NATURAL LANGUAGE INFORMATION RETRIEVAL: TREC-5 REPORT

Tomek Strzalkowski[1] Louise Guthrie[2] Jussi Karlgren[3] Jim Leistensnider[2]
Fang Lin[1] Jose Perez-Carballo[4] Troy Straszheim[3] Jin Wang[1] Jon Wilding[2]

[1]GE Corporate Research & Development
[2]Lockheed Martin Corporation
[3]Department of Computer Science, New York University
[4]School of Communication, Information and Library Studies, Rutgers University

ABSTRACT

In this paper we report on the joint GE/Lockheed Martin/Rutgers/NYU natural language information retrieval project as related to the 5th Text Retrieval Conference (TREC-5). The main thrust of this project is to use natural language processing techniques to enhance the effectiveness of full-text document retrieval. Since our first TREC entry in 1992 (as NYU team) the basic premise of our research was to demonstrate that robust if relatively shallow NLP can help to derive a better representation of text documents for statistical search. TREC-5 marks a shift in this approach away from text representation issues and towards query development problems. While our TREC-5 system still performs extensive text processing in order to extract phrasal and other indexing terms, our main focus this year was on query construction using words, sentences, and entire passages to expand initial topic specifications in an attempt to cover their various angles, aspects and contexts. Based on our earlier TREC results indicating that NLP is more effective when long, descriptive queries are used, we allowed for liberal expansion with long passages from related documents imported verbatim into the queries. This method appears to have produced a dramatic improvement in the performance of two different statistical search engines that we tested (Cornell's SMART and NIST's Prise) boosting the average precision by at least 40%.

The overall architecture of TREC-5 system has also changed in a number of ways from TREC-4. The most notable new feature is the stream architecture in which several independent, parallel indexes are built for a given collection, each index reflecting a different representation strategy for text documents. Stream indexes are built using a mixture of different indexing approaches, term extracting, and weighting strategies. We used both SMART and Prise base indexing engines, and selected optimal term weighting strategies for each stream, based on a training collection of approximately 500 MBytes. The final results are produced by a merging procedure that combines ranked list of documents obtained by searching all stream indexes with appropriately preprocessed queries. This allows for an effective combination of alternative retrieval and filtering methods, creating into a meta-search where the contribution of each stream can be optimized through training.

1.0 INTRODUCTION AND BACKGROUND

A typical (full-text) information retrieval (IR) task is to select documents from a database in response to a user's query, and rank these documents according to relevance. This has been usually accomplished using statistical methods (often coupled with manual encoding) that (a) select terms (words, phrases, and other units) from documents that are deemed to best represent their content, and (b) create an inverted index file (or files) that provide an easy access to documents containing these terms. A subsequent search process will attempt to match preprocessed user queries against term-based representations of documents in each case determining a degree of relevance between the two which depends upon the number and types of matching terms. Although many sophisticated search and matching methods are available, the crucial problem remains to be that of an adequate representation of content for both the documents and the queries.

In term-based representation, a document (as well as a query) is transformed into a collection of weighted terms, derived directly from the document text or indirectly through thesauri or domain maps. The representation is anchored on these terms, and thus their careful selection is critical. Since each unique term can be thought to add a new dimensionality to the representation, it is equally critical to weigh them properly against one another so that the document is placed at the correct position in the N-dimensional term space. Our goal here is to have the documents on the same topic placed close together, while those on different topics placed sufficiently apart. Unfortunately, we often do not know how to compute terms weights. The statistical weighting formulas, based on terms distribution within the database, such as tf*idf, are far from optimal, and the assumptions of term independence which are routinely made are false in most cases. This situation is even worse when single-word terms are intermixed with phrasal terms and the term independence becomes harder to justify.

The simplest word-based representations of content, while relatively better understood, are usually inadequate since single words are rarely specific enough for accurate discrimination, and their grouping is often accidental. A better method is to identify groups of words that create meaningful phrases, especially if these phrases denote important concepts in the database domain. For example, “joint venture” is an important term in the Wall Street Journal (WSJ henceforth) database, while neither “joint” nor “venture” are important by themselves. In the retrieval experiments with the training TREC database, we noticed that both “joint” and “venture” were dropped from the list of terms by the system because their idf (inverted document frequency) weights were too low. In large databases, such as TIPSTER, the use of phrasal terms is not just desirable, it becomes necessary.

There are a number of ways to obtain “phrases” from text. These include generating simple collocations, statistically validated N-grams, part-of-speech tagged sequences, syntactic structures, and even semantic concepts. Some of these techniques are aimed primarily at identifying multi-word terms that have come to function like ordinary words, for example “white collar” or “electric car”, and capturing other co-occurrence idiosyncrasies associated with certain types of texts. This simple approach has proven quite effective for some systems, for example the Cornell group reported (Buckley, 1995) that adding simple collocations to the list of available terms can increase retrieval precision by as much as 10%.

Other more advanced techniques of phrase extraction, including extended N-grams and syntactic parsing, attempt to uncover “concepts”, which would capture underlying semantic uniformity across various surface forms of expression. Syntactic phrases, for example, appear reasonable indicators of content, arguably better than proximity-based phrases, since they can adequately deal with word order changes and other structural variations (e.g., “college junior” vs. “junior in college” vs. “junior college”). A subsequent regularization process, where alternative structures are reduced to a “normal form”, helps to achieve a desired uniformity, for example, “college+junior” will represent a college for juniors, while “junior+college” will represent a junior in a college. A more radical normalization would have also “verb object”, “noun rel-clause”, etc. converted into collections of such ordered pairs. This head+modifier normalization has been used in our system, and is further described in section 3. It has to be noted here that the use of full-scale syntactic analysis is severely pushing the limits of practicality of an information retrieval system because of the increased demand for computing power and storage. At the same time, while the gain in recall and precision has not been negligible, no dramatic breakthrough has occurred either.

This state of affairs has prompted us take a closer look at the phrase selection and representation process. In TREC-3 we showed that an improved weighting scheme for compound terms, including phrases and proper names, leads to an overall gain in retrieval accuracy. The fundamental problem, however, remained to be the system’s inability to recognize, in the documents searched, the presence or absence of the concepts or topics that the query is asking for. The main reason for this was, we noted, the limited amount of information that the queries could convey on various aspects of topics they represent. Therefore, starting with TREC-4, and continuing on a much larger scale in TREC-5, we started experimenting with manual and automatic query building techniques. The purpose was to devise a method for full-text query expansion that would allow for creating exhaustive search queries such that: (1) the performance of any system using these queries would be significantly better than when the system is run using the original topics, and (2) the method could be eventually automated or semi-automated so as to be useful to a non-expert user. Our preliminary results from TREC-5 evaluations show that this approach is indeed very effective.

