NATURAL LANGUAGE INFORMATION RETRIEVAL TREC-6 REPORT

TOMEK STRZALKOWSKI AND FANG LIN
GE Corporate Research & Development
Schenectady, NY 12301, USA

AND

JOSE PEREZ-CARBALLO
School of Communication, Information and Library Studies
Rutgers University
New Brunswick, NJ 04612

Abstract. Natural language processing techniques may hold a tremendous potential for overcoming the inadequacies of purely quantitative methods of text information retrieval, but the empirical evidence to support such predictions has thus far been inadequate, and appropriate scale evaluations have been slow to emerge. In this chapter, we report on the progress of the Natural Language Information Retrieval project, a joint effort of several sites led by GE Research, and its evaluation in the 6th Text Retrieval Conferences (TREC-6).

1. Introduction and Motivation

Recently, we noted a renewed interest in using NLP techniques in information retrieval, sparked in part by the sudden prominence, as well as the perceived limitations, of existing IR technology in rapidly emerging commercial applications, including on the Internet. This has also been reflected in what is being done at TREC: using phrasal terms and proper name annotations became a norm among TREC participants, and a special interest track on NLP took off for the first time in TREC-5.

In this paper we discuss particulars of the joint GE/Rutgers TREC-6 entry.

2. Stream-based Information Retrieval Model

The stream model was conceived to facilitate a thorough evaluation and optimization of various text content representation methods, including simple quantitative techniques as well as those requiring complex linguistic processing. Our system encompasses a number of statistical and natural language processing techniques that capture different aspects of document content: combining these into a coherent whole was in itself a major challenge. Therefore, we designed a distributed representation model in which alternative methods of document indexing (which we call "streams") are strung together to perform in parallel. Streams are built using a mixture of different indexing approaches, term extracting and weighting strategies, even different search engines.

The following term extraction steps correspond to some of the streams used in our system:

1. Elimination of stopwords: Original text words minus certain no-content and low-content stopwords are used to index documents. Included in the stopwords category are closed-class words such as determiners, prepositions, pronouns, etc., as well as certain very frequent words.
2. Morphological stemming: Words are normalized across morphological variants (e.g., "proliferation", "proliferate", "proliferating") using a lexicon-based stemmer. This is done by chopping off a suffix (*-ing, -s, -ment*) or by mapping onto root form in a lexicon (e.g., *proliferation* to *proliferate*).
3. Phrase extraction: Various shallow text processing techniques, such as part-of-speech tagging, phrase boundary detection, and word co-occurrence metrics are used to identify relatively stable groups of words, e.g., *joint venture.*
4. Phrase normalization: "Head+Modifier" pairs are identified in order to normalize across syntactic variants such as *weapon proliferation, proliferation of weapons, proliferate weapons*, etc., and reduce to a common "concept", e.g., **weapon+proliferate**.
5. Proper name extraction: Proper names are identified for indexing, including people names and titles, location names, organization names, etc.

The final results are produced by merging ranked lists of documents obtained from searching all streams with appropriately preprocessed queries, i.e., phrases for phrase stream, names for names stream, etc. The merging process weights contributions from each stream using a combination that was found the most effective in training runs. This allows for an easy combination of alternative retrieval and routing methods, creating a meta-search strategy which maximizes the contribution of each stream. Cornell's

SMART (Salton, 1989), Umass' Inquery (Croft et al., 19xx), and NIST's Prise (Harman & Candella, 1989) information retrieval systems were used as search engines for different streams.

Among the advantages of the stream architecture we may include the following:

- stream organization makes it easier to compare the contributions of different indexing features or representations. For example, it is easier to design experiments which allow us to decide if a certain representation adds information which is not contributed by other streams.
- it provides a convenient testbed to experiment with algorithms designed to merge the results obtained using different IR engines and/or techniques.
- it becomes easier to fine-tune the system in order to obtain optimum performance
- it allows us to use any combination of IR engines without having to adapt them in any way.

The notion of combining evidence from multiple sources is not new in information retrieval. Several researchers have noticed in the past that different systems may have similar performance but retrieve different documents, thus suggesting that they may complement one another. It has been reported that the use of different sources of evidence increases the performance of a hybrid system (see for example, (Callan et al., 1995); (Fox et al., 1993); (Saracevic and Kantor, 1988)). Nonetheless, the stream model used in our system is unique in that it explicitly addresses the issue of document representation as well as provides means for subsequent optimization.

