Precision per Topic

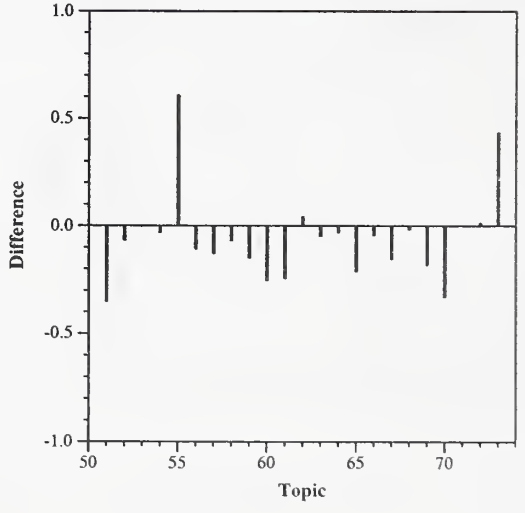
Summary Statistics	
Run Number	derasru1-b1.ret
Run Description	baseline1
Number of Topics	23
Total number of documents over all topics	
Retrieved:	21901
Relevant:	390
Rel-ret:	358

Recall Level Precision Averages	
Recall	Precision
0.00	0.5875
0.10	0.5179
0.20	0.4321
0.30	0.3890
0.40	0.3677
0.50	0.3556
0.60	0.2748
0.70	0.2376
0.80	0.2044
0.90	0.1512
1.00	0.1247
Average precision over all relevant docs	
non-interpolated	0.3164

Document Level Averages	
	Precision
At 5 docs	0.3739
At 10 docs	0.2826
At 15 docs	0.2435
At 20 docs	0.2130
At 30 docs	0.1797
At 100 docs	0.0896
At 200 docs	0.0548
At 500 docs	0.0274
At 1000 docs	0.0156
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3420



Recall-Precision Curve

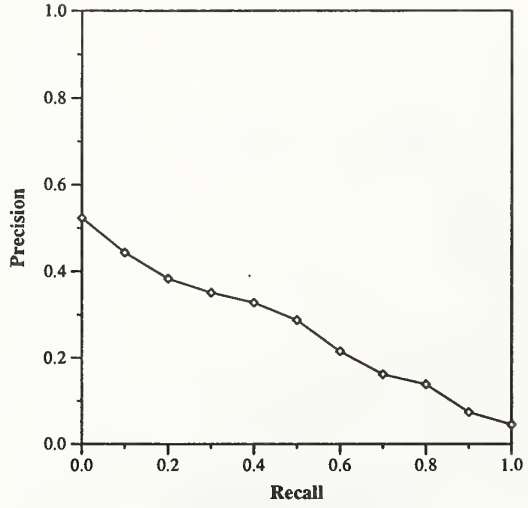


Difference from Median in Average Precision per Topic

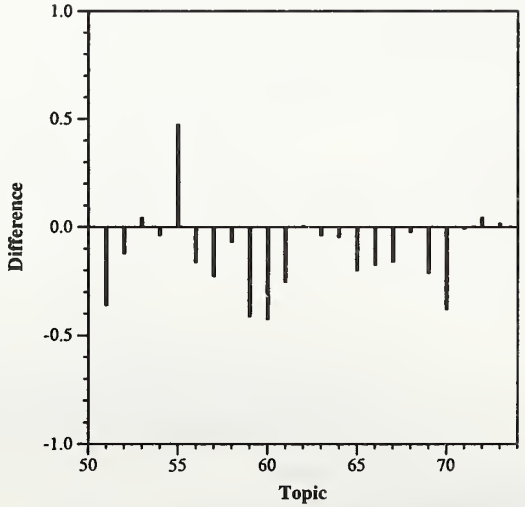
Summary Statistics	
Run Number	derasru1-b2.ret
Run Description	baseline2
Number of Topics	23
Total number of documents over all topics	
Retrieved:	22202
Relevant:	390
Rel-ret:	352

Recall Level Precision Averages	
Recall	Precision
0.00	0.5235
0.10	0.4435
0.20	0.3833
0.30	0.3512
0.40	0.3279
0.50	0.2876
0.60	0.2152
0.70	0.1618
0.80	0.1387
0.90	0.0739
1.00	0.0453
Average precision over all relevant docs	
non-interpolated	0.2551

Document Level Averages	
	Precision
At 5 docs	0.2957
At 10 docs	0.2391
At 15 docs	0.2174
At 20 docs	0.1870
At 30 docs	0.1536
At 100 docs	0.0826
At 200 docs	0.0504
At 500 docs	0.0266
At 1000 docs	0.0153
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2449



Recall-Precision Curve

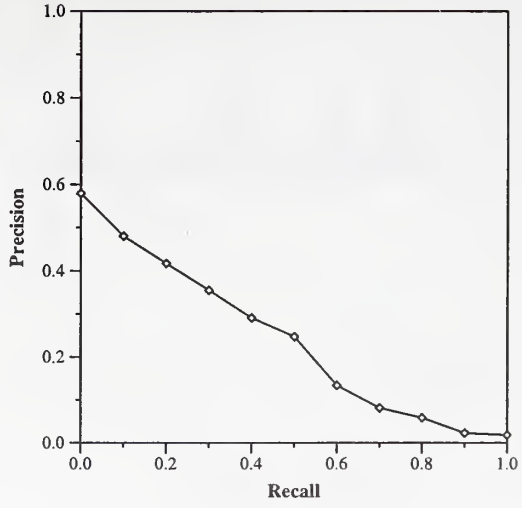


Difference from Median in Average Precision per Topic

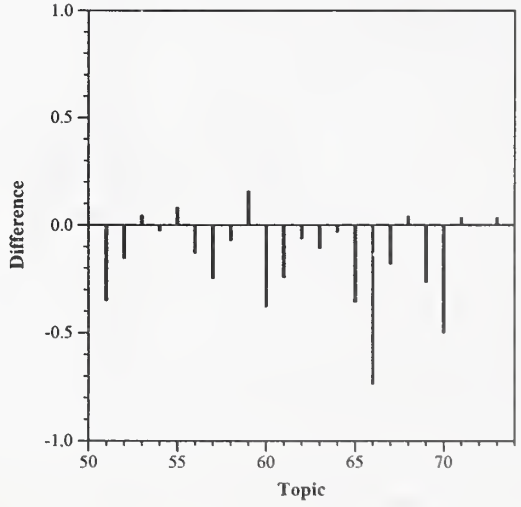
Summary Statistics	
Run Number	derasru1-s1.ret
Run Description	recognizer
Number of Topics	23
Total number of documents over all topics	
Retrieved:	21675
Relevant:	390
Rel-ret:	338

Recall Level Precision Averages	
Recall	Precision
0.00	0.5796
0.10	0.4809
0.20	0.4178
0.30	0.3555
0.40	0.2913
0.50	0.2476
0.60	0.1342
0.70	0.0817
0.80	0.0587
0.90	0.0231
1.00	0.0183
Average precision over all relevant docs	
non-interpolated	0.2242

Document Level Averages	
	Precision
At 5 docs	0.3217
At 10 docs	0.2478
At 15 docs	0.2174
At 20 docs	0.1870
At 30 docs	0.1565
At 100 docs	0.0778
At 200 docs	0.0498
At 500 docs	0.0250
At 1000 docs	0.0147
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2565



Recall-Precision Curve

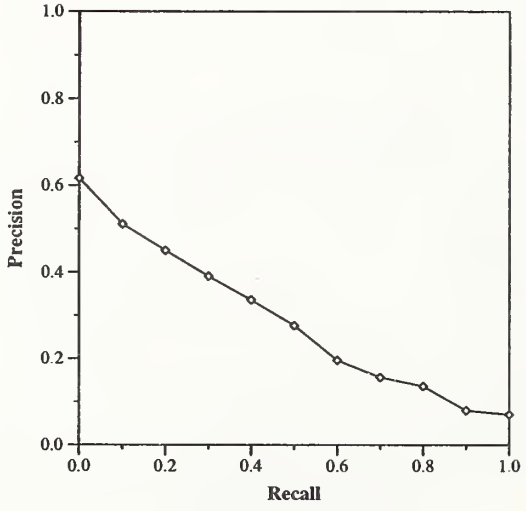


Difference from Median in Average Precision per Topic

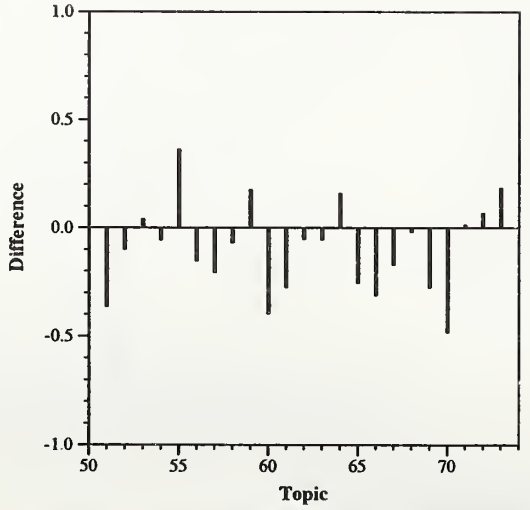
Summary Statistics	
Run Number	derasru1-s21.ret
Run Description	recognizer
Number of Topics	23
Total number of documents over all topics	
Retrieved:	21826
Relevant:	390
Rel-ret:	341

Recall Level Precision Averages	
Recall	Precision
0.00	0.6165
0.10	0.5112
0.20	0.4503
0.30	0.3908
0.40	0.3360
0.50	0.2770
0.60	0.1965
0.70	0.1570
0.80	0.1363
0.90	0.0801
1.00	0.0707
Average precision over all relevant docs	
non-interpolated	0.2768

Document Level Averages	
	Precision
At 5 docs	0.3565
At 10 docs	0.2478
At 15 docs	0.2174
At 20 docs	0.2000
At 30 docs	0.1623
At 100 docs	0.0796
At 200 docs	0.0498
At 500 docs	0.0259
At 1000 docs	0.0148
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2876



Recall-Precision Curve

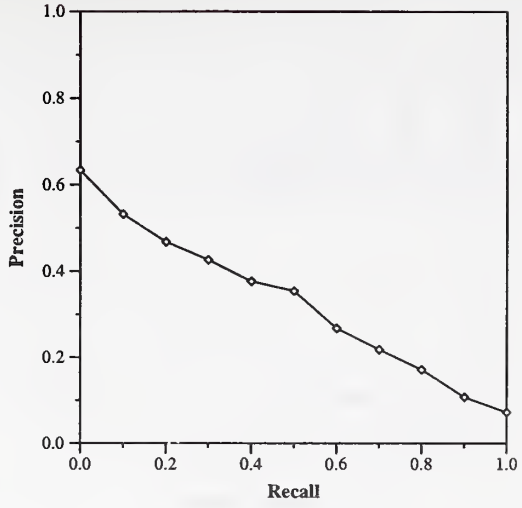


Difference from Median in Average Precision per Topic

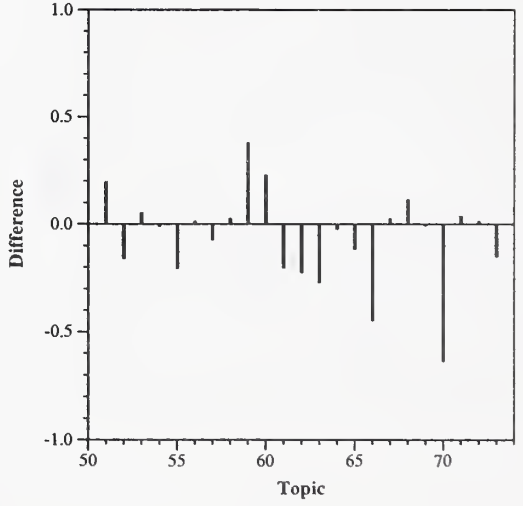
Summary Statistics	
Run Number	mds-r1.ret
Run Description	reference
Number of Topics	23
Total number of documents over all topics	
Retrieved:	22690
Relevant:	390
Rel-ret:	363

Recall Level Precision Averages	
Recall	Precision
0.00	0.6340
0.10	0.5323
0.20	0.4683
0.30	0.4266
0.40	0.3772
0.50	0.3546
0.60	0.2679
0.70	0.2186
0.80	0.1717
0.90	0.1076
1.00	0.0721
Average precision over all relevant docs	
non-interpolated	0.3107

Document Level Averages	
	Precision
At 5 docs	0.4261
At 10 docs	0.3478
At 15 docs	0.3101
At 20 docs	0.2565
At 30 docs	0.2174
At 100 docs	0.0978
At 200 docs	0.0585
At 500 docs	0.0297
At 1000 docs	0.0158
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2904