In this paper we describe the overall organization of our TREC-5 system, and then discuss the official and "unofficial" experiments and their results, as well as our future research plans.

2.0 STREAM-BASED INFORMATION RETRIEVAL MODEL

Our NLIR system encompasses several statistical and natural language processing (NLP) techniques for robust text analysis. These has been organized together into a "stream model" in which alternative methods of document indexing are strung together to perform in parallel. Stream indexes are built using a mixture of different indexing approaches, term extracting and weighting strategies, even different search engines. The final results are produced by merging ranked lists of documents obtained from searching all stream indexes 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. The stream model is illustrated in Figure 1. We used both Cornell's SMART version 11, and NIST's Prise search engines.

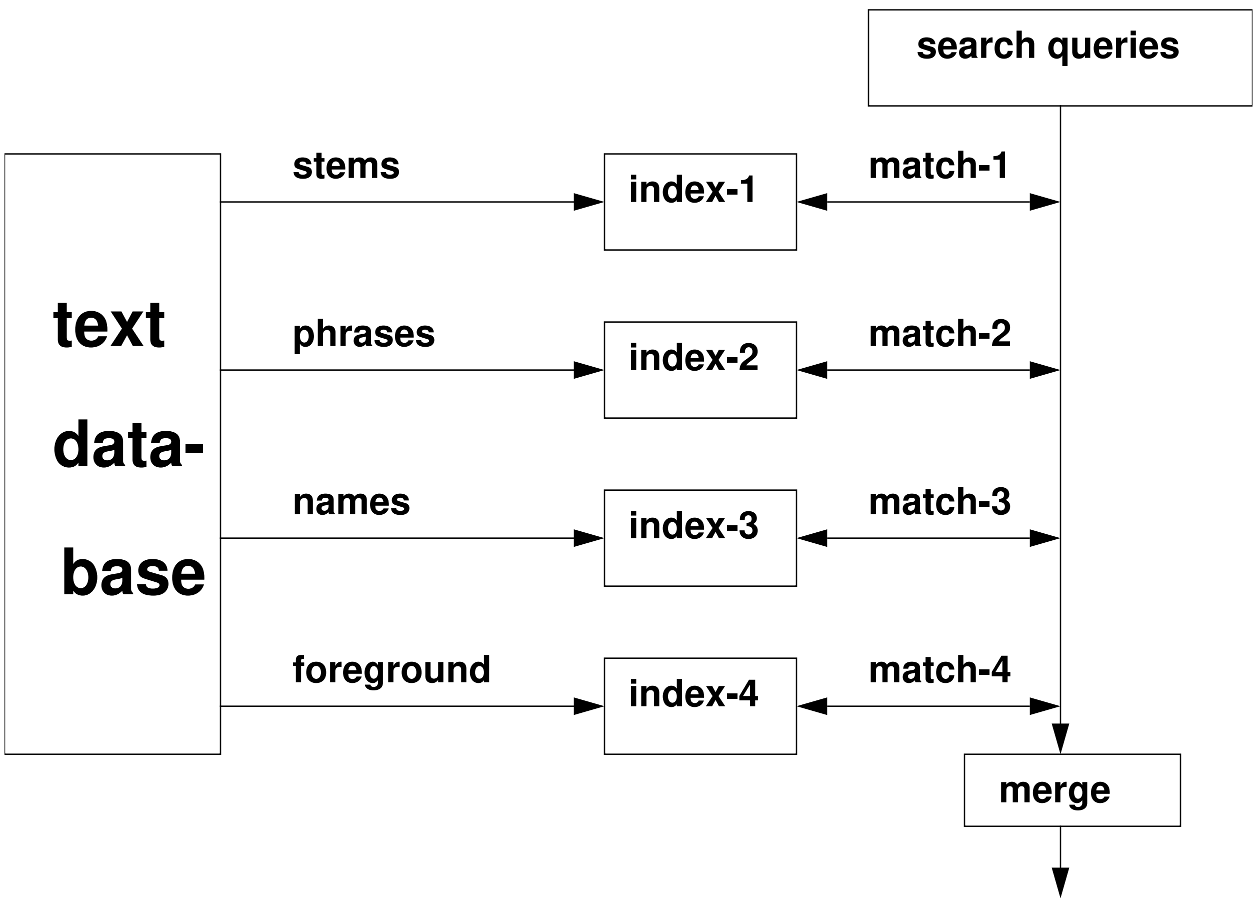

FIGURE 1. Stream organization concept.

Our TREC-5 system employs a suite of advanced natural language processing techniques in order to assist the statistical retrieval engine in selecting appropriate indexing terms for documents at hand, and to assign them semantically validated weights. The following term extraction methods has been used, which correspond to some of the streams we used:

1. Eliminate stopwords: original text words minus certain no-content words are used to index documents.

2. Morphological stemming: we normalize across morphological word variants (e.g., “proliferation”, “proliferate”, “proliferating”) using a lexicon-based stemmer.
3. Phrase extraction: we use various shallow text processing techniques, such as part-of-speech tagging, phrase boundary detection, and word co-occurrence metrics to identify stable strings of words, such as “joint venture”.
4. Phrase normalization: we identify “head+modifier” pairs in order to normalize across syntactic variants such as “weapon proliferation”, “proliferation of weapons”, “proliferate weapons”, etc. into “weapon+proliferate”.
5. Proper names: we identify proper names for indexing, including people names and titles, location names, organization names, etc.

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 Tipster-compliant IR engines without having to modify their code at all.

In addition, several researchers in past TRECs have noticed 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 system (see for example, ref Saracevic and Kantor and ref Bruce Croft).

3.0 ADVANCED LINGUISTIC STREAMS

3.1 Head-Modifier Pairs Stream

Our most linguistically 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.

The pairs stream is derived through a sequence of processing steps that include:

- Part-of-speech tagging
- Lexicon-based word normalization (extended “stemming”)
- Syntactic analysis with TTP parser
- Extraction of head+modifier pairs
- Corpus-based disambiguation of long noun phrases

3.1.1 Part-of-speech tagging

We used a version of Brill’s rule based tagger trained on Wall Street Journal texts to preprocess linguistic streams used by SMART. We also used BBN’s POST tagger as part of our NYU-based Prise system. Both systems use Penn. Treebank Tagset developed at 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. Below is a small example of text before and after stemming.