3. Advanced Linguistic Streams

3.1. HEAD+MODIFIER PAIRS STREAM

Our linguistically most advanced stream is the head+modifier pairs stream. In this stream, documents are reduced to collections of word pairs derived via syntactic analysis of text followed by a normalization process intended to capture semantic uniformity across a variety of surface forms, e.g., "information retrieval", "retrieval of information", "retrieve more information", "information that is retrieved", etc. are all reduced to "retrieve+information" pair, where "retrieve" is a head or operator, and "information" is a modifier or argument. It has to be noted that while the head-modifier relation may suggest semantic dependence, what we obtain here is strictly syntactic, even though the semantic relation is what we are really after. This means in particular that the inferences of the kind where a *head+modifier* is taken as a specialized instance of *head*, are inherently risky, because the head is not necessarily a semantic head, and the *modifier* is not necessarily a semantic modifier, and in fact the opposite may be the case. In the experiments that we describe here, we have generally refrained from semantic interpretation of head-modifier relationship, treating it primarily as an *ordered* relation between otherwise equal elements. Nonetheless, even this simplified relationship has already allowed us to cut through a variety of surface forms, and achieve what we thought was a non-trivial level of normalization. The apparent lack of success of linguistically-motivated indexing in information retrieval may suggest that we haven't still gone far enough.

In our system, the head+modifier pairs stream is derived through a sequence of processing steps that include:

1. Part-of-speech tagging
2. Lexicon-based word normalization (extended "stemming")
3. Syntactic analysis with TTP parser
4. Extraction of head+modifier pairs
5. Corpus-based disambiguation of long noun phrases

These steps are described briefly below. For details the reader is referred to past TREC articles, and other works, including (Strzalkowski, 1995) and (Strzalkowski et al., 1997).

3.1.1. *Part-of-speech tagging*

Part of speech tagging allows for resolution of lexical ambiguities in a running text, assuming a known general type of text (e.g., newspaper, technical documentation, medical diagnosis, etc.) and a context in which a word is used. This in turn leads to a more accurate lexical normalization or stemming. It also is a basis for a phrase boundary detection.

We used a version of Brill's rule based tagger (Brill, 1992) trained on Wall Street Journal texts to preprocess linguistic streams used by SMART. We also used BBN's stochastic POST tagger as part of our NYU-based Prise system. Both systems are based on the Penn Treebank Tagset developed at the University of Pennsylvania, and have compatible levels of performance.

3.1.2. *Lexicon-based word normalization*

Word stemming has been an effective way of improving document recall since it reduces words to their common morphological root, thus allowing more successful matches. On the other hand, stemming tends to decrease retrieval precision, if care is not taken to prevent situations where otherwise unrelated words are reduced to the same stem. In our system we replaced a traditional morphological stemmer with a conservative dictionary-assisted suffix trimmer.[1]

The suffix trimmer performs essentially two tasks:

1. it reduces inflected word forms to their root forms as specified in the dictionary, and
2. it converts nominalized verb forms (e.g., "implementation", "storage") to the root forms of corresponding verbs (i.e., "implement", "store").

This is accomplished by removing a standard suffix, e.g., "stor+age", replacing it with a standard root ending ("+e"), and checking the newly created word against the dictionary, i.e., we check whether the new root ("store") is indeed a legal word.

3.1.3. *Syntactic analysis with TTP*

Parsing reveals finer syntactic relationships between words and phrases in a sentence, relationships that are hard to determine accurately without a comprehensive grammar. Some of these relationships do convey semantic dependencies, e.g., in *Poland is attacked by Germany* the subject+verb and verb+object relationships uniquely capture the semantic relationship of who attacked whom. The surface word-order alone cannot be relied on to determine which relationship holds. From the onset, we assumed that capturing semantic dependencies may be critical for accurate text indexing. One way to approach this is to exploit the syntactic structures produced by a fairly comprehensive parser.

TTP (Tagged Text Parser) is based on the Linguistic String Grammar developed by Sager (Sager, 1981) . The parser currently encompasses some 400 grammar productions, but it is by no means complete. The parser's output is a regularized parse tree representation of each sentence, that is, a representation that reflects the sentence's logical predicate-argument structure. For example, logical subject and logical object are identified in both passive and active sentences, and noun phrases are organized around their head elements. The parser is equipped with a powerful skip-and-fit recovery mechanism that allows it to operate effectively in the face of ill-formed input or under a severe time pressure. TTP has been shown to produce parse structures which are no worse than those generated by full-scale linguistic parsers when compared to hand-coded Treebank parse trees (Strzalkowski and Scheyen, 1996).