Recall-Precision Curve

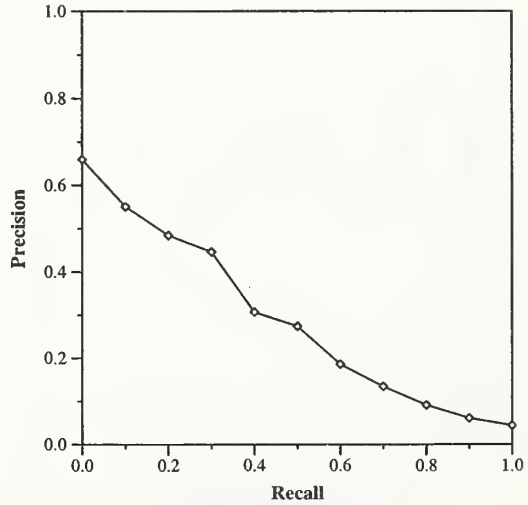


Difference from Median in Average Precision per Topic

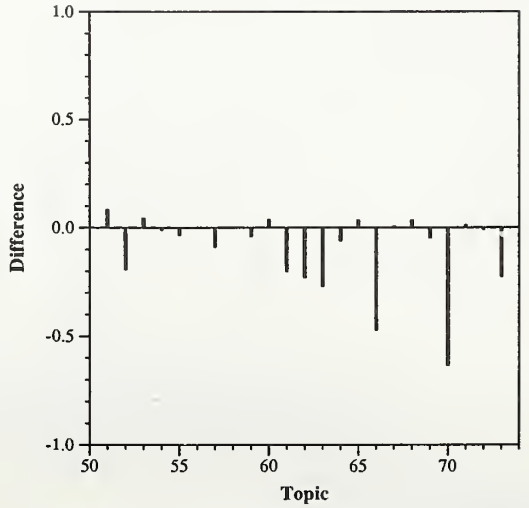
Summary Statistics	
Run Number	mds-b1.ret
Run Description	baseline1
Number of Topics	23
Total number of documents over all topics	
Retrieved:	22586
Relevant:	390
Rel-ret:	358

Recall Level Precision Averages	
Recall	Precision
0.00	0.6596
0.10	0.5509
0.20	0.4846
0.30	0.4462
0.40	0.3070
0.50	0.2742
0.60	0.1863
0.70	0.1345
0.80	0.0916
0.90	0.0613
1.00	0.0442
Average precision over all relevant docs	
non-interpolated	0.2753

Document Level Averages	
	Precision
At 5 docs	0.3739
At 10 docs	0.3217
At 15 docs	0.2696
At 20 docs	0.2391
At 30 docs	0.1957
At 100 docs	0.0913
At 200 docs	0.0546
At 500 docs	0.0287
At 1000 docs	0.0156
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2889



Recall-Precision Curve

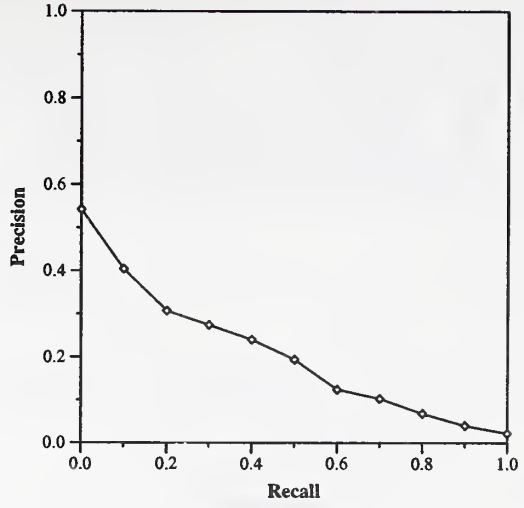


Difference from Median in Average Precision per Topic

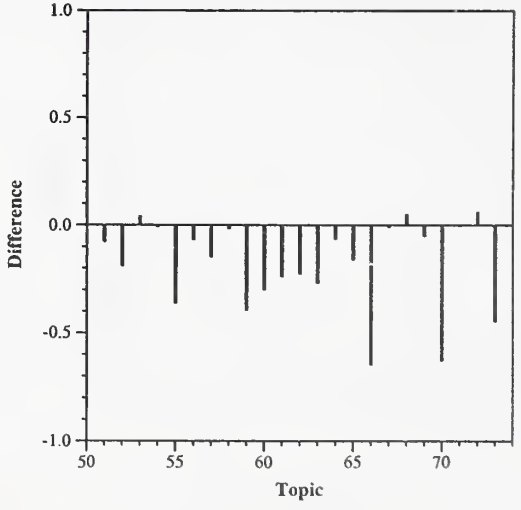
Summary Statistics	
Run Number	mds-b2.ret
Run Description	baseline2
Number of Topics	23
Total number of documents over all topics	
Retrieved:	22007
Relevant:	390
Rel-ret:	351

Recall Level Precision Averages	
Recall	Precision
0.00	0.5423
0.10	0.4043
0.20	0.3076
0.30	0.2746
0.40	0.2406
0.50	0.1943
0.60	0.1245
0.70	0.1030
0.80	0.0683
0.90	0.0402
1.00	0.0221
Average precision over all relevant docs	
non-interpolated	0.1937

Document Level Averages	
	Precision
At 5 docs	0.2870
At 10 docs	0.2913
At 15 docs	0.2464
At 20 docs	0.2152
At 30 docs	0.1768
At 100 docs	0.0822
At 200 docs	0.0502
At 500 docs	0.0281
At 1000 docs	0.0153
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2046



Recall-Precision Curve

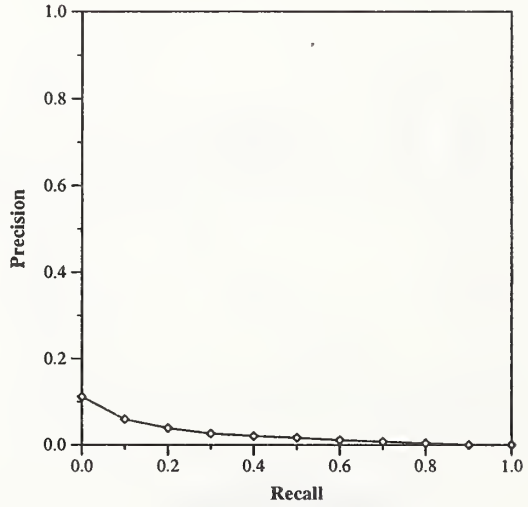


Difference from Median in Average Precision per Topic

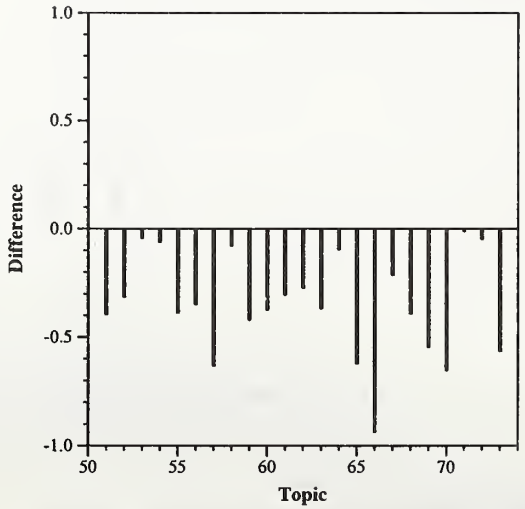
Summary Statistics	
Run Number	mds-s1.ret
Run Description	recognizer
Number of Topics	23
Total number of documents over all topics	
Retrieved:	22677
Relevant:	390
Rel-ret:	268

Recall Level Precision Averages	
Recall	Precision
0.00	0.1119
0.10	0.0596
0.20	0.0394
0.30	0.0266
0.40	0.0208
0.50	0.0168
0.60	0.0114
0.70	0.0080
0.80	0.0042
0.90	0.0002
1.00	0.0002
Average precision over all relevant docs	
non-interpolated	0.0223

Document Level Averages	
	Precision
At 5 docs	0.0435
At 10 docs	0.0391
At 15 docs	0.0377
At 20 docs	0.0326
At 30 docs	0.0333
At 100 docs	0.0226
At 200 docs	0.0193
At 500 docs	0.0137
At 1000 docs	0.0117
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0288



Recall-Precision Curve

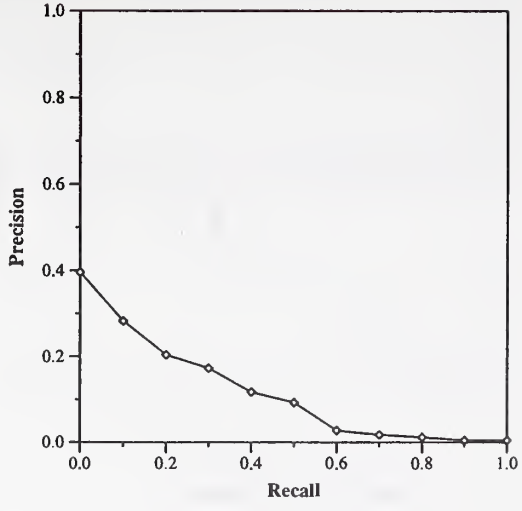


Difference from Median in Average Precision per Topic

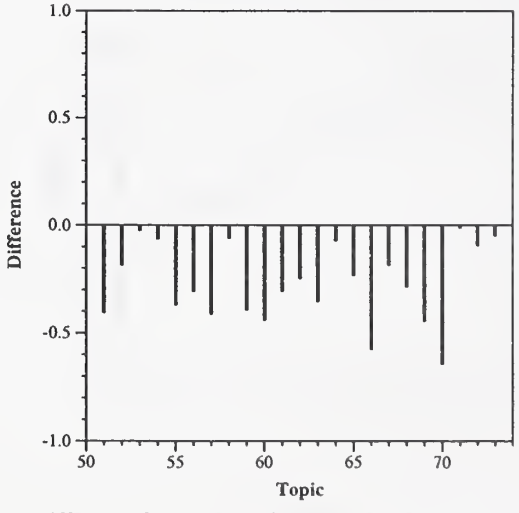
Summary Statistics	
Run Number	mds-s2.ret
Run Description	recognizer
Number of Topics	23
Total number of documents over all topics	
Retrieved:	22700
Relevant:	390
Rel-ret:	275

Recall Level Precision Averages	
Recall	Precision
0.00	0.3962
0.10	0.2830
0.20	0.2044
0.30	0.1733
0.40	0.1175
0.50	0.0934
0.60	0.0278
0.70	0.0178
0.80	0.0119
0.90	0.0046
1.00	0.0046
Average precision over all relevant docs	
non-interpolated	0.1063

Document Level Averages	
	Precision
At 5 docs	0.1391
At 10 docs	0.1174
At 15 docs	0.1043
At 20 docs	0.0913
At 30 docs	0.0710
At 100 docs	0.0465
At 200 docs	0.0302
At 500 docs	0.0187
At 1000 docs	0.0120
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1255



Recall-Precision Curve

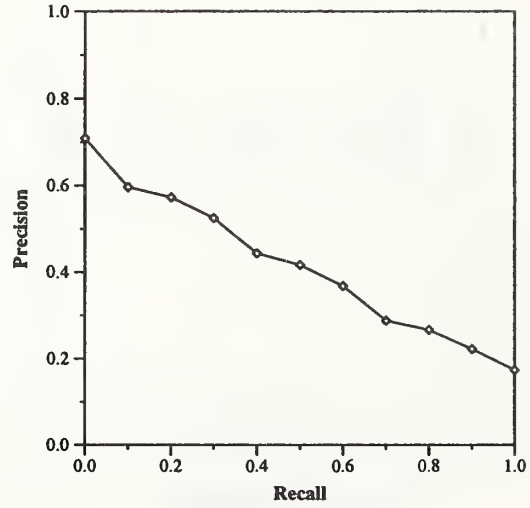


Difference from Median in Average Precision per Topic

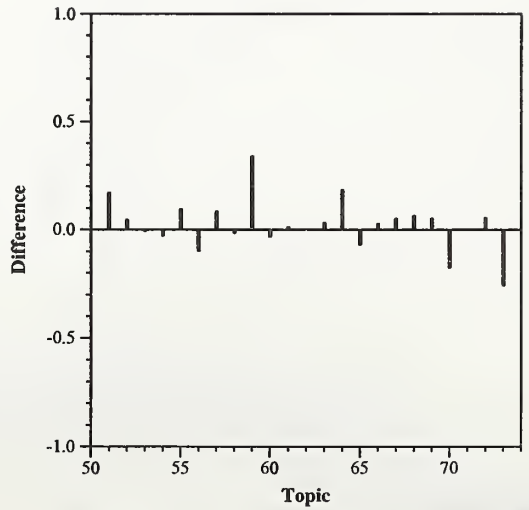
Summary Statistics	
Run Number	tno-r1.ret
Run Description	reference
Number of Topics	23
Total number of documents over all topics	
Retrieved:	14299
Relevant:	390
Rel-ret:	352

Recall Level Precision Averages	
Recall	Precision
0.00	0.7090
0.10	0.5964
0.20	0.5728
0.30	0.5250
0.40	0.4436
0.50	0.4167
0.60	0.3682
0.70	0.2882
0.80	0.2669
0.90	0.2224
1.00	0.1737
Average precision over all relevant docs	
non-interpolated	0.3970