> *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.*
>
> serve south vietnam number u.s. soldier expose defoliant agent orange veteran entitle award monetary compensate medical assist physical damage agent orange

Please note that proper names, such as South Vietnam and Agent Orange are identified separately through the name extraction process described below. Note also that various “stopwords” (e.g., prepositions, conjunctions, articles, etc.) are removed from text.

3.1.3 Syntactic analysis with TTP

TTP (Tagged Text Parser) is based on the Linguistic String Grammar developed by 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. When parsing the TREC-3 collection of more than 500 million words, we found that the parser’s speed averaged between 0.17 and 0.26 seconds per sentence, or up to 80 words per second, on a Sun’s SparcStation10. In addition, 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.

TTP is a full grammar parser, and initially, it attempts to generate a complete analysis for each sentence. However, unlike an ordinary parser, it has a built-in timer which regulates the amount of time allowed for parsing any one sentence. If a parse is not returned before the allotted time elapses, the parser enters the skip-and-fit mode in which it will try to “fit” the parse. While in the skip-and-fit mode, the parser will attempt to forcibly reduce incomplete constituents, possibly skipping portions of input in order to restart processing at a next unattempted constituent. In other words, the parser will favor reduction to backtracking while in the skip-and-fit mode. The result of this strategy is an approximate parse, partially fitted using top-down predictions. The fragments skipped in the first pass are not thrown out, instead they are analyzed by a simple phrasal parser that looks for noun phrases and relative clauses and then attaches the recovered material to the main parse structure. Full details of TTP parser have been described in the

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 negation

TREC-1 report (Strzalkowski, 1993a), as well as in other works (Strzalkowski, 1992; Strzalkowski & 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 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 three-word phrase "former Soviet president" has been 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 negative effect on retrieval precision. This is one place where a longer phrase appears more appropriate.

3.1.5 Corpus-based disambiguation of long noun phrases

The notorious structural ambiguity of nominal compounds remains a serious difficulty in obtaining quality head-modifier pairs. What it means is that word order information cannot be reliably used to determine relationships between words in complex phrases, which is required to decompose longer phrases into meaningful head+modifier pairs. In order to cope with ambiguity, the pair extractor looks at the distribution statistics of the compound terms to decide whether the association between any two words (nouns and adjectives) in a noun phrase is both syntactically valid and semantically significant. For example, we may accept language+natural and processing+language from "natural language processing" as correct, however, case+trading would make a mediocre term when extracted from "insider trading case". On the other hand, it is important to extract trading+insider to be able to match documents containing phrases "insider trading sanctions act" or "insider trading activity". Phrasal terms are extracted in two phases. In the first phase, only unambiguous head-modifier pairs are generated, while all structurally ambiguous noun phrases are passed to the second phase "as is". In the second phase, the distributional statistics gathered in the first phase are used to predict the strength of alternative modifier-modified links within ambiguous phrases. For example, we may have multiple unambiguous occurrences of "insider trading", while very few of "trading case". At the same time, there are numerous phrases such as "insider trading case", "insider trading legislation", etc., where the pair "insider trading" remains stable while the other elements get changed, and significantly fewer cases where, say, "trading case" is constant and the other words change.

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 Tipster 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 Linguistic Phrase Stream

To test the effectiveness of noun phrases, we choose a stream which utilize simple noun phrases as atomic index terms. The original text is part-of-speech tagged and stemmed. The noun phrases are then identified by regular expression rules on the part-of-speech tags. The major rules are:

1. a sequence of modifiers (vbn|vbg|jj) followed by at least one noun, such as: "cryonic suspend", "air traffic control system";
2. proper noun(s) modifying a noun, such as: "u.s. citizen", "china trade";
3. proper noun(s) (might contain '&'), such as: "warren commission", "national air traffic controller".

The length of phrases is limited to maximum 7 words.

The Smart retrieval system gives us many choices of weighting schemes. In order to choose an effective weighting scheme, some experiments were carried out. The result suggests that the best weighting for phrases is lnc.lsc (lnc for documents, lsc for queries).

3.3 Name Stream

Proper names, of people, places, events, organizations, etc., are often critical in deciding relevance of a document. Since names are traditionally capitalized in English text, spotting them is relatively easy, most of the time. Many names are composed of more than a single word, in which case all words that make up the name are capitalized, except for prepositions and such, e.g., The United States of America. It is important that all names recognized in text, including those made up of multiple words, e.g., South Africa or Social Security, are represented as tokens, and not broken into single words, e.g., South and Africa, which may turn out to be different names altogether by themselves. On the other hand, we need to make sure that variants of the same name are indeed recognized as such, e.g., U.S. President Bill Clinton and President Clinton, with a degree of confidence. One simple method, which we use in our system, is to represent a compound name dually, as a compound token and as a set of single-word terms. This way, if a corresponding full name variant cannot be found in a document, its component words matches can still add to the document score. A more accurate, but arguably more expensive method would be to use a substring comparison procedure to recognize variants before matching.

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
u.s+constitution
gun+control+legislation
national+railroad+transportation+corporation
superfund+hazardous+waste+cleanup+programme
u.s+government
united+states
exxon+valdez
dow_corning+corporation
chairman+julius+d+winer
new+york
wall+street+journal

4.0 OTHER STREAMS

4.1 Stems Stream

The stems stream is the simplest, yet, it turns out, the most effective of all streams, a backbone in our multi-stream model. It consists of stemmed non-stop single-word tokens (plus hyphenated phrases). Our early experiments with multi-stream indexing using SMART suggested that the most effective weighting of this stream is lnc.ltc, which yields the best average precision, whereas lnc.ntc slightly sacrifices the average precision, but gives better recall.

4.2 Unstemmed Word Stream

For the routing experiment with PRISE we used also a plain text stream. This stream was obtained by indexing the text of the documents "as is" without stemming or any other processing and running the unprocessed text of the queries against that index.

4.3 Fragment Stream

For the routing experiments with PRISE we also used a stream of fragments. This was the result of spliting the documents of the STEM stream into fragments of constant length (1024 characters) and indexing each fragment as if it were a different document. The queries used with this stream were the usual stem queries. For each query, the resulting ranking was filtered to keep, for each document, the highest score obtained by the fragments of that document.

5.0 MERGING STRATEGY

The results obtained from different streams, i.e., ranked lists of documents retrieved from searching each stream, were merged into a single final ranking. The final score of each document is computed by combining the relevance estimate given in the scores of all the streams where it is retrieved. The merge is based on two factors:

1. document's relevance estimates from various streams;
2. the overall retrieval effectiveness of those streams in general.