3.1.4. *Extracting head+modifier pairs*

Syntactic phrases extracted from TTP parse trees are head+modifier pairs. The head in such a pair is a central element of a phrase (main verb, main

[1] Dealing with prefixes is a more complicated matter, since they may have quite strong effect upon the meaning of the resulting term, e.g., "un- " usually introduces explicit negations.

noun, etc.), while the modifier is one of the adjunct arguments of the head. It should be noted that the parser's output is a predicate-argument structure centered around main elements of various phrases. The following types of pairs are considered: (1) a head noun and its left adjective or noun adjunct, (2) a head noun and the head of its right adjunct, (3) the main verb of a clause and the head of its object phrase, and (4) the head of the subject phrase and the main verb. These types of pairs account for most of the syntactic variants for relating two words (or simple phrases) into pairs carrying compatible semantic content. This also gives the pair-based representation sufficient flexibility to effectively capture content elements even in complex expressions. There are of course exceptions. For example, the threeword phrase "former Soviet president" would be broken into two pairs "former president" and "Soviet president", both of which denote things that are potentially quite different from what the original phrase refers to, and this fact may have potentially a negative effect on retrieval precision. This is one place where a longer phrase appears more appropriate. Below is a small sample of head+modifier pairs extracted (proper names are not included):

original text:
While serving in South Vietnam, a number of U.S. Soldiers were reported as having been exposed to the defoliant Agent Orange. The issue is veterans entitlement, or the awarding of monetary compensation and/or medical assistance for physical damages caused by Agent Orange.

head+modifier pairs:
damage+physical, cause+damage, award+assist, award+compensate, compensate+monetary, assist+medical, entitle+veteran

3.1.5. *Corpus-based disambiguation of long noun phrases*

The phrase decomposition procedure is performed after the first phrase extraction pass in which all unambiguous pairs (noun+noun and noun+adjective) and all ambiguous noun phrases are extracted. Any nominal string consisting of three or more words of which at least two are nouns is deemed structurally ambiguous. In the TREC corpus, about 80% of all ambiguous nominals were of length 3 (usually 2 nouns and an adjective), 19% were of length 4, and only 1% were of length 5 or more. The phrase decomposition algorithm has been described in detail in (Strzalkowski, 1995). The algorithm was shown to provide about 70% recall and 90% precision in extracting correct head+modifier pairs from 3 or more word noun groups in TREC collection texts. In terms of the total number of pairs extracted unambiguously from the parsed text, the disambiguation step recovers an additional 10% to 15% of pairs, all of which would otherwise be either discarded or misrepresented.

3.2. SIMPLE NOUN PHRASE STREAM

In contrast to the elaborate process of generating the head+modifier pairs, unnormalized noun groups are collected from part-of-speech tagged text using a few regular expression patterns. No attempt is made to disambiguate, normalize, or get at the internal structure of these phrases, other than the stemming which has been applied to text prior to the phrase extraction step. The following phrase patterns have been used, with phrase length arbitrarily limited to the maximum 7 words:

1. a sequence of modifiers (adjectives, participles, etc.) followed by at least one noun, such as: "cryonic suspension", "air traffic control system";
2. proper noun sequences modifying a noun, such as: "u.s. citizen", "china trade";
3. proper noun sequences (possibly containing '&'): "warren commission", "national air traffic controller".

The motivation for having a phrase stream is similar to that for head+-modifier pairs since both streams attempt to capture significant multi-word indexing terms. The main difference is the lack of normalization, which makes the comparison between these two streams particularly interesting.

3.3. NAME STREAM

In our system names are identified by the parser, and then represented as strings, e.g., south+africa. The name recognition procedure is extremely simple, in fact little more than the scanning of successive words labeled as proper names by the tagger ("np" and "nps" tags). Single-word names are processed just like ordinary words, except for the stemming which is not applied to them. We also made no effort to assign names to categories, e.g., people, companies, places, etc., a classification which is useful for certain types of queries (e.g., "To be relevant a document must identify a specific generic drug company"). In the TREC-5 database, compound names make up about 8% of all terms generated. A small sample of compound names extracted is listed below:

right+wing+christian+fundamentalism, gun+control+legislation,
u.s+government, exxon+valdez, plo+leader+arafat,
national+railroad+transportation+corporation,
suzuki+samurai+soft-top+4wd

TABLE 1. How different streams perform relative to one another (11-pt avg. Prec)

RUNS	*short queries*	*long queries*
Stems	0.1070	0.2684
Phrases	0.0846	0.2541
H+M Pairs	0.0405	0.1787
Names	0.0648	0.0753

3.4. STEMS STREAM

The stems stream is the simplest, yet the most effective of all streams, a backbone of the multistream model. It consists of stemmed single-word tokens (plus hyphenated phrases) taken directly from the document text (exclusive of stopwords). The stems stream provides the most comprehensive, though not very accurate, image of the text it represents, and therefore it is able to outperform other streams that we used thus far. We believe however, that this representation model has reached its limits, and that further improvement can only be achieved in combination with other text representation methods. This appears consistent with the results reported at TREC.

In addition, we use WordNet (Miller, 1980) to identify unambiguous single-sense words and give them premium weights as reliable discriminators. Many words, when considered out of context, display more than one sense in which they can be used. When such words are used in text they may assume any of their possible senses thus leading to undesired matches. This has been a problem for word based IR systems, and have spurred attempts at sense disambiguation in text indexing (Krovetz and Croft, 1992). Another way to address this problem is to focus on words that do not have multiple-sense ambiguities, and treat these as special, because they seem to be more reliable as content indicators. This modification has produced a slightly stronger stream.