Document Level Averages	
	Precision
At 5 docs	0.4609
At 10 docs	0.3826
At 15 docs	0.3449
At 20 docs	0.3109
At 30 docs	0.2667
At 100 docs	0.1117
At 200 docs	0.0667
At 500 docs	0.0303
At 1000 docs	0.0153
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3719



Recall-Precision Curve

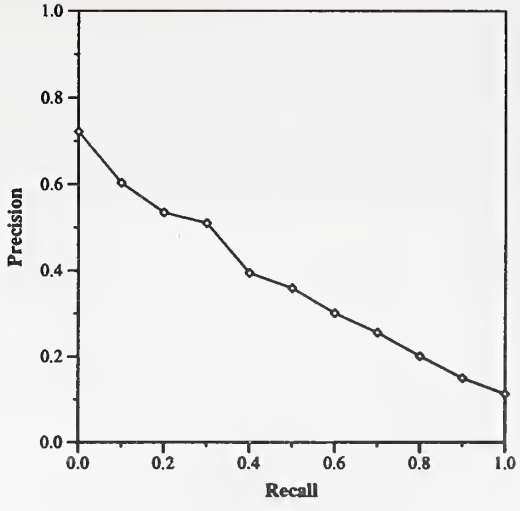


Difference from Median in Average Precision per Topic

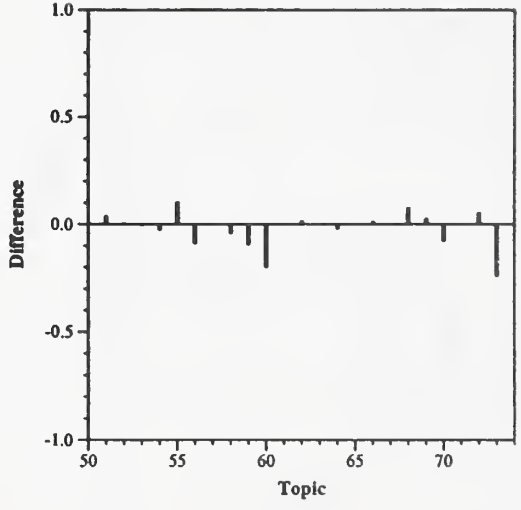
Summary Statistics	
Run Number	tno-b1.ret
Run Description	baseline1
Number of Topics	23
Total number of documents over all topics	
Retrieved:	14764
Relevant:	390
Rel-ret:	340

Recall Level Precision Averages	
Recall	Precision
0.00	0.7217
0.10	0.6034
0.20	0.5346
0.30	0.5102
0.40	0.3947
0.50	0.3594
0.60	0.3013
0.70	0.2561
0.80	0.2014
0.90	0.1500
1.00	0.1130
Average precision over all relevant docs	
non-interpolated	0.3533

Document Level Averages	
	Precision
At 5 docs	0.4174
At 10 docs	0.3565
At 15 docs	0.3217
At 20 docs	0.2978
At 30 docs	0.2377
At 100 docs	0.1061
At 200 docs	0.0641
At 500 docs	0.0291
At 1000 docs	0.0148
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3517



Recall-Precision Curve

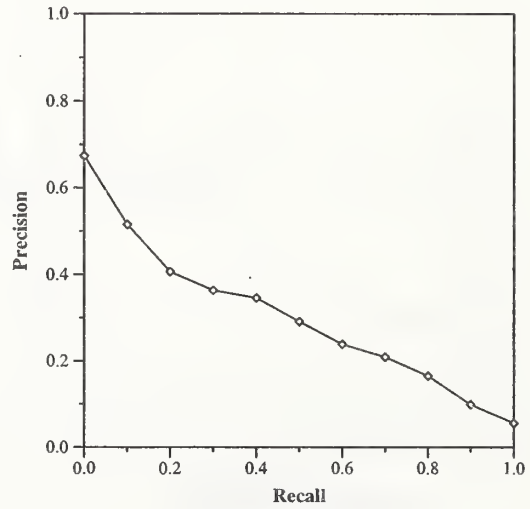


Difference from Median in Average Precision per Topic

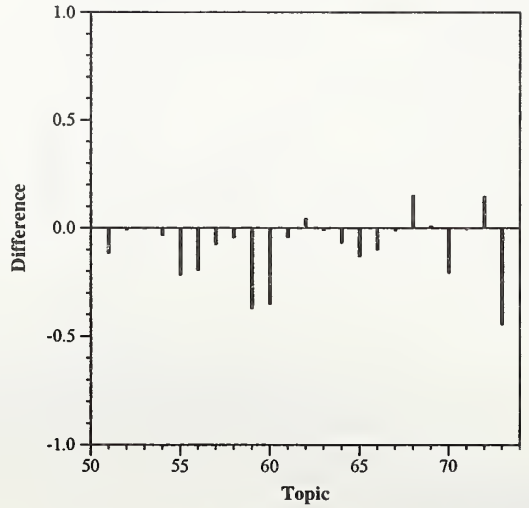
Summary Statistics	
Run Number	tno-b2.ret
Run Description	baseline2
Number of Topics	23
Total number of documents over all topics	
Retrieved:	15130
Relevant:	390
Rel-ret:	332

Recall Level Precision Averages	
Recall	Precision
0.00	0.6741
0.10	0.5157
0.20	0.4065
0.30	0.3636
0.40	0.3460
0.50	0.2914
0.60	0.2389
0.70	0.2095
0.80	0.1655
0.90	0.0993
1.00	0.0562
Average precision over all relevant docs	
non-interpolated	0.2833

Document Level Averages	
	Precision
At 5 docs	0.4000
At 10 docs	0.3043
At 15 docs	0.2870
At 20 docs	0.2652
At 30 docs	0.2174
At 100 docs	0.0978
At 200 docs	0.0604
At 500 docs	0.0281
At 1000 docs	0.0144
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2762



Recall-Precision Curve

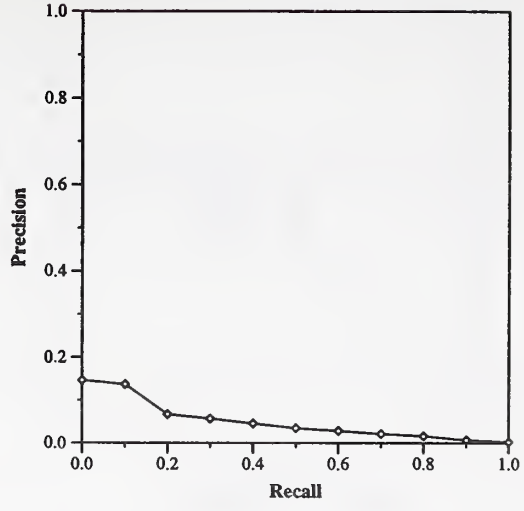


Difference from Median in Average Precision per Topic

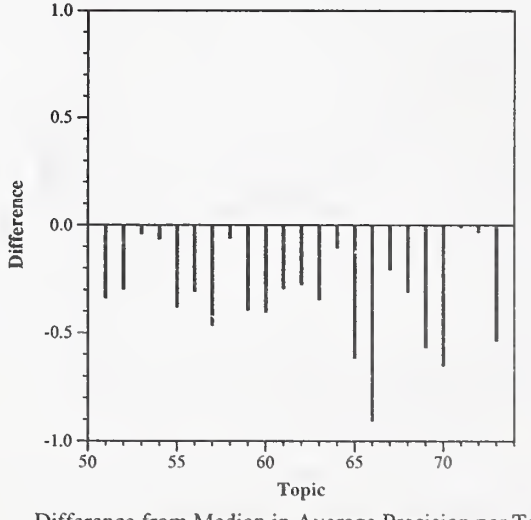
Summary Statistics	
Run Number	tno-s1-abb.ret
Run Description	recognizer
Number of Topics	23
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	326

Recall Level Precision Averages	
Recall	Precision
0.00	0.1465
0.10	0.1371
0.20	0.0669
0.30	0.0569
0.40	0.0455
0.50	0.0347
0.60	0.0284
0.70	0.0214
0.80	0.0160
0.90	0.0069
1.00	0.0024
Average precision over all relevant docs	
non-interpolated	0.0436

Document Level Averages	
	Precision
At 5 docs	0.0435
At 10 docs	0.0391
At 15 docs	0.0377
At 20 docs	0.0348
At 30 docs	0.0493
At 100 docs	0.0396
At 200 docs	0.0287
At 500 docs	0.0204
At 1000 docs	0.0142
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.0580



Recall-Precision Curve

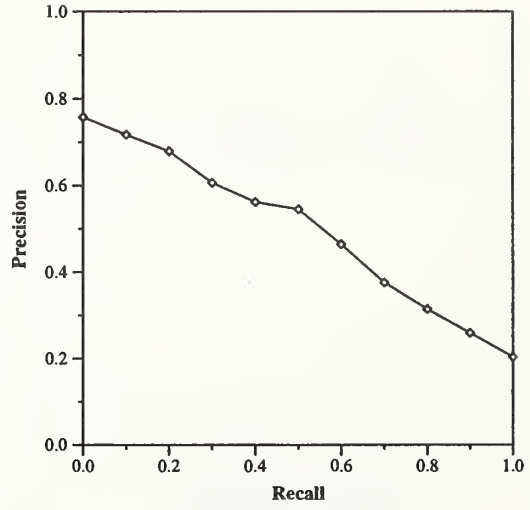


Difference from Median in Average Precision per Topic

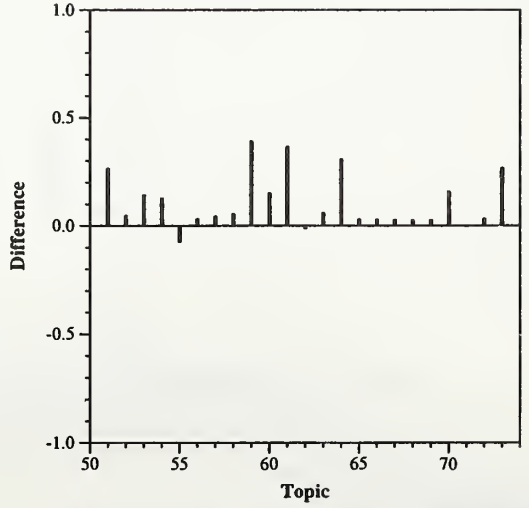
Summary Statistics	
Run Number	cuhtk-r1.ret
Run Description	reference
Number of Topics	23
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	367

Recall Level Precision Averages	
Recall	Precision
0.00	0.7580
0.10	0.7174
0.20	0.6795
0.30	0.6066
0.40	0.5620
0.50	0.5453
0.60	0.4646
0.70	0.3762
0.80	0.3149
0.90	0.2596
1.00	0.2038
Average precision over all relevant docs	
non-interpolated	0.4817

Document Level Averages	
	Precision
At 5 docs	0.6000
At 10 docs	0.4739
At 15 docs	0.4058
At 20 docs	0.3587
At 30 docs	0.2855
At 100 docs	0.1230
At 200 docs	0.0696
At 500 docs	0.0317
At 1000 docs	0.0160
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4603



Recall-Precision Curve

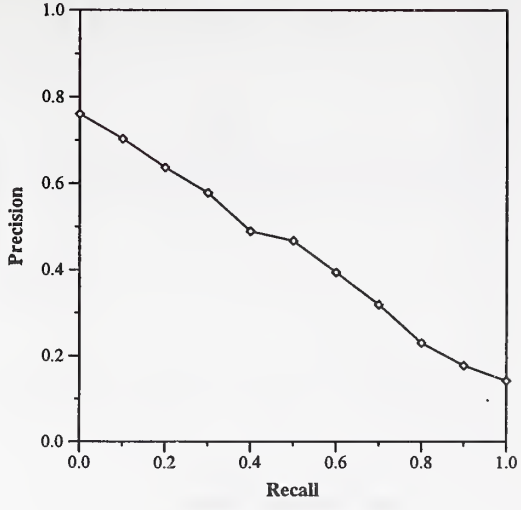


Difference from Median in Average Precision per Topic

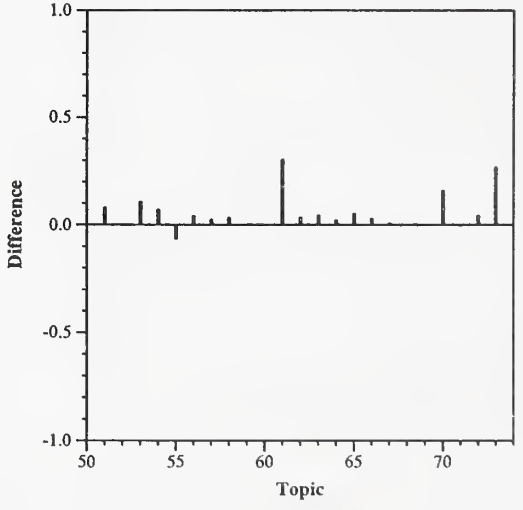
Summary Statistics	
Run Number	cuhtk-b1.ret
Run Description	baseline1
Number of Topics	23
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	355