A more effective stream will carry a higher weight, and a higher ranking in that stream will have a larger effect to move the document up in the merged ranking. The entire merging procedure is carried out in two steps. The first step is same-system inter-stream merge, in which the rankings obtained from the streams using the same retrieval system (e.g., SMART) are combined together. The second step is the inter-system merge. In each case a different merging algorithm is used.

5.1 Inter-stream merging in SMART

Each stream carries some unique type of information about the documents it indexes. It is therefore critical that the merging process knows how to properly combine and translate the stream-level rankings into one global ranking.

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 range of ranks.

This estimate varies from stream to stream. For example, an estimate of the stems stream may consist of the following (see also Table 1): the precision within top 10 is 39%, between the 11th to the 20th is 34%, etc.

TABLE 1. Precision distribution

RANK	PRECISION
1-10	0.39
11-20	0.34
31-50	0.28
51-100	0.21
101-200	0.13

The final rank of a document (d) is defined as:

$$\sum_{'i = 1..4} \{A\,[i] \times (\mathrm{Score[i](d)} \times \mathrm{Prec[i](rank)})\}$$

where i = 1..4 stands for the four different streams we used; A[i] is the weight for each stream which is acquired through experiments; Score[i](d) is the normalized score of the document against the query within the stream i; Prec[i](rank) is the estimate precision described above.

5.2 Inter-stream merging in PRISE

There are many ways in which the information obtained from different streams can be merged in order to obtain the final results. In our experiments with PRISE we tried several methods and chose the ones that seemed to be the best for our official results. The experiments that helped us chose the best merging technique were performed using a 500-MBytes dry-run collection that was created using past TRECs data for which we already had relevance information available. Some of the methods we tried include:

- linear combinations: i.e. multiply the score obtained by each stream by a constant and then add the score of all streams together.
- change the scores of the documents in order to push to the top of the ranking all documents that appeared in more than one stream. The first group of documents of the resulting ranking appeared in n streams, the second group in n-1, etc. At the bottom of this ranking would be the documents that appeared in only one stream.
- a combination of the previous two: multiply each stream by a different constant (determined by experiment) and add all streams together. Then for each document multiply the score by a number which is a function of the number of streams in which that document appeared.

In the dry run experiments that we performed with PRISE the third method achieved the highest performance. Using this method we obtained increases in performance of around 40% over the performance of the stem stream alone. The assumption that supports the use of the third method is that each additional stream in which a document appears adds to the evidence that the document may be relevant. The function that we used to multiply the score of each document (which was a linear combination of the scores of all the streams in which it appeared) was:

(0.9 + number-of-streams/10)

where *number-of-streams* is the number of streams in which the document appears. Thus, if the document appears in only one ranking the score is not changed, if it appears in 2 rankings the new score is 1.1 times the old score, etc.

5.3 Inter-system merging

The same merging technique can be also used to merge results from two different retrieval systems. Conceivably the less similarity two systems have, the better result can be expected from the merge. The reason is that dissimilar systems tends to make their decisions based on different document and query features. Incorporating more information is the key for the merging to be successful. Nonetheless, even with two rather similar systems, Smart and Prise, we still see 10-20% improvement in the runs where we have used the inter-system merging.

6.0 SMART AND PRISE

6.1 SMART system as used in TREC-5

We used Smart system V.11 available from Cornell University. In some runs, including NLP Track evaluations **sbase1, sbase2, genlp2, genlp3,** SMART own text pre-processing facilities were used: stopword filtering, suffix stemming, proper noun detection, statistical phrases grouping using high frequency adjacent word pairs (bigrams). In all other runs where the multi-stream model was utilized, these simple techniques are replaced by more advanced linguistic processes that include lexicon-based morphological normalization, proper names recognition, syntactic phrase extraction and so forth.

6.2 PRISE system as used in TREC-5

Prise is a statistical information retrieval system developed by Donna Harman at NIST. The system has been unchanged since TREC-4. For details, the reader is referred to our earlier TREC-based publications (e.g., Strzalkowski, 1994), or (Harman&Candela, 1991).

7.0 TREC-5 AD-HOC RUNS

7.1 Automatic Ad-Hoc Runs

7.1.1 Ad-hoc experiments using PRISE

For the automatic ad-hoc experiment with PRISE we used the following streams, along with the following merging coefficients:

Coeff	Stream
1	*stems*
4	*locality with n=20*
1	*fragments*
1	*words*
3	*pairs*
1	*names*

There was not enough time to run experiments involving many other possible combinations. The numbers listed above were determined through limited experiments using the dry-run collection. There was no time at all to run experiments for the fragments and plain-text streams.

The scoring function used with both the ad-hoc and routing experiments is as follows:

final-score(d) = score(d)(0.9 + number-of-streams(d)/10)*

where number-of-streams is the number of streams in which document *d* is retrieved.

7.1.2 Ad-hoc experiments using SMART

For the automatic ad-hoc experiment with PRISE we used the following streams, along with the following merging coefficients:

Coeff	Stream
4	*stems*
3	*phrases*
3	*pairs*
1	*names*

The scoring function is discussed in detail in section 6.2

7.2 Query Expansion Experiments in Manual Runs

The purpose of query expansion in information retrieval is to make the user query resemble more closely the documents it is expected to retrieve. In a typical situation, content terms from documents judged relevant documents are added to the query, and other terms weights are adjusted in order to reflect this new evidence. This process can be performed automatically using a relevance feedback method, e.g., Roccio's, or it can be done manually by the user. A serious problem with the 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. Additionally, it depends on the partial relevance information, which is normally unavailable, or unreliable.

An alternative to term-only expansion is a full-text expansion which we tried for the first time this year in TREC-5. In our approach, queries are expanded by pasting in entire sentences, paragraphs, and other sequences directly from ANY document. To make this process efficient, we first perform a search with the original, un-expanded queries, and then use top N (10, 20) returned documents for query expansion. These documents are not judged on relevancy nor assumed relevant, instead, they are scanned for passages that contain a concept referred to in the query. Subject to some further "fitness criteria", these passages are then imported verbatim into the query. This can be accomplished manually, as we did in TREC-5 main adhoc manual runs, or automatically, as we tried in one of the NLP Track runs. The resulting expanded queries undergo the usual text processing steps, before the search is run again.

The initial evaluations indicate that queries expanded this way are improving the system's performance (precision and recall) by as much as 40%. At this time, automatic text expansion produces less effective queries than manual expansion, primarily due to a relatively unsophisticated mechanism used to identify concepts in the queries (see section 10.1 for details).