The results in Table 1 are somewhat counter-intuitive, particularly the unexpectedly weak performance of H+M Pairs stream. While we have noticed that Phrases often outperform Pairs (cf. TREC-5 results), the difference was never this pronounced. One possible explanation is a worse than expected quality of parse structures generated by TTP, which may be related to sub-optimal setting of critical parameters, particularly the time-out value. We continue to investigate these results.

For streams using SMART indexing, we selected optimal term weighting

TABLE 2. Term weighting across streams using SMART

STREAM	*weighting scheme*
Stems	lnc.ntn
Phrases	ltn.ntn
H+M Pairs	ltn.nsn
Names	ltn.ntn

schemes from among a dozen or so variants implemented with version 11 of the system. These schemes vary in the way they calculate and normalize basic term weights. For example, in *lnc.ntn* scheme, *lnc* scoring (*log-tf, no-idf, cosine-normalization*) is applied to documents, and *ntn* scoring (*straight-tf, idf, nonormalization*) is applied to query terms. The selection of one scheme over another can have a dramatic effect on system's performance. For details the reader is referred to (Buckley, 1993).

4. Stream Merging and Weighting

The results obtained from different streams are lists of documents ranked in order of relevance: the higher the rank of a retrieved document, the more relevant it is presumed to be. In order to obtain the final retrieval result, ranking lists obtained from each stream have to be combined together by a process known as merging or fusion. The final ranking is derived by calculating the combined relevance scores for all retrieved documents. The following are the primary factors affecting this process:

1. document relevancy scores from each stream
2. retrieval precision distribution estimates within ranks from various streams, e.g., projected precision between ranks 10 and 20, etc.;
3. the overall effectiveness of each stream (e.g. measured as average precision on training data)
4. the number of streams that retrieve a particular document, and
5. the ranks of this document within each stream.

Generally, a stronger (i.e., better performing) stream will more effect on shaping the final ranking. A document which is retrieved at a high rank from such a stream is more likely to end up ranked high in the final result. In addition, the performance of each stream within a specific range of ranks is taken into account. For example, if phrases stream tends to pack relevant documents between the top 10th and 20th retrieved documents (but not so much into 1-10) we would give premium weights to the documents found

TABLE 3. Precision improvements over stems-only retrieval based on TREC-5 data

Streams merged	*short queries* *% change*	*long queries* *% change*
All streams	+5.4	+20.94
Stems+Phrases+Pairs	+6.6	+22.85
Stems+Phrases	+7.0	+24.94
Stems+Pairs	+2.2	+15.27
Stems+Names	+0.6	+2.59

in this region of phrase-based ranking, etc. Table 3 gives some additional data on the effectiveness of stream merging. Further details are available in our TREC-5 conference article (Strzalkowski et al., 1997).

Note that long text queries benefit more from linguistic processing.

4.1. INTER-STREAM MERGING USING PRECISION DISTRIBUTION ESTIMATES

We used the following two principal sources of information about each stream to weigh their relative contributions to the final ranking:

- an actual ranking obtained from a training run (training data, old queries);
- an estimated retrieval precision at certain ranges of ranks.

Precision estimates are used to order results obtained from the streams, and this ordering may vary at different rank ranges. Table 4 shows precision estimates for selected streams at certain rank ranges as obtained from a training collection derived from TREC-4 data.

The final score of a document (d) is calculated using the following formula:

$$\begin{aligned} finalscore(d) &= \sum_{i=1 \ldots N} A(i) \times score(i)(d) \times \\ &\quad prec(\{ranks(i) | rank(i,d) \in ranks(i)\} \end{aligned}$$

where N is the number of streams; $A(i)$ is the stream coefficient; and $score(i)(d)$ is the normalized score of the document against the query within the stream i; $prec(ranks(i))$ is the precision estimate from the precision distribution table for stream i; and $rank(i,d)$ is the rank of document d in stream i.

TABLE 4. Precision distribution estimates for selected streams

RANKS	STEMS	PHRASES	PAIRS	NAMES
1-5	0.49	0.45	0.33	0.23
6-10	0.42	0.38	0.27	0.18
11-20	0.37	0.32	0.23	0.13
21-30	0.33	0.28	0.21	0.10
31-50	0.27	0.25	0.17	0.08
51-100	0.19	0.17	0.12	0.06
101-200	0.12	0.11	0.08	0.04

TABLE 5. Stream merging coefficient structures used in TREC-5

	STREAMS			
RUNS	stems	phrases	pairs	names
ad-hoc gerua1	4	3	3	1
ad-hoc gerua3	5	3	3	1
routing gerou1	4	3	3	1
routing gesri2	4	3	3	1

4.2. STREAM COEFFICIENTS

For merging purposes, streams are assigned numerical coefficients, referred to as A(i) above, that have two roles:

1. Control the relative contribution of a document score assigned to it within a stream when calculating the final score for this document. This applies primarily to streams producing normalized document scores, such as SMART.
2. Change stream-to-stream document score relationships for un-normalized ranking system, e.g., PRISE.