Recall Level Precision Averages	
Recall	Precision
0.00	0.7607
0.10	0.7036
0.20	0.6370
0.30	0.5785
0.40	0.4894
0.50	0.4670
0.60	0.3941
0.70	0.3194
0.80	0.2301
0.90	0.1776
1.00	0.1418
Average precision over all relevant docs	
non-interpolated	0.4272

Document Level Averages	
	Precision
At 5 docs	0.5478
At 10 docs	0.4261
At 15 docs	0.3681
At 20 docs	0.3457
At 30 docs	0.2681
At 100 docs	0.1148
At 200 docs	0.0657
At 500 docs	0.0303
At 1000 docs	0.0154
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4187



Recall-Precision Curve

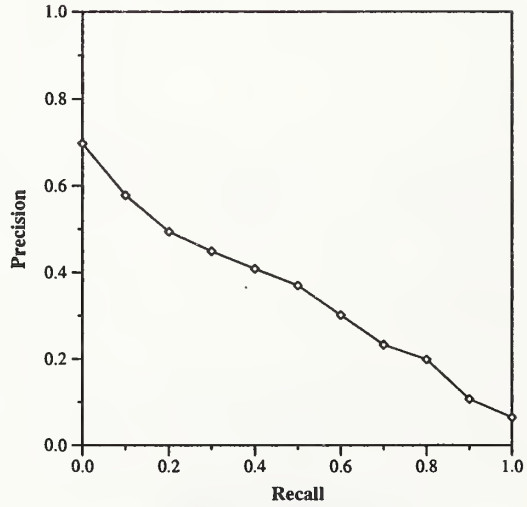


Difference from Median in Average Precision per Topic

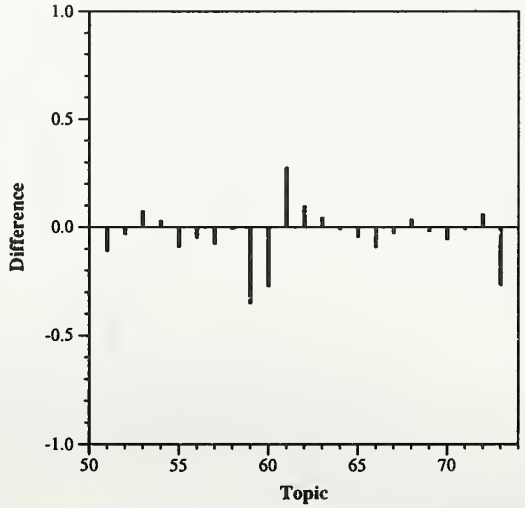
Summary Statistics	
Run Number	cuhtk-b2.ret
Run Description	baseline2
Number of Topics	23
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	351

Recall Level Precision Averages	
Recall	Precision
0.00	0.6974
0.10	0.5785
0.20	0.4944
0.30	0.4492
0.40	0.4087
0.50	0.3703
0.60	0.3022
0.70	0.2334
0.80	0.1993
0.90	0.1073
1.00	0.0650
Average precision over all relevant docs	
non-interpolated	0.3352

Document Level Averages	
	Precision
At 5 docs	0.4348
At 10 docs	0.3826
At 15 docs	0.3275
At 20 docs	0.2978
At 30 docs	0.2319
At 100 docs	0.1074
At 200 docs	0.0630
At 500 docs	0.0295
At 1000 docs	0.0153
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3619



Recall-Precision Curve

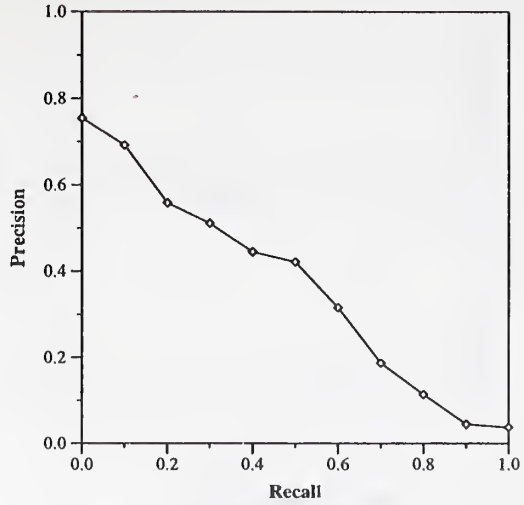


Difference from Median in Average Precision per Topic

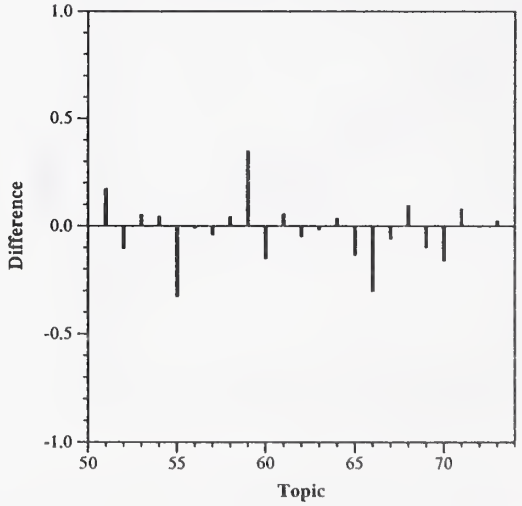
Summary Statistics	
Run Number	cuhtk-cr-derasru1.ret
Run Description	recognizer
Number of Topics	23
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	332

Recall Level Precision Averages	
Recall	Precision
0.00	0.7549
0.10	0.6926
0.20	0.5587
0.30	0.5114
0.40	0.4454
0.50	0.4219
0.60	0.3159
0.70	0.1872
0.80	0.1141
0.90	0.0447
1.00	0.0374
Average precision over all relevant docs	
non-interpolated	0.3521

Document Level Averages	
	Precision
At 5 docs	0.5478
At 10 docs	0.3957
At 15 docs	0.3449
At 20 docs	0.3065
At 30 docs	0.2435
At 100 docs	0.1009
At 200 docs	0.0574
At 500 docs	0.0270
At 1000 docs	0.0144
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3806



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run Number	cuhtk-cr-shef1.ret
Run Description	recognizer
Number of Topics	23

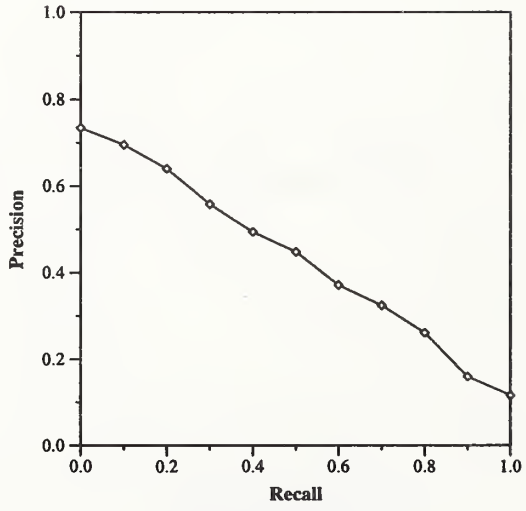
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	361

Recall Level Precision Averages	
Recall	Precision
0.00	0.7345
0.10	0.6956
0.20	0.6404
0.30	0.5586
0.40	0.4944
0.50	0.4484
0.60	0.3715
0.70	0.3246
0.80	0.2615
0.90	0.1598
1.00	0.1164

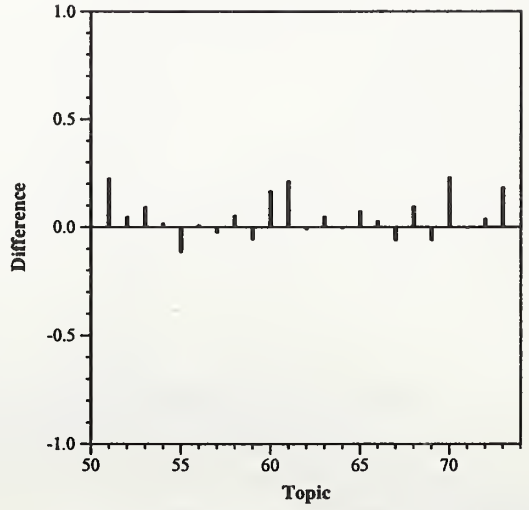
Average precision over all relevant docs	
non-interpolated	0.4251

Document Level Averages	
	Precision
At 5 docs	0.5478
At 10 docs	0.4391
At 15 docs	0.3797
At 20 docs	0.3326
At 30 docs	0.2681
At 100 docs	0.1143
At 200 docs	0.0674
At 500 docs	0.0308
At 1000 docs	0.0157

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4015



Recall-Precision Curve

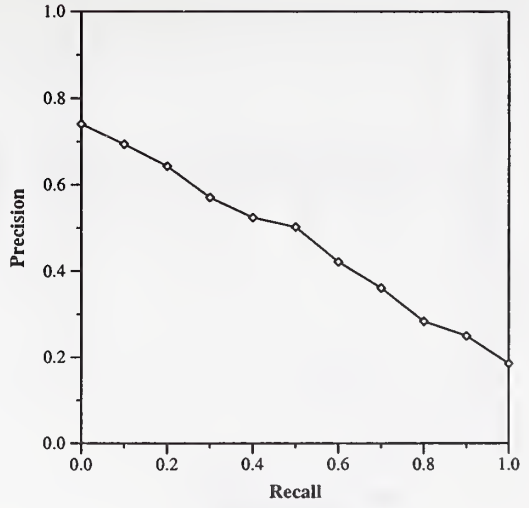


Difference from Median in Average Precision per Topic

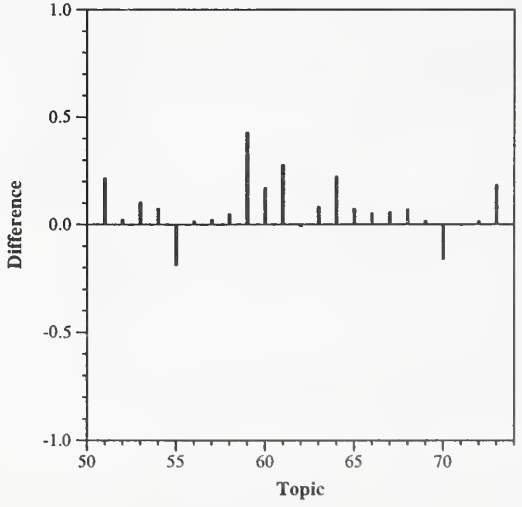
Summary Statistics	
Run Number	cuhtk-s1.ret
Run Description	recognizer
Number of Topics	23
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	368

Recall Level Precision Averages	
Recall	Precision
0.00	0.7408
0.10	0.6942
0.20	0.6433
0.30	0.5711
0.40	0.5244
0.50	0.5026
0.60	0.4220
0.70	0.3614
0.80	0.2839
0.90	0.2504
1.00	0.1857
Average precision over all relevant docs	
non-interpolated	0.4509

Document Level Averages	
	Precision
At 5 docs	0.5565
At 10 docs	0.4522
At 15 docs	0.3971
At 20 docs	0.3522
At 30 docs	0.2855
At 100 docs	0.1209
At 200 docs	0.0683
At 500 docs	0.0313
At 1000 docs	0.0160
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4330



Recall-Precision Curve

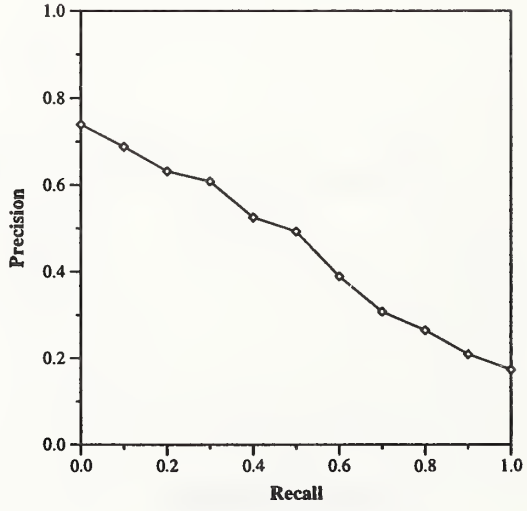


Difference from Median in Average Precision per Topic

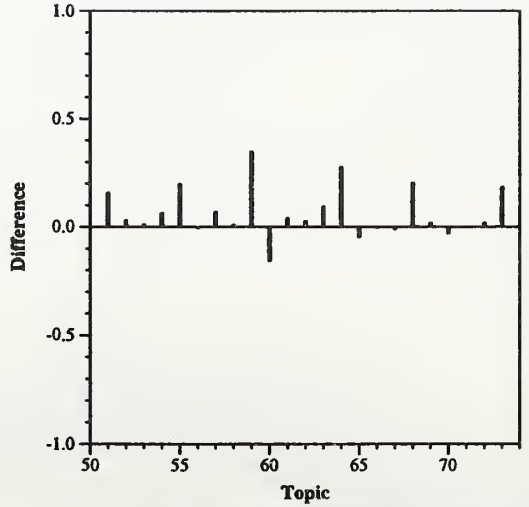
Summary Statistics	
Run Number	umd-r1.ret
Run Description	reference
Number of Topics	23
Total number of documents over all topics	
Retrieved:	19142
Relevant:	390
Rel-ret:	361