7.2.1 Query expansion guidelines

We have adopted the following guidelines for query expansion. They were constructed to observe realistic limits of the manual process, and to prepare ground for eventual automation.

1. SMART and Prise are run using the original queries, with all streams, and our regular merge.
2. Users (i.e., team members) get top 10 docs retrieved by each of "their" queries (we used 5 to 10 queries per person, taking advantage of our team size).
3. Each query is manually expanded using phrases, sentences, etc. found in any of the top 10 documents for this query. Text can both added and deleted, though care is taken to assure that the final query has the same format as the original, and that all expressions added are well-formed English strings (though not necessarily sentences) ended with a period. **A limit of 30 minutes per query in a single block of time is observed.**

4. Expanded queries are sent for NL processing, then run through all streams and search engines as in step 1. This constitutes our genrl3 run.[2]
5. Queries, and new top 10 documents are returned to their "expanders" who are now asked to judge relevance.[3]
6. A relevance feedback is run based on top 10 relevant/non-relevant info. This constitutes our genrl4 run.

The actual time table used to prepare TREC-5 manual run is given below:

July 08, noon: queries and top docs distributed
July 10, noon: first round expansion due
July 10, 4 pm: extended queries NLP processed
July 11, noon: extended queries and new top 10 docs distributed
July 11, 4 pm: additional query revisions, if any, due
July 11, 6 pm: all revised queries NLP processed
July 12, noon: new top 10 docs redistributed for revised queries
July 12, 3 pm: relevant/non-relevant judgements on final top 10 due

8.0 TREC-5 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 routing 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.

8.1 Standard Routing

8.1.1 Routing experiments using PRISE

For the standard routing experiment using PRISE we used the following streams and coefficients:

Coeff	Stream
1	*stems*
4	*locality with n=20*
3	*pairs*
1	*names*

2. A few queries were "corrected" to fix formatting problems. Other changes were allowed if expanders felt "unhappy" with their queries.

3. For some queries, no relevant documents were found in top 10. These queries were further expanded and rerun one more time.

The value of the coefficients was determined using the dry-run collection.

The same scoring function was used with the ad-hoc and routing experiments:

$$final\_score(d) = score(d)*(.9 + number\text{-}of\text{-}streams(d)/10)$$

where number-of-streams is the number of streams in which document *d* is retrieved.

8.1.2 Routing experiments using SMART

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 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 Rocchio 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.

For the routing runs with SMART we used the following streams, along with the merge coefficients:

Coeff	Stream
4	*stems*
3	*phrases*
3	*pairs*
1	*names*

8.2 Classification-based Routing

One of our routing streams was based upon a probabilistic classification system developed at Lockheed Martin. This system has been in development for less than one year, so many of its parameters have not yet been optimized. Also, only capitalized, stemmed single words are used as terms. Bi-grams, phrases, and extracted information will be added in the future.

The system generates routing scores for documents using three complementary components.

1 . A probabilistic scorer, which assigns a score to a document for a topic based upon the probability that the document belongs to the topic. Estimated probabilities for distinguishing terms (number of term occurrences divided by the number of occurrences of all terms in the training set) are gathered from training documents, and the union of all of the distinguishing terms for all of the topics defines a multinomial distribution. In considering a document to be routed, these estimated probabilities are combined with the count of terms which exist in the document to determine the probabilities that the document belongs to each of the topics [Guthrie et al, 1996].

The probabilistic scorer as described above has two features which need to be overcome to get good routing scores in a TREC type of situation. First, for each term there is a very good probability that the term does not occur in a document. This leads to good scores for documents which have none of the distinguishing terms for a topic. Second, even for documents which have some of the distinguishing terms for a topic, no special weight is given to key terms in the query. This leads to good scores for `near miss' documents, for example, documents about Russian joint ventures scoring highly for Topic 3, Japanese Joint Ventures.

2. The first feature, good scores for documents which contain no distinguishing terms, is overcome by including a score based upon the document frequency (number of relevant topic training documents which contain the term divided by the number of topic training documents) of `required terms'. For each topic, the document is compared to a list of 10 to 20 of these terms, which is usually a subset of the distinguishing terms. If none of these terms for a topic occur in the document, the document is removed from consideration for that topic. For those required terms which do occur in the document, a score is calculated which is a sum of a function of the document frequencies of these terms. In addition to eliminating good scores for documents which contain no distinguishing terms, this score complements the probabilistic score because the probabilistic score does not increase the importance of terms which occur only once in most of the topic training documents, thus having a fairly low estimated probability but a high relevancy, or decrease the importance of terms which occur many times in only one of the topic training documents, thus having a fairly high estimated probability but a low relevancy.

3. The second feature, good scores for near miss documents, is overcome by including a score for each topic based upon a manually written boolean expression for each topic. If a document contains terms which fulfill the expression it gets a boost to its score, but if it does not fulfill the expression the document is not eliminated from consideration. The boolean expression helps weed out near misses, while not overly penalizing those documents which are relevant but which do not fulfill the expression.

For a new routing system the performance was satisfactory, scoring just below the median 11 point average precision. Unfortunately, the software had a small bug in the probabilistic portion which reduced the score somewhat. Due to a misplaced `else' statement, a counter which was supposed to be counting the number of terms which were not distinguishing for any topic was actually counting, for each topic, the number of words which were not required terms. This resulted in a count which was incorrect, but in a rough sense was about 50 times what it was supposed to be, giving a final result which was still reasonable. Correcting the error increased the 11 point average precision over 45 topics from 0.1867 to 0.2299, a 23% improvement, and the number of relevant document retrieved from 2798 to 3375, a 21% improvement.

Future improvements for this system include the consideration of bi-grams, phrases, and extracted information, additional automation in term selection and boolean expression creation, improved stemming (the current stemmer is rule-based), and optimization of all of the parameters.

9.0 SUMMARY OF RESULTS

We submitted the total of 6 official runs in the main evaluation, and 4 official NLP track runs. In addition, 2 NLP baseline runs using SMART system have been submitted.

9.1 Adhoc runs

Adhoc runs were all in category A (entire 2 GByte database). The following 4 ad-hoc runs were submitted:

- **genrl1**: automatic run, short queries, with auto feedback on top 10
- **genrl2**: automatic run, long queries
- **genrl3**: manual run, long queries, with query expansion
- **genrl4**: manual run, long queries, with query expansion and auto feedback on top 10

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 long 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: 252

<title> Topic: Combating Alien Smuggling

<desc> Description:

What steps are being taken by governmental or even private entities world-wide to stop the smuggling of aliens.