An example of a coefficient structure is shown below. They are obtained empirically to maximize the performance of any specific combination of streams. Table 5 summarizes stream coefficient structures used in TREC-5 experiments. Typically, a new combination was created for a given collection, a retrieval mode (ad-hoc vs. routing) and the search engines used.

5. Query Expansion Experiments

5.1. WHY QUERY EXPANSION?

The purpose of query expansion is to make the user query resemble more closely the documents it is expected to retrieve. This includes both content, as well as some other aspects such as composition, style, language type, etc. If the query is indeed made to resemble a "typical" relevant document, then suddenly everything about this query becomes a valid search criterion: words, collocations, phrases, various relationships, etc. Unfortunately, an average search query does not look anything like this, most of the time. It is more likely to be a statement specifying the semantic criteria of relevance. This means that except for the semantic or conceptual resemblance (which we cannot model very well as yet) much of the appearance of the query (which we can model reasonably well) may be, and often is, quite misleading for search purposes. Where can we get the right queries?

In today's information retrieval, query expansion usually pertains content and typically is limited to adding, deleting or re-weighting of terms. For example, content terms from documents judged relevant are added to the query while weights of all terms are adjusted in order to reflect the relevance information. Thus, terms occurring predominantly in relevant documents will have their weights increased, while those occurring mostly in non-relevant documents will have their weights decreased. This process can be performed automatically using a relevance feedback method, e.g., (Rocchio, 1971), with the relevance information either supplied manually by the user (Harman, 1988), or otherwise guessed, e.g. by assuming top 10 documents relevant, etc. (Buckley, et al., 1995). A serious problem with this content-term expansion is its limited ability to capture and represent many important aspects of what makes some documents relevant to the query, including particular term co-occurrence patterns, and other hard-to-measure text features, such as discourse structure or stylistics. Additionally, relevance-feedback expansion depends on the inherently partial relevance information, which is normally unavailable, or unreliable. Other types of query expansions, including general purpose thesauri or lexical databases (e.g., Wordnet) have been found generally unsuccessful in information retrieval (cf. (Voorhees, 1993); (Voorhees, 1994)).

An alternative to term-only expansion is a full-text expansion which we tried for the first time in TREC-5. In our approach, queries are expanded by pasting in entire sentences, paragraphs, and other sequences directly from any text document. To make this process efficient, we first perform a search with the original, un-expanded queries (short queries), and then use top N (10, 20) returned documents for query expansion. These documents are not judged for relevancy, nor assumed relevant; instead, they are scanned for passages that contain concepts referred to in the query. Expansion material can be found in both relevant and non-relevant documents, benefitting the final query all the same. In fact, the presence of such text in otherwise non-relevant documents underscores the inherent limitations of distribution-based term reweighting used in relevance feedback. Subject to some further "fitness criteria", these expansion passages are then imported verbatim into the query. The resulting expanded queries undergo the usual text processing steps, before the search is run again.

Full-text expansion can be accomplished manually, as we did initially to test feasibility of this approach in TREC-5, or semi-automatically, as we tried this year with excellent results. Our goal is to fully automate this process. (We did try an automatic expansion in TREC-5, but it was very simplistic and not very sucessful, cf. our TREC-5 report.)

The initial evaluations indicate that queries expanded manually following the prescribed guidelines are improving the system's performance (precision and recall) by as much as 40% or more. This appear to be true not only for our own system, but also for other systems: we asked other groups participating in TREC-5 to run search using our expanded queries, and they reported nearly identical improvements. Below, we describe the three different query expansion techniques explored in TREC-6.

5.2. SUMMARIZATION-BASED QUERY EXPANSION

We used an automatic text summarizer to derive query-specific summaries of documents returned from the first round of retrieval. The summaries were usually 1 or 2 consecutive paragraphs selected from the original document text. The purpose was to demonstrate, in a quick-read abstract, why a given document has been retrieved. If the summary appeared relevant and moreover captured some new aspect of relevant information, then it was pasted into the query. Note that it wasn't important if the document itself was relevant.

The summaries were produced automatically using GE Summarizer-Tool, a prototype developed for Tipster Phase 3 project. It works by extracting passages from the document text, and producing perfectly readable, very brief summaries, at about 5 to 10% of original text length.

A preliminary examination of TREC-6 results indicate that this mode of expansion is at least as effective as the purely manual expansion used in TREC-5. This is a very good news, since we now appear to be a step closer to an automatic expansion. The human-decision factor has been reduced to an accept/reject decision for expanding the search query with a summary – no need to read the whole document in order to select expansion passages.