Recall Level Precision Averages	
Recall	Precision
0.00	0.7396
0.10	0.6884
0.20	0.6321
0.30	0.6085
0.40	0.5255
0.50	0.4932
0.60	0.3894
0.70	0.3079
0.80	0.2652
0.90	0.2097
1.00	0.1739
Average precision over all relevant docs	
non-interpolated	0.4386

Document Level Averages	
	Precision
At 5 docs	0.5304
At 10 docs	0.4304
At 15 docs	0.3884
At 20 docs	0.3435
At 30 docs	0.2681
At 100 docs	0.1152
At 200 docs	0.0641
At 500 docs	0.0308
At 1000 docs	0.0157
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4152



Recall-Precision Curve



Difference from Median in Average Precision per Topic

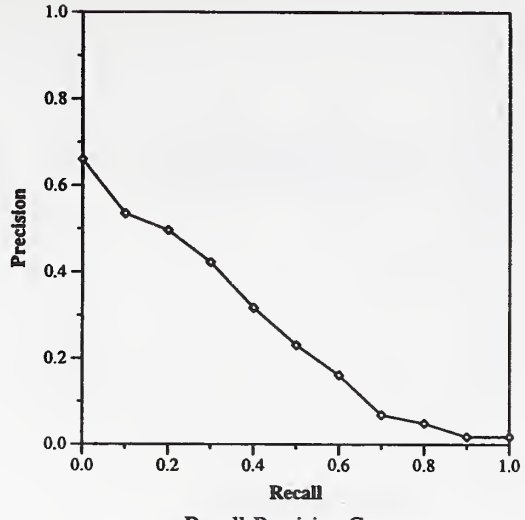
Summary Statistics	
Run Number	umd-b1.ret
Run Description	baseline1
Number of Topics	23
Total number of documents over all topics	
Retrieved:	18020
Relevant:	390
Rel-ret:	223

Recall Level Precision Averages	
Recall	Precision
0.00	0.6605
0.10	0.5351
0.20	0.4965
0.30	0.4230
0.40	0.3173
0.50	0.2311
0.60	0.1611
0.70	0.0685
0.80	0.0491
0.90	0.0180
1.00	0.0180

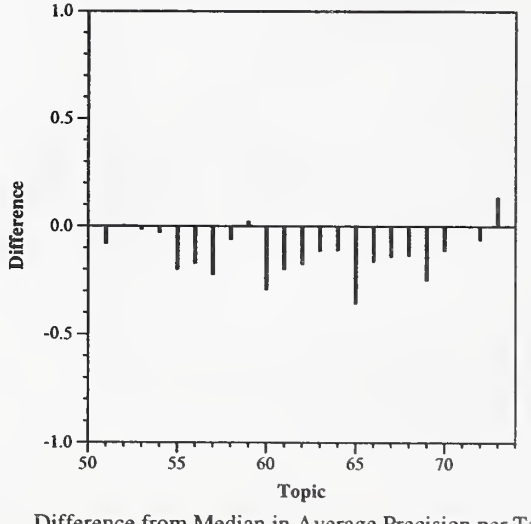
Average precision over all relevant docs	
non-interpolated	0.2557

Document Level Averages	
	Precision
At 5 docs	0.4000
At 10 docs	0.3130
At 15 docs	0.2638
At 20 docs	0.2304
At 30 docs	0.1754
At 100 docs	0.0743
At 200 docs	0.0413
At 500 docs	0.0190
At 1000 docs	0.0097

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3010



Recall-Precision Curve



Difference from Median in Average Precision per Topic

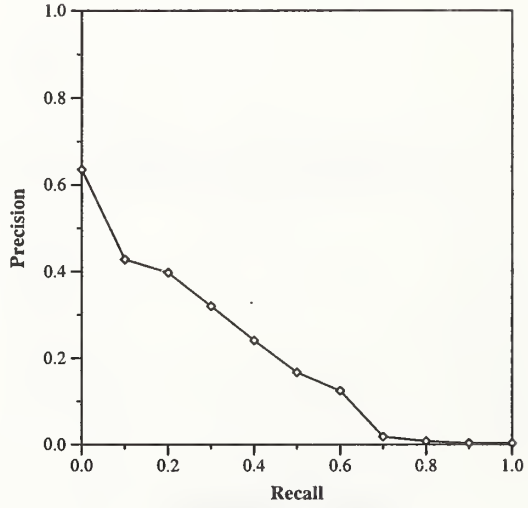
Summary Statistics	
Run Number	umd-b2.ret
Run Description	baseline2
Number of Topics	23
Total number of documents over all topics	
Retrieved:	18719
Relevant:	390
Rel-ret:	226

Recall Level Precision Averages	
Recall	Precision
0.00	0.6356
0.10	0.4284
0.20	0.3980
0.30	0.3204
0.40	0.2413
0.50	0.1673
0.60	0.1248
0.70	0.0184
0.80	0.0080
0.90	0.0032
1.00	0.0032

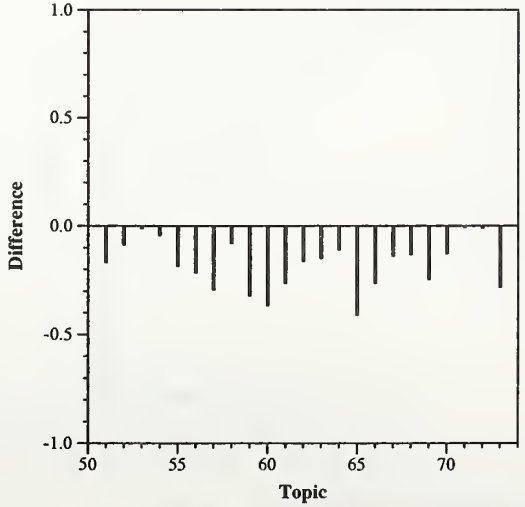
Average precision over all relevant docs	
non-interpolated	0.1967

Document Level Averages	
	Precision
At 5 docs	0.3130
At 10 docs	0.2565
At 15 docs	0.2290
At 20 docs	0.1935
At 30 docs	0.1551
At 100 docs	0.0696
At 200 docs	0.0393
At 500 docs	0.0186
At 1000 docs	0.0098

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2407



Recall-Precision Curve

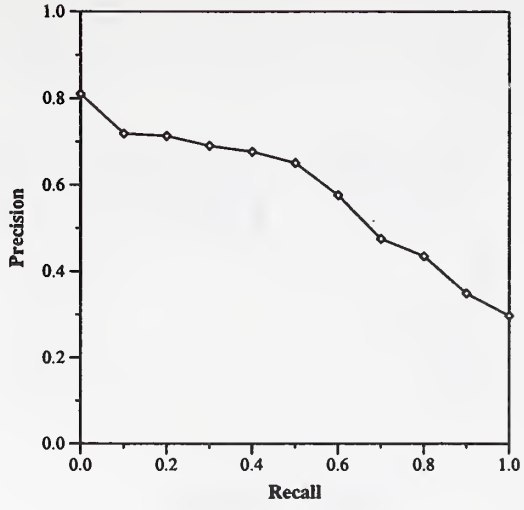


Difference from Median in Average Precision per Topic

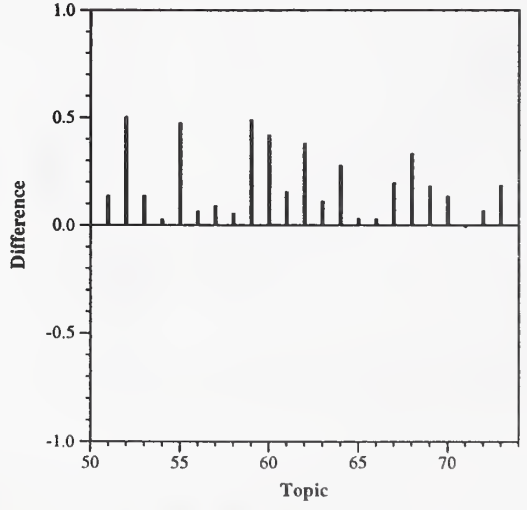
Summary Statistics	
Run Number	umass-r1.ret
Run Description	reference
Number of Topics	23
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	382

Recall Level Precision Averages	
Recall	Precision
0.00	0.8104
0.10	0.7192
0.20	0.7136
0.30	0.6909
0.40	0.6772
0.50	0.6512
0.60	0.5765
0.70	0.4755
0.80	0.4356
0.90	0.3492
1.00	0.2974
Average precision over all relevant docs	
non-interpolated	0.5668

Document Level Averages	
	Precision
At 5 docs	0.6000
At 10 docs	0.5217
At 15 docs	0.4435
At 20 docs	0.4087
At 30 docs	0.3362
At 100 docs	0.1352
At 200 docs	0.0741
At 500 docs	0.0320
At 1000 docs	0.0166
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.5096



Recall-Precision Curve

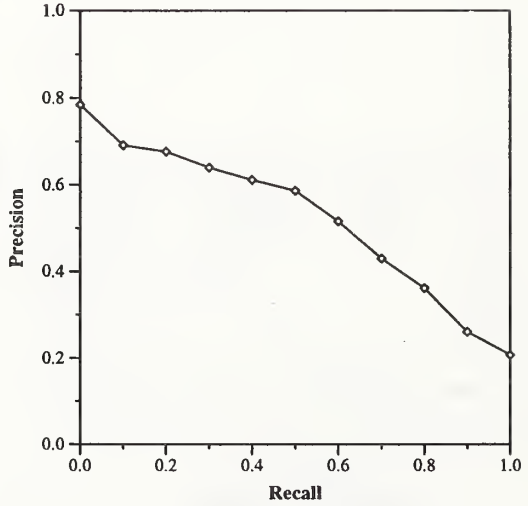


Difference from Median in Average Precision per Topic

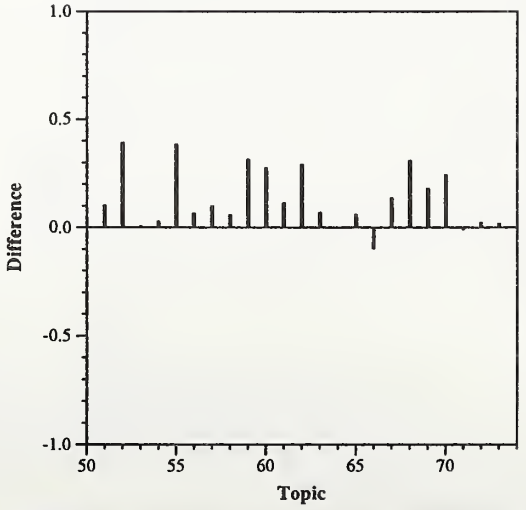
Summary Statistics	
Run Number	umass-b1.ret
Run Description	baseline1
Number of Topics	23
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	378

Recall Level Precision Averages	
Recall	Precision
0.00	0.7845
0.10	0.6914
0.20	0.6766
0.30	0.6399
0.40	0.6114
0.50	0.5866
0.60	0.5158
0.70	0.4300
0.80	0.3617
0.90	0.2607
1.00	0.2074
Average precision over all relevant docs	
non-interpolated	0.5063

Document Level Averages	
	Precision
At 5 docs	0.5478
At 10 docs	0.4739
At 15 docs	0.4145
At 20 docs	0.3826
At 30 docs	0.3072
At 100 docs	0.1300
At 200 docs	0.0722
At 500 docs	0.0317
At 1000 docs	0.0164
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4709



Recall-Precision Curve

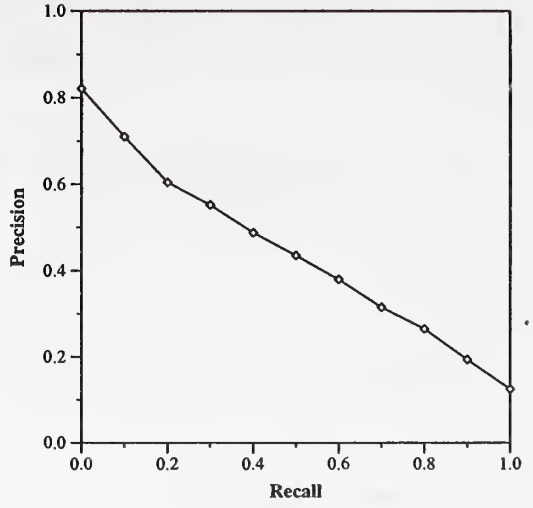


Difference from Median in Average Precision per Topic

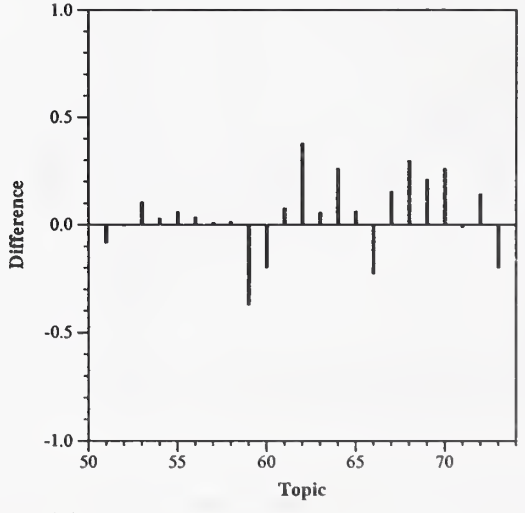
Summary Statistics	
Run Number	umass-b2.ret
Run Description	baseline2
Number of Topics	23
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	369