<narr> Narrative:

To be relevant, a document must describe an effort being made (other than routine border patrols) in any country of the world to prevent the illegal penetration of aliens across borders.

</top>

TABLE 2. Precision changes across Ad-Hoc runs

PRECISION	GENRL1 automatic	GENRL2 automatic	GENRL3 manual	GENRL4 manual
11pt. average %change	0.1524	0.2093 +37%	0.2847 +87%	0.2741 +80%
R-Precision	0.1965	0.2441	0.3126	0.3042
At 10 docs	0.3064	0.3809	0.5191	0.3042
At 100 docs	0.1694	0.3000	0.2615	0.2604

9.2 Routing runs

Routing submissions included two official runs:

- genrl5: automatic run, long queries, standard SMART+Prise run
- genrl6: automatic run, long queries, using multi-bin classification approach

A mistake was made in selecting weighting scheme in SMART portion of genrl5: a wrong weighting scheme (lnn) was accidentally used for indexing the pivotal stem stream in the test collection. Once we re-build the index with the correct lnc weighting, the result have improved substantially, as shown in the table below. The average precision checked against the summary posted by NIST shows now 4 queries at the best, 28 above median, and 13 below median. Note that this correction is independent of any specific database or queries.

TABLE 3. Average Precision on 45 routing queries: GENRL5

IR ENGINE	Corrected Prec.	Corrected R-Prec.	Official Prec.
SMART	0.2755	0.3145	0.0631
PRISE	0.2099	0.2473	0.2099
GENRL5	0.3023	0.3359	0.1968

The merge of Smart and Prise improved 9.7% on the average precision over the best individual component.The classification run (genrl6) was a merge of a classification scheme routing developed at Lockheed Martin, and standard

Prise routing used in previous TRECs. Again, a correction of a small error in the classification scheme improves our results as shown below.

TABLE 4. Average Precision on 45 routing queries: GENRL6

IR ENGINE	Corrected Prec.	Official Prec.
LM CLASS. %change	0.2299 +24%	0.1867
PRISE	0.2099	0.2099
GENRL6 %change	0.2575 +16%	0.2222

Note that the classification scheme did quite well, outperforming (after correction) the standard Prise routing by some 10%. Our merging algorithm is also performing well consistently adding some 10% precision, provided that component runs are themselves relatively good.

10.0 NLP TRACK

The NLP Track was a specialized smaller-scale evaluation to experiment with more advanced NLP techniques. Our focus this year was on (1) evaluating value of special-purpose terms such as foreign-country references, and single-sense words, and (2) approaches to automatic full-text query expansion methods.

10.1 Automatic runs in NLP Track

We generated three automatic runs for the NLP track: **genlp1, genlp2, genlp3.** We focused on experimenting NLP related automatic query enhancement techniques. **genlp1** is the automatic run that uses the same multi-stream retrieval model as used in **genrl2** main evaluation run, with two added enhancements: foreign country tagging and hyphened phrases tracking. In **genlp2**, we tested foreign country tagging exclusively against the corresponding SMART baseline (**sbase2**). Finally, in **genlp3**, we attempted to explore means of automating the full-text query expansion technique used in manual runs (**genrl3, genlp4**). In addition to the official runs, we also discuss an experiment with weighting of single-sense words in section 11.1.4.

10.1.1 Hyphenation

We used occurrences of hyphenated phrases in text as a guide for extracting other multi-word terms for indexing. Many semi-fixed expressions in English are occasionally hyphenated, which may indicate that their non-hyphenated occurrences should also be treated as single terms. In this experiment, we collected all the hyphenated words from the corpora (the less meaningful ones are eliminated by setting a threshold on the number of occurrences they appear), such as: *alien-smuggle, quick-freeze, roto-router, heart-surgery cigarette-smoke, lung-cancer, per-capita*, etc. The next step was to identify all the occurrences of those phrases in the collection and in the queries where they were not hyphenated and add the normalized term.

Unfortunately, the result of the stems stream shows slight deterioration comparing with performance before adding the hyphened phrases. Among the 33 applicable queries, 22 queries show precision loss (on the average 43% per query) and only 11 queries show improvement (on the average 45% per query).

10.1.2 Foreign Country Tagging

For queries involving references to foreign countries, either direct, or indirect, e.g., good of foreign manufacture, we added special tokens for each reference of the concept 'foreign'. The identification is done simply by looking up certain key words and phrases (e.g. *foreign, other countries, international,* etc.). Using a list of foreign countries and major cities acquired from the Internet, we tagged the documents in the collection with the same special "foreign" token whenever a foreign country or city was mentioned.

Only 10 queries (out of 45) were affected. Comparing with the base run, 9 out of these 10 show improvement, and only one shows a modest 5% performance loss. On the average, the precision gain is 27% for those queries. The result may suggest that type-tagging of terms in general, e.g., people, organizations, locations, etc. may lead to significant improvement in retrieval accuracy, a subject that has been the focus of much debate in Tipster community. The challenge is to identify a sufficient number of basic categories so that they can be found in a number of different queries, and such that an efficient object identifier can be implemented for them.

10.1.3 Concept Expansion

In our manual runs (**genrl3, genrl4, genlp4**), we tested the full-text query expansion, in which original queries were liberally augmented with text copied from database documents. The results were most encouraging, which prompted us to investigate ways of performing such expansions automatically.

One way to approximate the manual text selection process, we reasoned, is to focus on those text passages that refer to some key concepts identified in the query, for example, "alien smuggling" for query 252 and "cryonic suspension" for query 253.

The key concepts (for now limited to noun phrases) are identified by their repetitions as well as their relative locations within the query, e.g., in the title. We then take the top 100 retrieved documents for each query in an unexpanded query run, e.g., **genlp1**, and extract all paragraphs which contain references to any of the key concepts identified in the original query. These paragraphs are the pasted verbatim into the query. The original portion of the query may be saved in a special field to allow differential weighting. The expanded query were then run against the baseline index, producing **genlp3** ranking. Please note that, unlike **genrl3,** or **genlp4**, this run uses only one stream, namely stems stream, additionally augmented with SMART bigram phrases. This means that direct comparisons with any manual extension runs may not be valid here.

The above, clearly simplistic technique has produced some interesting results. Out of the fifty queries we tested, 34 has undergone the expansion. Among these 34 queries, we noted precision gains in 13, precision loss in 18 queries, with 3 more basically unchanged. However, for these queries where the improvement did occur it was very substantial indeed: the average gain was 754% in 11-pt precision, while the average loss (for the queries that lost precision) was only 140%. Overall, we still can see a 7% improvement on all 50 queries (vs. 40%+ when manual expansion is used).