5.3. EXTRACTION-BASED QUERY EXPANSION

We used automatic information extraction techniques to score text passages for presence of concepts (rather than keywords) identified by the query. Small extraction grammars were manually constructed for 23 out of 47 routing queries. Using SRI's FASTUS information extraction system, we selected highest score sentences from known relevant documents in the training corpus. Please note that this was a routing run, and the setup was somewhat different than in other query expansion runs. In particular, there was only one run against the test collection (the routing mode allows no feedback).

This run was constructed in collaboration with SRI's team. SRI has developed FASTUS grammars, run FASTUS over the training documents, scored each sentence, and sent the sentences to GE. GE team applied stream model processing to the queries, run the queries against the test collection, and submitted the results to NIST.

5.4. INTERACTIVE QUERY EXPANSION WITH INQUERY

The results produced at Rutgers were obtained using an interactive system. We believe that through interaction with the system and the database the user can create significantly better queries. The support to the user provided by the interface in order to build better queries is at least as important as any other part of the system.

In our previous contributions we have devoted very significant resources in terms of processing power and time to the creation of better document representations. In particular, we have applied NLP techniques to thousands of megabytes of text in order to add less ambiguous terms to the document representation. In the interaction experiment we attempted to move processing power and "intelligence" from the representation to the interface. What we are trying to do is to spend a few tenths of a second executing even more sophisticated techniques (including, in future interfaces, NLP) on the query instead of days processing several gugabytes of the corpus in order to generate a better representation.

A new user interface for InQuery, called RUINQ2, was developed at Rutgers for this experiment. This is a variation of RUINQ, the InQuery interface developed for use in the interactive track experiments reported by the Rutgers team (see Rutgers paper in these proceedings).

RUINQ2 supports the use of negative and positive feedback. The user is shown a list of 10 document titles at a time. The user can scroll to see another 10 as many times as needed. Any number of the titles presented can be declared either relevant or non relevant by the user (by clicking next to the title). When a document is declared relevant (non relevant) some terms are offered to the user on a positive (negative) feedback window. The user can add to the query any number of terms from those windows by clicking on the desired term.

RUINQ2 also supported the use of phrases (any sequence of words entered by the user inside double quotes) and required terms (preceeded by a plus sign).

The interactive run was created in order to have a baseline to compare query expansion using automatically generated summaries, with query expansion using interaction with document text plus negative/positive feedback. Further experiments based on more refined user interfaces for both systems should help us answer questions such as: which system is easier to use, which one allows users to create queries faster and which system helps user create more effective queries.

The interactive run was created by allowing a single user (one of the authors) who had never seen the topics before, to interact with the system for no more than 15mins per topic in order to build the corresponding query. When the user was satisfied with the query he would click on a button that would print out the rankings of (at most) 1000 documents in TREC format. In several cases less than 1000 documents were found.

We discovered a bug in the program after we had submmited the results. The ranking printed out began by the first document displayed on the document title screen at the moment the user decided to print out the rankings (as opposed to the first document of the ranking). So, if the user was looking at the second page of document titles at the time he printed out the ranking, the first 10 documents of the ranking were not printed. This happened with about 8 queries. The corrected results will be presented in our talk at the conference.

6. SUMMARY OF RESULTS

6.1. AD-HOC RUNS

Ad-hoc retrieval is when an arbitrary query is issued to search a database for relevant documents. In a typical ad-hoc search situation, a query is used once, then discarded, thus leaving little room for optimization. Our ad-hoc experiments were conducted in several subcategories, including automatic, manual, and using different sizes of databases and different types of queries. An automatic run means that there was no human intervention in the process at any time. A manual run means that some human processing was done to the queries, and possibly multiple test runs were made to improve the queries. A short query is derived using only one section of a TREC-5 topic, namely the DESCRIPTION field. A full query is derived from any or all fields in the topic. An example TREC-5 query is show below; note that the Description field is what one may reasonably expect to be an initial search query, while Narrative provides some further explanation of what relevant material may look like. The Topic field provides a single concept of interest to the searcher; it was not permitted in the short queries.

< *top* >
< *num* > Number: 324
< *title* > Argentine/British Relations
< *desc* > Description:
Define Argentine and British international relations
< *narr* > Narrative:
It has been 15 years since the war between Argentina and the United Kingdom in 1982 over sovereignty in the Falkland Islands. A relevant report will describe their relations after that period. Any kind of international contact between the two countries is relevant, to include commercial, economic, cultural, diplomatic, or military exchanges. Negative reports on the absence of such exchanges are also desirable. Reports containing information on direct exchanges between Argentina and the Falkland Islands are also relevant.
< */top* >

Table 6 summarizes selected runs performed with our NLIR system on TREC-6 database using 50 queries numbered 301 through 350. The SMART baselines were produced by Cornell-SaBir team using version 11 of the system. The rightmost column is an unofficial rerun of the GERUA1 after fixing of a simple bug. Table 7 compares the performance of UMass' InQuery system on the same set of queries, and the same database. Note the consistently large improvements in retrieval precision attributed to the expanded queries.