Recall Level Precision Averages	
Recall	Precision
0.00	0.8213
0.10	0.7100
0.20	0.6046
0.30	0.5522
0.40	0.4880
0.50	0.4353
0.60	0.3796
0.70	0.3150
0.80	0.2651
0.90	0.1937
1.00	0.1252
Average precision over all relevant docs	
non-interpolated	0.4191

Document Level Averages	
	Precision
At 5 docs	0.4870
At 10 docs	0.4000
At 15 docs	0.3652
At 20 docs	0.3326
At 30 docs	0.2710
At 100 docs	0.1204
At 200 docs	0.0687
At 500 docs	0.0306
At 1000 docs	0.0160
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4052



Recall-Precision Curve

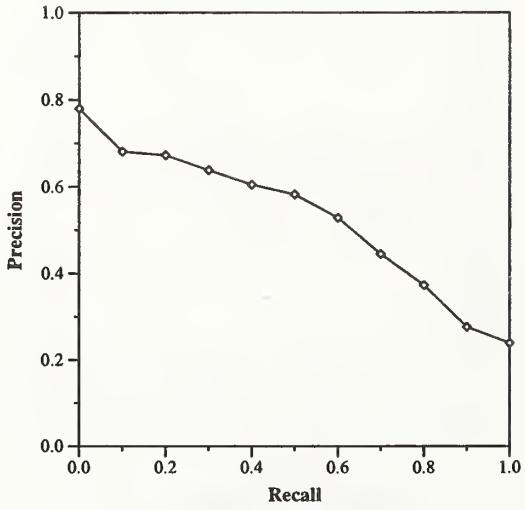


Difference from Median in Average Precision per Topic

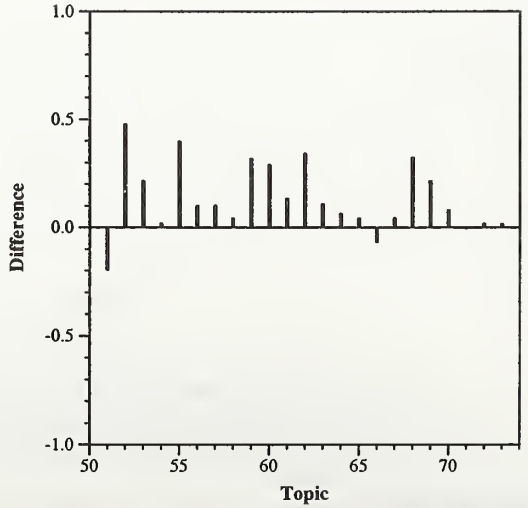
Summary Statistics	
Run Number	umass-s1-dragon.ret
Run Description	recognizer
Number of Topics	23
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	378

Recall Level Precision Averages	
Recall	Precision
0.00	0.7802
0.10	0.6813
0.20	0.6729
0.30	0.6386
0.40	0.6052
0.50	0.5825
0.60	0.5288
0.70	0.4452
0.80	0.3733
0.90	0.2765
1.00	0.2394
Average precision over all relevant docs	
non-interpolated	0.5075

Document Level Averages	
	Precision
At 5 docs	0.5652
At 10 docs	0.4957
At 15 docs	0.4174
At 20 docs	0.3761
At 30 docs	0.2928
At 100 docs	0.1330
At 200 docs	0.0737
At 500 docs	0.0320
At 1000 docs	0.0164
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4850



Recall-Precision Curve



Difference from Median in Average Precision per Topic

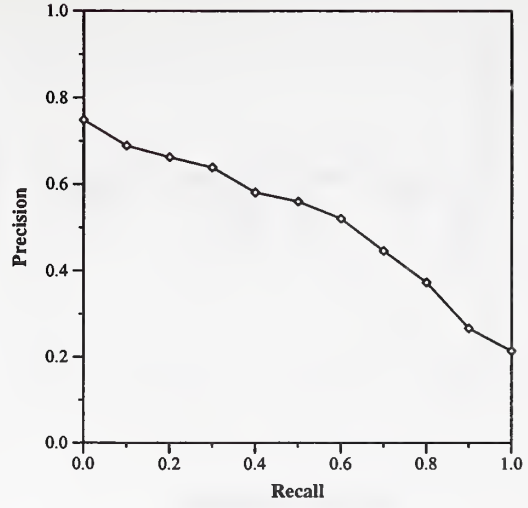
Summary Statistics	
Run Number	umass-s2-dragon.ret
Run Description	recognizer
Number of Topics	23
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	372

Recall Level Precision Averages	
Recall	Precision
0.00	0.7489
0.10	0.6893
0.20	0.6629
0.30	0.6389
0.40	0.5810
0.50	0.5602
0.60	0.5204
0.70	0.4458
0.80	0.3729
0.90	0.2665
1.00	0.2138

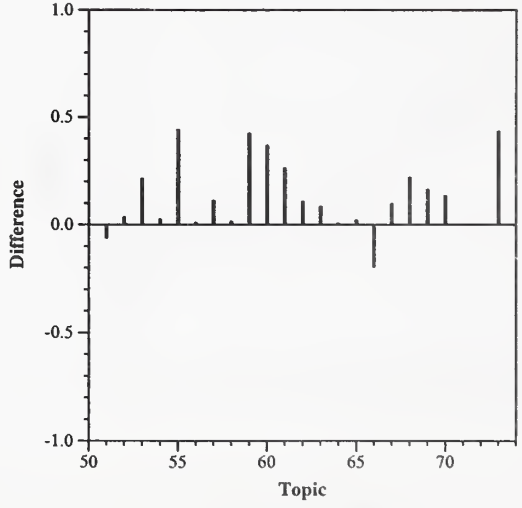
Average precision over all relevant docs	
non-interpolated	0.5000

Document Level Averages	
	Precision
At 5 docs	0.5130
At 10 docs	0.4435
At 15 docs	0.3913
At 20 docs	0.3478
At 30 docs	0.2986
At 100 docs	0.1309
At 200 docs	0.0737
At 500 docs	0.0315
At 1000 docs	0.0162

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4738



Recall-Precision Curve

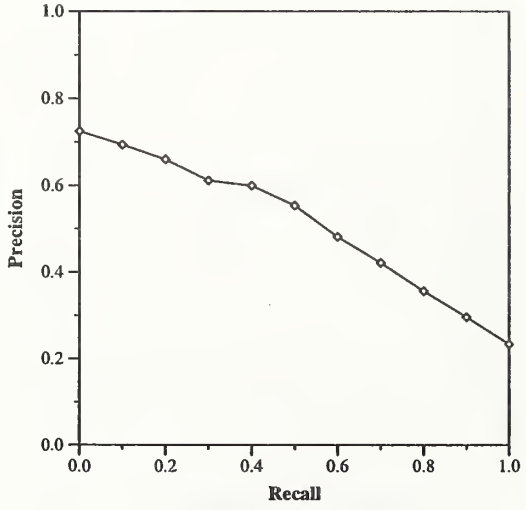


Difference from Median in Average Precision per Topic

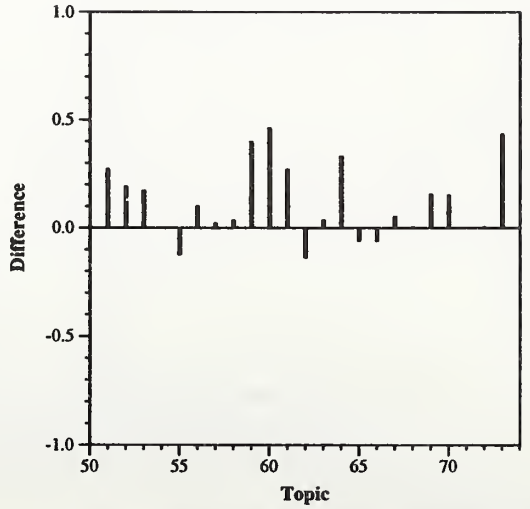
Summary Statistics	
Run Number	shef-r1.ret
Run Description	reference
Number of Topics	23
Total number of documents over all topics	
Retrieved:	17590
Relevant:	390
Rel-ret:	364

Recall Level Precision Averages	
Recall	Precision
0.00	0.7249
0.10	0.6941
0.20	0.6602
0.30	0.6115
0.40	0.5997
0.50	0.5540
0.60	0.4818
0.70	0.4218
0.80	0.3565
0.90	0.2965
1.00	0.2338
Average precision over all relevant docs	
non-interpolated	0.4916

Document Level Averages	
	Precision
At 5 docs	0.5217
At 10 docs	0.4522
At 15 docs	0.4087
At 20 docs	0.3652
At 30 docs	0.3000
At 100 docs	0.1274
At 200 docs	0.0689
At 500 docs	0.0311
At 1000 docs	0.0158
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4777



Recall-Precision Curve

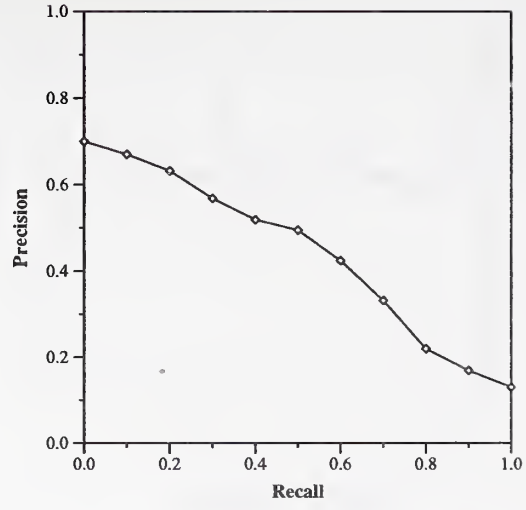


Difference from Median in Average Precision per Topic

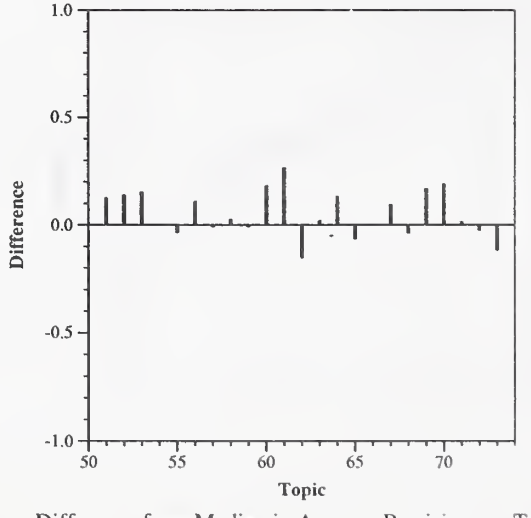
Summary Statistics	
Run Number	shef-b1.ret
Run Description	baseline1
Number of Topics	23
Total number of documents over all topics	
Retrieved:	18165
Relevant:	390
Rel-ret:	360

Recall Level Precision Averages	
Recall	Precision
0.00	0.7001
0.10	0.6705
0.20	0.6325
0.30	0.5685
0.40	0.5192
0.50	0.4951
0.60	0.4242
0.70	0.3316
0.80	0.2195
0.90	0.1691
1.00	0.1302
Average precision over all relevant docs	
non-interpolated	0.4243

Document Level Averages	
	Precision
At 5 docs	0.5304
At 10 docs	0.4348
At 15 docs	0.3652
At 20 docs	0.3348
At 30 docs	0.2754
At 100 docs	0.1248
At 200 docs	0.0674
At 500 docs	0.0301
At 1000 docs	0.0157
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4396



Recall-Precision Curve

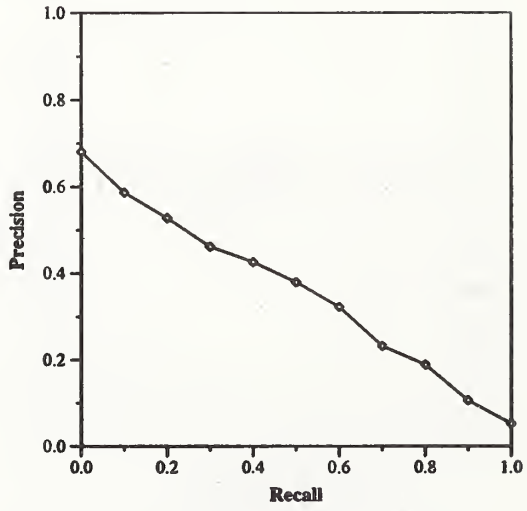


Difference from Median in Average Precision per Topic

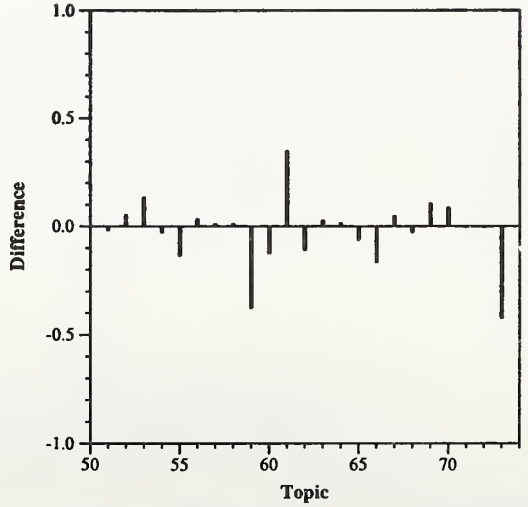
Summary Statistics	
Run Number	shef-b2.ret
Run Description	baseline2
Number of Topics	23
Total number of documents over all topics	
Retrieved:	18727
Relevant:	390
Rel-ret:	357