In conclusion, the experiment shows that picking up the right paragraphs from documents to expand the query can indeed improve the performance dramatically. The future challenge is to devise a more precise automatic means to select those "good" paragraphs.

10.1.4 Single-sense Enhancement

Many words, when considered out of context, display more than one sense in which they can be used. When such word are used in actual text they may assume any of their possible senses, which can only be determined by examining the context. This has been a problem for word-based IR systems, and have spurred a number of largely unsuccessful attempts at sense disambiguation in text indexing. 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 just more reliable as content indicators.

We found that single-sense[4] words tend to be more specific and thus more informative. We added more weight to terms that we considered low-number-of-senses, with single-sense words receiving the highest premium (duplicate standard tf*idf weight). The results were mixed with the average precision gaining a modest 4.6%.

10.2 Manual run in NLP Track

genlp4 is a counterpart to the **genrl3** adhoc run. It is the only manual run we submitted in the NLP track, which uses the manually expanded queries against the multi-stream indexes. The goal is to see how the system performs with relatively high quality and long queries and compare it with the Smart baselines.

The evaluation over 45 queries of **genlp4** and the comparison with other manual NLP runs are shown in the table below. The average precision of most queries are above median.

TABLE 5. Manual NLP track run with query expansion (GENLP4)

Recall	814 out of 1064
11-pt Avg. Precision	0.3176
R-Precision	0.3090
Queries with best avg. precision	22
Queries with above avg. precision	17
Queries below average	6

To further comparing the retrieval performance of our multi-stream model and Smart with bi-grams, the un-processed manually expanded queries are then run against the Smart baseline index, the result shows noticeable improvement.

10.3 Summary of NLP Track runs

The following runs were obtained:

genlp1: automatic multi-stream run with foreign country tagging and hyphened phrases.

genlp2: automatic single-stream run (stems and bigrams) with foreign country tagging.

genlp3: automatic, single-stream run with automatic the full-text query expansion.

genlp4: manual, multi-stream run with manual full-text query expansion.

sbase1: SMART baseline with stems and bigrams on “short” queries

sbase2: SMART baseline with stems and bigrams on “long” queries

sbase3: SMART baseline with stems and bigrams on full-text expanded queries

4. Or near-single-sense words, with a predominant single-sense. Generally, the fewer senses a word can have, the more reliable index term it appears to be.

TABLE 6. Precision improvement in NLP Track runs using

PRECISION	SBASE1	SBASE2	GENLP1	GENLP2	GENLP3	SBASE3	GENRL4
11pt. average %change	0.1478	0.2078 +41.0	0.1773 +20.0	0.2083 +41.0	0.2220 +50.0	0.2992 +102.0	0.3176 +115.0
R-Precision %change	0.1609	0.2176 +35.0	0.1776 +10.0	0.2121 +32.0	0.2242 +39.0	0.3074 +91.0	0.3091 +92.0
At 10 docs %change	0.1578	0.2044 +30.0	0.2044 +30.0	0.2044 +30.0	0.2089 +32.0	0.3089 +96.0	0.3156 +100.0
At 100 docs %chabge	0.0544	0.0696 +28.0	0.0664 +22.0	0.0713 +31.0	0.0709 +30.0	0.0929 +71.0	0.0998 +83.0

10.4 Stream Performance Evaluation

The weighting schemes used in genlp1 and genlp4 are the same:

TABLE 7. Stream weighting in NLP Track runs (genlp1 & 4)

STREAM	Weighting Scheme
Stems	lnc.ntn
Phrases	ltn.ntn
H+M Pairs	ltn.nsn
Names	ltn.ntn

Selecting the optimal weighting for each stream is essential. The issue is further complicated by the fact that the optimal weighting vary from collection to collection. The weighting schemes used in the submitted genlp1 and genlp4 appear reasonable, but they are not optimal. Had we chosen lnc.ltn weighting on the stems stream, genlp1 would have moved up to 0.1883 and genlp4 up to 0.2792.

The average precision over 45 queries stream-vise are:

TABLE 8. How different streams perform relative to one another (11-pt avg. Prec)

STREAM	genlp1	genlp4
Stems	0.1682	0.2626
Phrase	0.1233	0.2365
H+M Pairs	0.0755	0.2040
Names	0.0844	0.0608

The average precision of different stream merging with the strongest single-stream(stems) retrieval are shown in Table 9.[5]

TABLE 9. How merging improves precision, wrt. queries used

Which Streams MERGED?	genlp1 %change	genlp4 %change
all 4	+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

The results indicate that syntactic phrases seem to be more effective with longer the queries.

10.5 Baseline Runs

Two baselines were generated for the NLP track: **sbase1** using “short” queries (the <desc> field only) and **sbase2** that utilizes all the fields in the adhoc topics (“long” queries). The document test collection was category B (250MBytes Wall Street Journal data). Both baselines were obtained using standard SMART processes, including “statistical phrase” terms, i.e., high frequency adjacent word pairs (bi-grams).[6]

11.0 Experiments in Stylistic Analysis

Texts vary not only by topic. *Stylistic* variation between texts of the same topic is often at least as noticeable as the *topical* variation between texts of different topic but same genre or variety; style is, broadly defined, the difference between two ways of saying the same thing.

Stylistic variation in a given text can occur in many ways and on many linguistic levels: lexical choice, choice of syntactic structures, choice of cohesion markers on a textual level, and so forth. In these experiments we measure several different types of simple *stylistic items*: lexical statistics such as average word length, long word counts, type/token ratios, pronoun counts and digit counts; syntactic statistics such as average sentence length and some parsing statistics and combine them using multivariate techniques (Karlgren and Cutting, 1994). We use the Wall Street Journal corpus for TREC-4 as a training set: we have attempted to find statistically significant stylistic differences between documents that have been judged relevant for some query on the one hand and documents that were not judged relevant for any query at all on the other.

We did find such differences; for most metrics tested, the difference was statistically significant even by univariate[7] tests, even if the difference between texts retrieved by some system and non-retrieved texts was larger by far than the difference between relevant and non-relevant. In summary, our results are that retrieved highly ranked texts - both relevant and non-relevant - are longer[8] , with a more complex sentence structure than the rest of the corpus, and that rel-

5. First, all four stream participate the merge (officially submitted). Second, the name stream is taken away; Next, the pair and name stream are taken away; Next, the phrase and name stream are left out. Finally, the phrase and pair stream are left out.
6. We had to resubmit the baselines after the official results were obtained from NIST, because of a mistake in the term weighting scheme used.
7. Mann Whitney U
8. Which also has been observed, pointed out, and utilized by the Cornell research group at the latest TREC conference (Buckley et al, 1995).

evant texts differ from nonrelevant in that they tend towards more complexity - textual, syntactic, and lexical - on most measurements. Moreover, we found that these differences varied for subsets of the Wall Street Journal: certain types of article had a higher percentage relevant documents than others. (Karlgren, 1996).