6.2. ROUTING RUNS

Routing is a process in which a stream of previously unseen documents are filtered and distributed among a number of standing profiles, also known as routing queries. In routing, documents can be assigned to multiple profiles. In categorization, a type of routing, a single best matching profile is selected for each document. Routing is harder to evaluate in a standardized setup than the retroactive retrieval because of its dynamic nature, therefore a simulated routing mode has been used in TREC. A simulated routing mode (TREC-style) means that all routing documents are available at once, but the routing queries (i.e., terms and their weights) are derived with respect to a different training database, specifically TREC collections from previous evaluations. This way, no statistical or other collection-specific information about the routing documents is used in building the profiles, and the participating systems are forced to make assumptions about the routing documents just like they would in real routing. However, no real routing occurs, and the prepared routing queries are run against the rout-

TABLE 6. Precision improvement in NLIR system vs. SMART (v.11) baselines

queries:	full	full	man long	man long	man long-1
PREC.	SMART	Best NL	SMART	Best NL	Best NL
11pt. avg	0.1429	0.1837	0.2672	0.2783	0.2859
%change		+28.5	+87.0	+94.7	+100.0
@10 docs	0.3000	0.3840	0.5060	0.5200	0.5200
%change		+28.0	+68.6	+73.3	+73.3
@30 docs	0.2387	0.2747	0.3887	0.3933	0.3940
%change		+15.0	+62.8	+64.7	+65.0
@100 doc	0.1600	0.1736	0.2480	0.2598	0.2574
%change		+8.5	+55.0	+62.3	+60.8
Recall	0.57	0.53	0.61	0.58	0.62
%change		-7.0	+7.0	+1.7	+8.7

TABLE 7. Results for UMass' InQuery (no NL indexing)

PREC.	automatic full queries (T+D)	manual long queries
11pt.avg	0.2103	0.3057
%change		+45.0
@20 docs	0.3620	0.4510
%change		+25.0
R-Prec	0.2461	0.3327
%change		+35.0

ing database much the same way they would be in an ad-hoc retrieval. Documents retrieved by each routing query, ranked in order of relevance, become the content of its routing bin.

6.2.1. *Query development against the training collection*

In Smart routing, automatic relevance feedback was performed to build routing queries using the training data available from previous TRECs. The routing queries, split into streams, were then run against stream-indexed routing collection. The weighting scheme was selected in such a way that no collection-specific information about the current routing data has been used. Instead, collection-wide statistics, such as *idf* weights, were those

TABLE 8. Precision averages for 47 routing queries

STREAMS	11pt. Prec	At 5 docs	At 10 docs	R-Prec
main routing, gerou1	0.2702	0.5532	0.4787	0.3176
query expansion gesri2	0.2458	0.5447	0.4894	0.2906
reranked gerou1, srige1	0.2730	0.5574	0.5021	0.3126

derived from the training data. The routing was carried out in the following four steps:

1. A subset of the previous TREC collections was chosen as the training set, and four index streams were built. Queries were also processed and run against the indexes. For each query, 1000 documents are retrieved. The weighting schemes used were: lnc.ltc for stems, ltc.ntc for phrases, ltc.ntc for head+modifier pairs, and ltc.ntc for names.
2. The final query vector was then updated through an automatic feedback step using the known relevance judgements. Up to 350 terms occurring in the most relevant documents were added to each query. Two alternative expanded vectors were generated for each query using different sets of Roccio parameters.
3. For each query, the best performing expansion was retained. These were submitted to NIST as official routing queries.
4. The final queries were run against the four-stream routing test collection and retrieved results were merged.

6.2.2. *Query expansion via sentence extraction*

This run was described in the preceding section on query expansion.

6.2.3. *Re-ranking using rescoring via extraction*

This run was created at SRI using the output from GE's main routing run. SRI's FASTUS (Hobbs et al., 1996) was used to score documents retrieved and rerank them if they contained concepts asked for in the query. For details of this run please refer to SRI's chapter (Bear & Israel, this volume).

The results of using information extraction techniques are shown in Table 8 and compared to our main routing run. We note a slight improvement in average precision, and a more definite precision improvement near the top of the ranking in FASTUS rescoring run. This is only a first attempt at a serious-scale experiment of this kind, and the results are definitely encouraging.

7. CONCLUSIONS

We presented in some detail our natural language information retrieval system consisting of an advanced NLP module and a 'pure' statistical core engine. While many problems remain to be resolved, including the question of adequacy of term-based representation of document content, we attempted to demonstrate that the architecture described here is nonetheless viable. In particular, we demonstrated that natural language processing can now be done on a fairly large scale and that its speed and robustness has improved to the point where it can be applied to real IR problems.