Recall Level Precision Averages	
Recall	Precision
0.00	0.6811
0.10	0.5868
0.20	0.5283
0.30	0.4622
0.40	0.4265
0.50	0.3800
0.60	0.3230
0.70	0.2325
0.80	0.1892
0.90	0.1065
1.00	0.0524
Average precision over all relevant docs	
non-interpolated	0.3471

Document Level Averages	
	Precision
At 5 docs	0.4522
At 10 docs	0.3696
At 15 docs	0.3188
At 20 docs	0.2891
At 30 docs	0.2464
At 100 docs	0.1135
At 200 docs	0.0641
At 500 docs	0.0302
At 1000 docs	0.0155
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3471



Recall-Precision Curve



Difference from Median in Average Precision per Topic

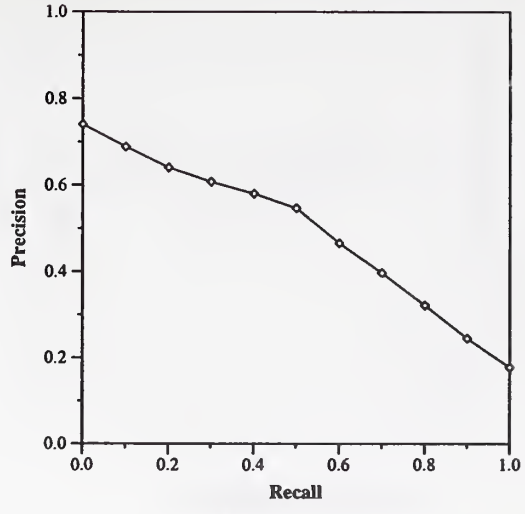
Summary Statistics	
Run Number	shef-cr-htk.ret
Run Description	recognizer
Number of Topics	23
Total number of documents over all topics	
Retrieved:	18176
Relevant:	390
Rel-ret:	366

Recall Level Precision Averages	
Recall	Precision
0.00	0.7404
0.10	0.6888
0.20	0.6409
0.30	0.6077
0.40	0.5803
0.50	0.5465
0.60	0.4657
0.70	0.3969
0.80	0.3218
0.90	0.2451
1.00	0.1779

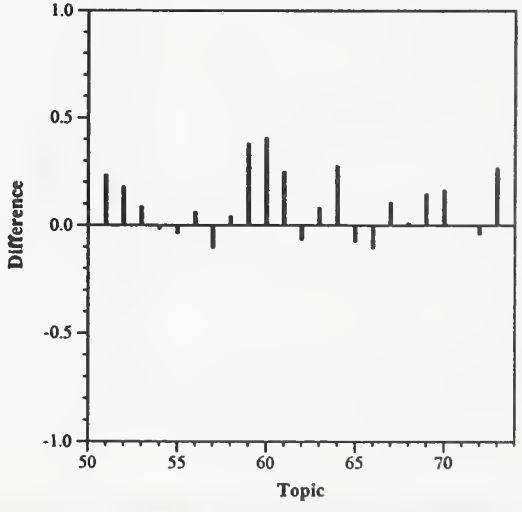
Average precision over all relevant docs	
non-interpolated	0.4713

Document Level Averages	
	Precision
At 5 docs	0.5565
At 10 docs	0.4652
At 15 docs	0.3971
At 20 docs	0.3565
At 30 docs	0.2942
At 100 docs	0.1257
At 200 docs	0.0685
At 500 docs	0.0309
At 1000 docs	0.0159

R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4549



Recall-Precision Curve

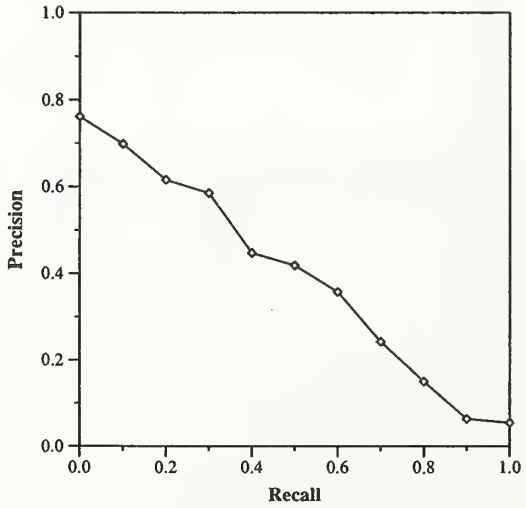


Difference from Median in Average Precision per Topic

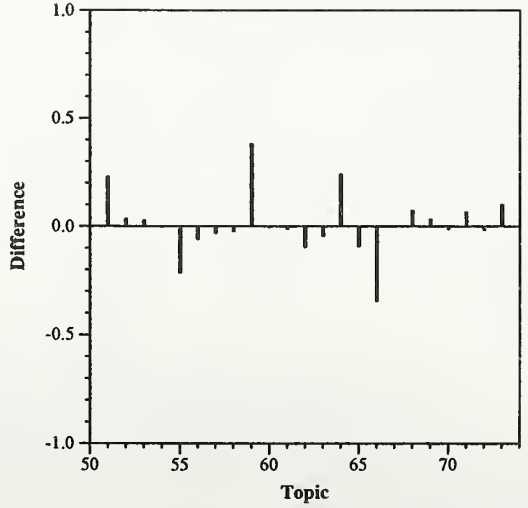
Summary Statistics	
Run Number	shef-cr-sru.ret
Run Description	recognizer
Number of Topics	23
Total number of documents over all topics	
Retrieved:	17869
Relevant:	390
Rel-ret:	335

Recall Level Precision Averages	
Recall	Precision
0.00	0.7617
0.10	0.6986
0.20	0.6159
0.30	0.5855
0.40	0.4476
0.50	0.4187
0.60	0.3578
0.70	0.2424
0.80	0.1503
0.90	0.0637
1.00	0.0547
Average precision over all relevant docs	
non-interpolated	0.3836

Document Level Averages	
	Precision
At 5 docs	0.4957
At 10 docs	0.3913
At 15 docs	0.3449
At 20 docs	0.3109
At 30 docs	0.2507
At 100 docs	0.1070
At 200 docs	0.0576
At 500 docs	0.0270
At 1000 docs	0.0146
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4158



Recall-Precision Curve

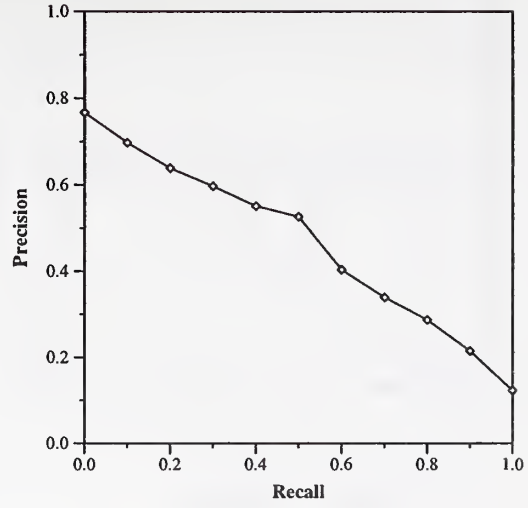


Difference from Median in Average Precision per Topic

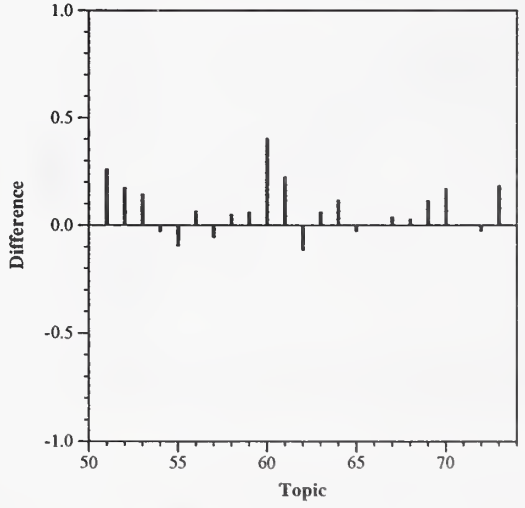
Summary Statistics	
Run Number	shef-s1.ret
Run Description	recognizer
Number of Topics	23
Total number of documents over all topics	
Retrieved:	18322
Relevant:	390
Rel-ret:	362

Recall Level Precision Averages	
Recall	Precision
0.00	0.7671
0.10	0.6976
0.20	0.6388
0.30	0.5971
0.40	0.5511
0.50	0.5265
0.60	0.4038
0.70	0.3395
0.80	0.2875
0.90	0.2151
1.00	0.1232
Average precision over all relevant docs	
non-interpolated	0.4495

Document Level Averages	
	Precision
At 5 docs	0.5478
At 10 docs	0.4522
At 15 docs	0.3681
At 20 docs	0.3478
At 30 docs	0.2797
At 100 docs	0.1235
At 200 docs	0.0670
At 500 docs	0.0303
At 1000 docs	0.0157
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.4284



Recall-Precision Curve

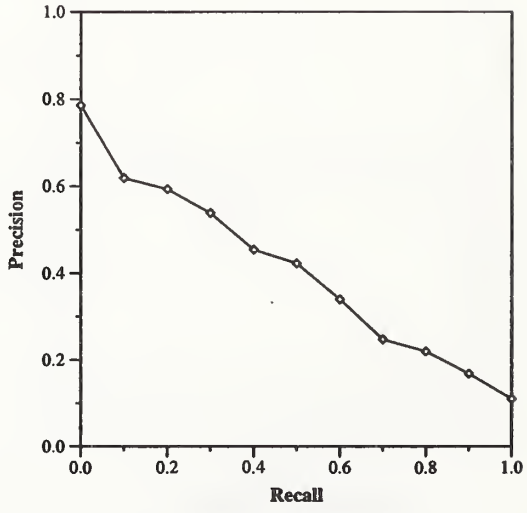


Difference from Median in Average Precision per Topic

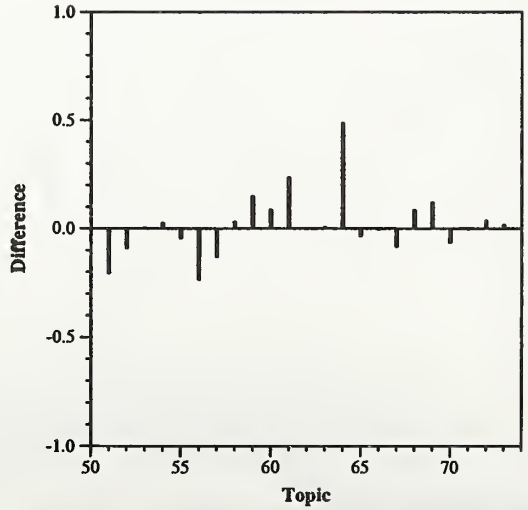
Summary Statistics	
Run Number	nsa-r1.ret
Run Description	reference
Number of Topics	23
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	335

Recall Level Precision Averages	
Recall	Precision
0.00	0.7866
0.10	0.6195
0.20	0.5937
0.30	0.5387
0.40	0.4544
0.50	0.4228
0.60	0.3399
0.70	0.2476
0.80	0.2199
0.90	0.1687
1.00	0.1105
Average precision over all relevant docs	
non-interpolated	0.3907

Document Level Averages	
	Precision
At 5 docs	0.4870
At 10 docs	0.3478
At 15 docs	0.2957
At 20 docs	0.2696
At 30 docs	0.2275
At 100 docs	0.1161
At 200 docs	0.0622
At 500 docs	0.0277
At 1000 docs	0.0146
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3703