11.1 Visualizing Stylistic Variation

Stylistic variation is not unrelated to topic: obviously certain topics will be more formal than others; others will be more technical; yet others more discursive. How much user preferences for this are reflected in the query itself or the initially retrieved set of documents is an open question. We envision using stylistic data primarily in an interactive setting, in an display tool as a user-manipulable filter for interactive retrieval.

We have a prototype tool which will compute stylistic statistics for a set of texts, and which will then display the texts in a 2-D plot for any pair of statistics chosen. It is clear that using simple variables in this way is risky: firstly, what variation the display models is unclear to the user; secondly, risk of random or chaotic variation is great. We have experimented with combining scores from the various variables in linear weightings using principal components analysis (Karlgren and Straszheim, 1997).

A useful strategy might be to pick a couple of parameters with a seemingly high spread. As example material we use here the top 50 retrieved texts for our system for query 203 “What is the economic impact of recycling tires?” together with the top 50 texts retrieved by Altavista from the World Wide Web for the same query. We find an example pair of variables which seems to disperse the material quite well as shown in Figure 12.1. The WWW material is marked with open circles; the TREC data with filled squares. Unsurprisingly, the WWW material is stylistically more heterogenous than the TREC data: the TREC outliers are Patent and Federal Register texts.

11.2 Stylistics As A Way To Improve Precision

Now, the realization that stylistic variation is related to topical variation led us to perform some experiments specifically to improve our average precision in the TREC evaluation. We took some queries from previous years and used a system essentially like the one we used for this year’s submissions to produce rankings for the documents.

We then tried to find methods that would identify non-relevant documents from the list of 1000 retrieved and submitted documents. These documents would then be moved to the end of the list, hopefully improving the evaluation results for the query. We used the C4.5 classification tool (Quinlan, 1993) which takes multivariate material and produces simple rules to partition items into classes using the variable values.

We knew from our first experiments that there are statistically significant differences between relevant and non-relevant documents in the TREC Wall Street Journal material as a whole. So far, so good, but the problem comes trying to apply these results on a query-by-query basis. The sets retrieved for each query are different stylistically; the genres and styles vary from topic to topic and thus from query to query. If we learn rules to distinguish relevant from non-relevant for the entire corpus and try to apply the rules across queries, we find we either degrade performance or at the least do not improve it.

The two rules below are examples of this. We found that these two rules work quite well for their respective training corpora and when tested on this year’s material they improve results for many queries as can be seen in Figure 12.2. However, both rules suffer breakdowns on at least one query; this reduces the advantage gained from the other queries so that average precision is a tad lower than for the unordered set.

The consequence is that to make use of stylistic variation for reliable relevance grading we need a query typology: each query must be identified for likely style preferences. This remains future work and results from experiments in query categorization are pending.

Rule 4: *Digits > 0.0179775* -> class non-relevant [95.0%]

Rule 876: *Word count <= 1308* &
Chars per word <= 5.10619 &
Digits > 0.0119782 &
Words per sentence > 17.4583 &
1st person pronouns <= 0.0166667 &
Persuasive adverbs <= 0.00234467 -> class non-relevant [90.3%]

TABLE 10. Effect on Ave. Precision

Baseline PRISE	Using Rule 4	Using Rule 876
0.1460	0.1459	0.1452

12.0 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 natural language processing is not as effective as we once hoped in to obtain better indexing and better term representations of queries. Using linguistic terms, such as phrases, head-modifier pairs, names, or even simple concepts does help to improve retrieval precision, but the gains remained quite modest. On the other hand, full text query expansion works remarkably well. Our main effort in the immediate future will be to explore ways to achieve at least partial automation of this process. An initial experiment in this direction has been performed as part of NLP Track (genlp3 run), and the results are encouraging.

ACKNOWLEDGEMENTS. We would like to thank Donna Harman of NIST for making her PRISE system available to us since the beginning of TREC. Will Rogers provided valuable assistance in installing updated versions of PRISE at NYU and Rutgers. We would also like to thank Ralph Weischedel for providing and assisting in the use of the BBN's part of speech tagger at NYU. This paper is based upon work supported in part by the Advanced Research Projects Agency under Tipster Phase-2 Contract 94-FI57900-000, and the National Science Foundation under Grant IRI-93-02615.

REFERENCES

Chris Buckley, Amit Singhal, Mandar Mitra, Gerard Salton. 1995. ``New Retrieval Approches Using SMART: TREC 4''. In Proceedings of TREC4.

Guthrie, Louise and Leistensnider, James, `A Simple Probabilistic Approach to Classification and Routing', Proceedings of the TIPSTER Text Program Phase II Workshop, Sponsored by Defense Advanced Research Projects Agency, May 6-8, 1996.

Jussi Karlgren. 1996.``Stylistic Variation in an Information Retrieval Experiment'' In Proceedings NeMLaP 2, Bilkent, September 1996. Ankara: Bilkent University. (In the Computation and Language E-Print Archive: cmp-lg/9608003).

Jussi Karlgren and Douglass Cutting. 1994. ``Recognizing Text Genres with Simple Metrics Using Discriminant Analysis'', Proceedings of COLING 94, Kyoto. (In the Computation and Language E-Print Archive: cmp-lg/9410008).

Jussi Karlgren and Troy Straszheim. 1997. ``Visualizing Stylistic Variation.'' In the Proceedings of the 30th HICSS, Maui.

J. Ross Quinlan. 1993. C4.5: Programs for Machine Learning. San Mateo: Morgan Kaufmann.

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

Strzalkowski, Tomek, Jose Perez-Carballo and Mihnea Marinescu. 1995. "Natural Language Information Retirieval: TREC-3 Report." Proceedings of the Third Text REtrieval Conference (TREC-3), NIST Special Publication 500-225, pp. 39-53.

Strzalkowski, Tomek, Jose Perez-Carballo and Mihnea Marinescu. 1996. "Natural Language Information Retirieval: TREC-4 Report." Proceedings of the Third Text REtrieval Conference (TREC-4), NIST Special Publication 500-2xx.

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. 1993. "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.