The main observation to make is that thus far natural language processing has not proven as effective as we would have hoped in to obtain better indexing and better term representations of queries. Using linguistic terms, such as phrases, head-modifier pairs, names, does help to improve retrieval precision, but the gains remain quite modest. On the other hand, full text query expansion works remarkably well, and even more so in combination with linguistic indexing. Our main effort in the immediate future will be to explore ways to achieve at least partial automation of this process. Using information extraction techniques to improve retrival either by building better queries, or by reorganizing the results is another promising line of investigation.

Acknowledgements We would like to thank Donna Harman for making the NIST's PRISE system available to this project since the beginning of TREC. We also thank Chris Buckley for helping us to understand the inner workings of SMART. We would like to thank Ralph Weischedel for providing and assisting in the use of the BBN's part of speech tagger. Finally, thanks to SRI's Jerry Hobbs, David Israel and John Bear for their collaboration on joint experiments. This paper is based upon work supported in part by the Defense Advanced Research Projects Agency under Tipster Phase-3 Contract 97-F157200-000.

References

Brill, Eric. 1992. "A Simple Rule-Based Part Of Speech Tagger." Proceedings of the Third Conference on Applied Computational Linguistics (ANLP).

Buckley, Chris, Amit Singhal, Mandar Mitra, Gerard Salton. 1995. "New Retrieval Approches Using SMART: TREC 4". Proceedings of the Fourth Text REtrieval Conference (TREC-4), NIST Special Publication 500-236.

Buckley, Chris. 1993. "The Importance of Proper Weighting Methods." Human Language Technology, Proceedings of the workshop, Princeton, NJ. Morgan-Kaufmann, pp. 349-352.

Callan, Jamie, Zhihong Lu, and W. Bruce Croft. 1995. "Searching Distributed Collections with Inference Networks." Proceedings of ACM SIGIR'95. pp. 21-29.

Fox, Ed, M. Koushik, J. Shaw, R. Modlin, and D. Rao. 1993. "Comining Evidence from Multiple Searches." Proceedings of First Text Retrieval Conference (TREC-1), NIST Special Publication 500-207, National Institute of Standards and Technology, Gaithersburg, MD. pp. 319-328.

Harman, Donna. 1988. "Towards interactive query expansion." Proceedings of ACM SIGIR-88, pp. 321-331.

Hobbs, Jerry, Douglas Appelt, John Bear, David Israel, Megumi Kameyama, Andrew Kehler, Mark Stickel, and Mabry Tyson. 1996. "SRI's Tipster II Project." Advances in Text Processing, Tipster Progran Phase 2. Morgan Kaufmann, pp. 201-208.

Krovetz, Robert and W. Bruce Croft. 1992. "Lexical ambiguity and information retrieval." *ACM Transactions on Information Systems*, 10(2), pp. 115-141.

Rocchio, J. J. 1971. "Relevance Feedback in Informatio Retrieval." In Salton, G. (Ed.), *The SMART Retrieval System*, pp. 313-323. Prentice Hall, Inc., Englewood Cliffs, NJ.

Sager, Naomi. 1981. *Natural Language Information Processing.* Addison-Wesley.

Saracevic, T., Kantor, P. 1988. "A Study of Information Seeking and Retrieving. III. Searchers, Searches, and Overlap." *Journal of the American Society for information Science*, 39(3):197-216.

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, and Peter Scheyen. 1993. "An Evaluation of TTP Parser: a preliminary report." Proceedings of International Workshop on Parsing Technologies (IWPT-93), Tilburg, Netherlands and Durbuy, Belgium, August 10-13.

Strzalkowski, Tomek and Jose Perez Carballo. 1994. "Recent Developments in Natural Language Text Retrieval." Proceedings of the First Text REtrieval Conference (TREC-2), NIST Special Publication 500-215, National Institute of Standards and Technology, Gaithersburg, MD. pp. 123-136.

Strzalkowski, Tomek. 1995. "Natural Language Information Retrieval" *Information Processing and Management*, Vol. 31, No. 3, pp. 397-417. Pergamon/Elsevier.

Strzalkowski, Tomek, and Peter Scheyen. 1996. "An Evaluation of TTP Parser: a preliminary report." In H. Bunt, M. Tomita (eds), *Recent Advances in Parsing Technology*, Kluwer Academic Publishers, pp. 201-220.

Voorhees, Ellen M. 1994. "Query Expansion Using Lexical-Semantic Relations." Proceedings of ACM SIGIR'94, pp. 61-70.

Voorhees, Ellen M. 1993. "Using WordNet to Disambiguate Word Senses for Text Retrieval." Proceedings of ACM SIGIR'93, pp. 171-180.