Recall-Precision Curve

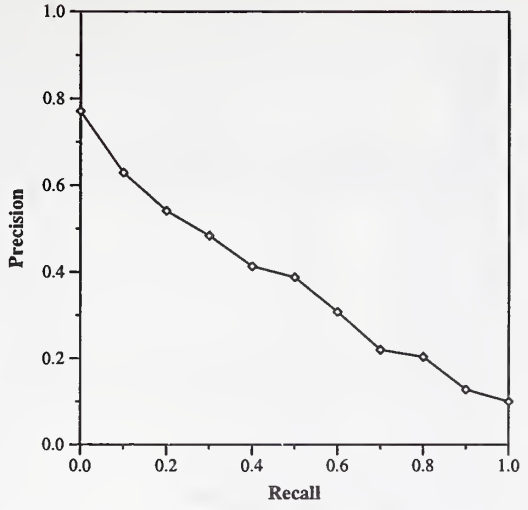


Difference from Median in Average Precision per Topic

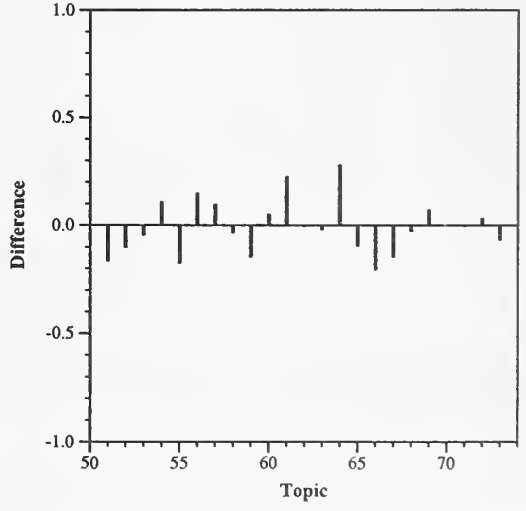
Summary Statistics	
Run Number	nsa-b1.ret
Run Description	baseline1
Number of Topics	23
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	328

Recall Level Precision Averages	
Recall	Precision
0.00	0.7713
0.10	0.6294
0.20	0.5414
0.30	0.4839
0.40	0.4133
0.50	0.3880
0.60	0.3080
0.70	0.2202
0.80	0.2038
0.90	0.1281
1.00	0.1002
Average precision over all relevant docs	
non-interpolated	0.3640

Document Level Averages	
	Precision
At 5 docs	0.4870
At 10 docs	0.3783
At 15 docs	0.3188
At 20 docs	0.2717
At 30 docs	0.2362
At 100 docs	0.1048
At 200 docs	0.0583
At 500 docs	0.0269
At 1000 docs	0.0143
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.3442



Recall-Precision Curve

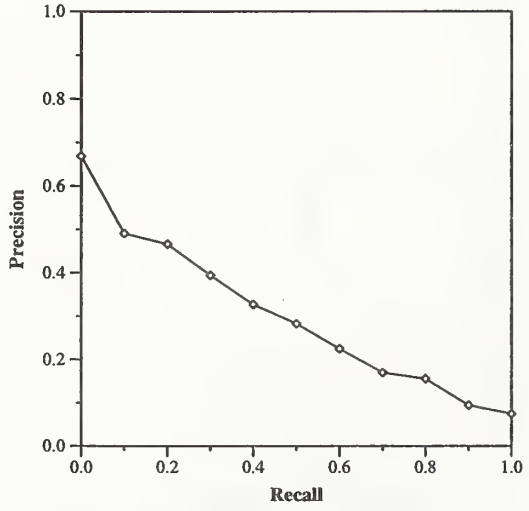


Difference from Median in Average Precision per Topic

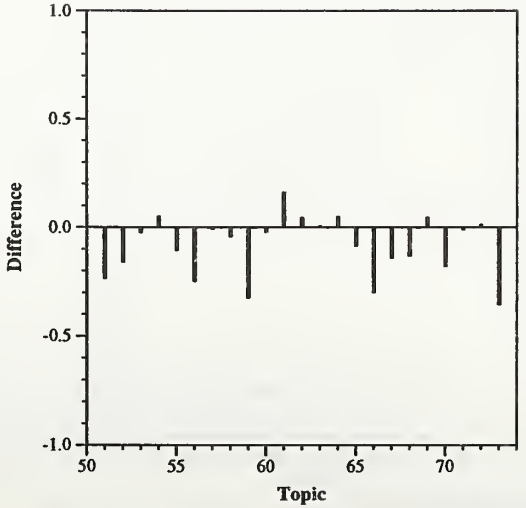
Summary Statistics	
Run Number	nsa-b2.ret
Run Description	baseline2
Number of Topics	23
Total number of documents over all topics	
Retrieved:	23000
Relevant:	390
Rel-ret:	332

Recall Level Precision Averages	
Recall	Precision
0.00	0.6688
0.10	0.4909
0.20	0.4662
0.30	0.3946
0.40	0.3271
0.50	0.2830
0.60	0.2249
0.70	0.1701
0.80	0.1558
0.90	0.0946
1.00	0.0748
Average precision over all relevant docs	
non-interpolated	0.2868

Document Level Averages	
	Precision
At 5 docs	0.3913
At 10 docs	0.2957
At 15 docs	0.2667
At 20 docs	0.2370
At 30 docs	0.2000
At 100 docs	0.0983
At 200 docs	0.0570
At 500 docs	0.0270
At 1000 docs	0.0144
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2475



Recall-Precision Curve



Difference from Median in Average Precision per Topic

SUMMARY PERFORMANCE COMPARISONS TREC-2 THROUGH TREC-7

Karen Sparck Jones
Computer Laboratory, University of Cambridge

December 22, 1998

The context

These comparisons continue my attempt to illustrate long-term performance trends in TREC. My last comparisons, for TREC-2 - 6, appeared as the final Appendix in the TREC-6 Proceedings. This year the tables are confined to adhoc performance, as routing has become less conspicuous in TREC. I have taken the opportunity to include a few minor corrections of earlier tables.

Over TREC as a whole there have been some important changes relating to a major variable, namely the topics (requests). First, their composition has changed; and second the conditions on their treatment for officially submitted runs have changed. Table 1 shows (a) the component fields in the topics in successive TRECs and (b) field lengths in successive TRECs. It also defines different topic *versions* as specified for different runs, ranging from Very short, titles only, to Long, covering title, description and narrative fields.

In earlier TRECs, up to TREC-4, automatic and manual *modes* of query formulation were treated simply as low-level optional alternatives, so in my earlier comparative tables I used the best performing run for each team regardless of mode. In TREC-5, and since, the modes have been treated as distinct conditions. At the same time there have been some changes, during TREC, in the definition of what is allowed in manual searching. These variations in test data and condition mean that overall trend comparisons can only be rather general. Because the search mode has become more important, runs for TREC-2 - 4 are now marked according to whether they were automatic or manual. For TREC-5 - 7 the modes are listed separately. Thus the main performance tables for these TRECs illustrate the pairings of topic version with search mode e.g. Very short with auto, Long with manual.

Table entries

The detailed figures for TREC-4 onwards are taken from the Conference Working Notes. They cover only Category A runs, and only higher levels of performance, not all runs.

The conventions are as follows: figures are not rounded; performance is assigned to 'blocks'; teams per block are NOT in merit order, but in in Working Notes results order; where there is more than one run per team the best is taken, regardless of the particular strategy used. Simple, hopefully sufficiently identifiable, short names have been given to the teams (with some streamlining where teams have changed name or composition over the years).

TREC ADHOC SEARCH RESULTS FOR PRECISION AT DOCUMENT CUTOFF 30

KEY TO TABLE NOTATIONS :

a = fully automatic searches
m = manual searches
- = manual searches in TREC 2 - 4

Topic fields available as base for queries :

	(TREC-1)	TREC-2	TREC-3	TREC-4	TREC-5	TREC-6	TREC-7
T = title	x	x	x		x	x	x
D = description	x	x	x	x	x	x	x
N = narrative	x	x	x		x	x	x
C = concepts	x	x					

Average topic and field length :

Total	107.4	130.8	103.4	16.3	82.7	88.4	57.6
T	3.8	4.9	6.5	-	3.8	2.7	2.5
D	17.9	18.7	22.3	16.3	15.7	20.4	14.3
N	64.5	78.8	74.6	-	63.2	65.3	40.8
C	21.2	28.5	-	-	-	-	-

TREC 2 - 4 did not distinguish queries by any specific sets of topic fields
TREC 5 - 7 distinguished runs by different sets of fields

V = very short queries, i.e.	title only from topics, aka	T
S = short queries	description only	D
M = medium queries	title+description	T+D
L = long queries	title+description+narrative	T+D+N

	TREC-2 a/m	TREC-3 a/m	TREC-4 a/m	TREC-5 a S	TREC-5 a L	TREC-5 m L
>= 60		-UMass City -Berkeley				
>= 55	-UMass -HNC -VT	Cornell -Mead				
>= 50	Cornell Berkeley Dortmund -CMU/Clarit -Verity -Siemens CUNY	-Verity -VT Westlaw ETH CUNY				
>= 45	-City Bellcore ETH CITRI/RMIT -Conquest	NYU CMU/Clarit RMIT -RutgersK	-Excalibur/ Conquest -CUNY -Waterloo			ETH
>= 40	...	...	-Berkeley -Clarit/CMU Cornell -GMU -UMass -InText -ANU			Waterloo
>= 35	...	...	City -GE/NYU			ANU Clarit Cornell GE/NYU GMUetc Lexis
>= 30	...	...	...		City CUNY ETH	OpenText CUNY Berkeley
>= 25	...	...	...	Apple City Cornell IBMTJW	Apple GE/NYU RMIT Berkeley	DCU IBM
>= 20	.....					

	TREC-6 a V	TREC-6 a S	TREC-6 a L	TREC-6 m L	TREC-7 a V	TREC-7 a S	TREC-7 a M	TREC-7 a L	TREC-7 m L
>=60	(best TREC 2-5)								
>=55									Clarit
>=50				Waterloo					ManInst Waterloo
>=45				Clarit					GMUetc
>=40				ANU		NEC	ATT Cityetc UMass	BBN Cityetc NEC UMass	ANU Harris Berkeley Toronto
>=35				GEetc Lexis	Cityetc	Cornell CUNY Fujitsu	Lexis RMIT	ANU Cornell CUNY TwentyO Iowa	GEetc Lexis IRIT
>=30	Apple ATT City IRIT Lexis CUNY Waterloo		ANU Cornell IRIT CUNY Berkeley	ISS Berkeley	ATT Cornell CUNY Fujitsu Lexis NEC NTTData RMIT Waterloo	IBMTJWs	IBMTJWg IRIT	GMUetc NTTData Rutgers Berkeley UNC	FS
>=25	DCU ISS	ATT ANU City Cornell GMUetc IBMTJWs IRIT Lexis Waterloo	City IBMTJWg MDS/RMIT UMass GMUetc	FS GMUetc	ANU Avignon GEetc IBMTJWg ETH Berkeley Maryland			FUB ImperC JHopk NSA	
>=20	MDS/RMIT Glasgow	Apple GEetc IBMTJWg MDS/RMIT CUNY Berkeley Maryland UMass Verity	Verity	Glasgow	FS GMUetc JHopk		Avignon ImperC MIT	NTHU	NTHU

Performance summary

Boiling down the larger tables for a summary picture of performance levels, I have taken the highest performance level reached for each version and mode over TREC-2 - 7 in the diagram below: the numbers refer to the corresponding TREC. This clearly shows high best levels of performance for TREC-2 and -3, and growing differences, for these respective top performers, between automatic and manual modes from TREC-4 onwards, generally reflecting less initial topic information along with more manual effort.

	V T a	S D a	M T+D a	L T+D+N a	L T+D+N m
>= 65					
>= 60				3333333	3333333
>= 55					222/777 +
>= 50				2222222 +	6666666
>= 45					444/555 -
>= 40		7777777	444/777	7777777	
>= 35	7777777				
>= 30	6666666			555/666	
>= 25		555/666			
>= 20					

Key: 222 = TREC-2 highest performance level, 333 = TREC-3 ditto, etc
+ TREC-2 also had Concept field
- TREC-4 did not have Narrative field

However it is important take the more detailed information of the main tables into consideration, as follows.

Overall comments

1. Many teams obtain similar performance, even at top levels.
2. Upper outliers are especially likely with manual mode, typically reflecting the amount of effort put into query development or user judgements on search output.
3. Though there has been some convergence on 'default' strategies, similar performance is obtained with very different strategies.
4. Performance trends over TREC clearly show the effects of data challenge, i.e. having less topic information or more difficult ('hard') topics: TREC-4 performance reflects the former, TREC-5 and -6 more the latter, since automatic performance is comparatively low regardless of query

version. Since TREC-7 full topics are shorter than TREC-6, but TREC-7 performance levels are better, the TREC-7 topics are presumably less hard.

5. However performance is not as tightly correlated with topic length, and specifically with version, as might be expected (setting aside the known-problematic TREC-6 descriptions). Thus similar good performance is obtained in automatic mode for different versions.

6. TREC-7 shows respectable absolute levels of automatic mode performance for intermediate length topics (Short and Medium), interestingly as good as for (the unrealistic) Long version; they are also comparable with all but the best manual mode. Performance with Very short is less good, but not negligible. Taking all other factors into consideration, these reasonable levels of performance with shorter versions of the topics must be primarily attributed, over TREC as a whole, to improvements in automatic mode methods.

U.S. GOVERNMENT PRINTING OFFICE: 1999 -- 454 - 531 / 17111

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 Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements are available from ACS, 1155 Sixteenth St., NW, Washington, DC 20056.

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