NATURAL LANGUAGE INFORMATION RETRIEVAL: TREC-3 REPORT

Tomek Strzalkowski, Jose Perez Carballo and Mihnea Marinescu
Courant Institute of Mathematical Sciences
New York University
715 Broadway, rm. 704
New York, NY 10003
tomek@cs.nyu.edu

ABSTRACT

In this paper we report on the recent developments in NYU's natural language information retrieval system, especially as related to the 3rd Text Retrieval Conference (TREC-3). The main characteristic of this system is the use of advanced natural language processing to enhance the effectiveness of term-based document retrieval. The system is designed around a traditional statistical backbone consisting of the indexer module, which builds inverted index files from pre-processed documents, and a retrieval engine which searches and ranks the documents in response to user queries. Natural language processing is used to (1) preprocess the documents in order to extract content-carrying terms, (2) discover inter-term dependencies and build a conceptual hierarchy specific to the database domain, and (3) process user's natural language requests into effective search queries. For the present TREC-3 effort, the total of 3.3 GBytes of text articles have been processed (Tipster disks 1 through 3), including material from the Wall Street Journal, the Associated Press newswire, the Federal Register, Ziff Communications's Computer Library, Department of Energy abstracts, U.S. Patents and the San Jose Mercury News, totaling more than 500 million words of English. Since the TREC-2 conference, many components of the system have been redesigned to facilitate its scalability to deal with ever increasing amounts of data. In particular, a randomized index-splitting mechanism has been installed which allows the system to create a number of smaller indexes that can be independently and efficiently searched.

INTRODUCTION

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* is important by itself. 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

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
NOV 1994		**02-11-1994 to 04-11-1994**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Natural Language Information Retrieval: TREC-3 Report	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **New York University,Courant Institute of Mathematical Sciences,715 Broadway, rm. 704,New York,NY,10003**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Third Text Retrieval Conference (TREC-3), Gaithersburg, MD, November 2-4, 1994

14. ABSTRACT

In this paper we report on the recent developments in NYU?s natural language information retrieval system, especially as related to the 3rd Text Retrieval Conference (TREC-3). The main characteristic of this system is the use of advanced natural language processing to enhance the effectiveness of term-based document retrieval. The system is designed around a traditional statistical backbone consisting of the indexer module, which builds inverted index files from pre-processed documents, and a retrieval engine which searches and ranks the documents in response to user queries. Natural language processing is used to (1) preprocess the documents in order to extract content-carrying terms, (2) discover inter-term dependencies and build a conceptual hierarchy specific to the database domain, and (3) process user?s natural language requests into effective search queries. For the present TREC-3 effort, the total of 3.3 GBytes of text articles have been processed (Tipster disks 1 through 3), including material from the Wall Street Journal, the Associated Press newswire, the Federal Register, Ziff Communications?s Computer Library, Department of Energy abstracts, U.S. Patents and the San Jose Mercury News, totaling more than 500 million words of English. Since the TREC-2 conference, many components of the system have been redesigned to facilitate its scalability to deal with ever increasing amounts of data. In particular, a randomized index-splitting mechanism has been installed which allows the system to create a number of smaller indexes that can be independently and efficiently searched.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**15**	

low. In large databases, such as TIPSTER, the use of phrasal terms is not just desirable, it becomes necessary.

An accurate syntactic analysis is an essential prerequisite for selection of phrasal terms. Various statistical methods, e.g., based on word co-occurrences and mutual information, as well as partial parsing techniques, are prone to high error rates (sometimes as high as 50%), turning out many unwanted associations. Therefore a good, fast parser is necessary, but it is by no means sufficient. While syntactic phrases are often better indicators of content than 'statistical phrases' — where words are grouped solely on the basis of physical proximity (e.g., "college junior" is not the same as "junior college") — the creation of compound terms makes term matching process more complex since in addition to the usual problems of synonymy and subsumption, one must deal with their structure (e.g., "college junior" is the same as "junior in college"). In order to deal with structure, the parser's output needs to be "normalized" or "regularized" so that complex terms with the same or closely related meanings would indeed receive matching representations. This goal has been achieved to a certain extent in the present work. As it will be discussed in more detail below, indexing terms were selected from among head-modifier pairs extracted from predicate-argument representations of sentences.

Introduction of compound terms also complicates the task of discovery of various semantic relationships among them, including synonymy and subsumption. For example, the term *natural language* can be considered, in certain domains at least, to subsume any term denoting a specific human language, such as *English*. Therefore, a query containing the former may be expected to retrieve documents containing the latter. The same can be said about *language* and *English*, unless *language* is in fact a part of the compound term *programming language* in which case the association *language - Fortran* is appropriate. This is a problem because (a) it is a standard practice to include both simple and compound terms in document representation, and (b) term associations have thus far been computed primarily at word level (including fixed phrases) and therefore care must be taken when such associations are used in term matching. This may prove particularly troublesome for systems that attempt term clustering in order to create "meta-terms" to be used in document representation.

The system presented here computes term associations from text at word and fixed phrase level and then uses these associations in query expansion. A fairly primitive filter is employed to separate synonymy and subsumption relationships from others including antonymy and complementation, some of which are strongly domain-dependent. This process has led to an increased retrieval precision in experiments with both ad-hoc and routing queries for TREC-1 and TREC-2 experiments. However, the actual improvement levels can vary substantially between different databases, types of runs (ad-hoc vs. routing), as well as the degree of prior processing of the queries. We continue to study more advanced clustering methods along with the changes in interpretation of resulting associations, as signaled in the previous paragraph. In the remainder of this paper we discuss particulars of the present system and some of the observations made while processing TREC-3 data.

OVERALL DESIGN

Our information retrieval system consists of a traditional statistical backbone (NIST's PRISE system; Harman and Candela, 1989) augmented with various natural language processing components that assist the system in database processing (stemming, indexing, word and phrase clustering, selectional restrictions), and translate a user's information request into an effective query. This design is a careful compromise between purely statistical non-linguistic approaches and those requiring rather accomplished (and expensive) semantic analysis of data, often referred to as 'conceptual retrieval'.

In our system the database text is first processed with a fast syntactic parser. Subsequently certain types of phrases are extracted from the parse trees and used as compound indexing terms in addition to single-word terms. The extracted phrases are statistically analyzed as syntactic contexts in order to discover a variety of similarity links between smaller subphrases and words occurring in them. A further filtering process maps these similarity links onto semantic relations (generalization, specialization, synonymy, etc.) after which they are used to transform a user's request into a search query.

The user's natural language request is also parsed, and all indexing terms occurring in it are identified. Certain highly ambiguous, usually single-word terms may be dropped, provided that they also occur as elements in some compound terms. For example, "natural" is deleted from a query already containing "natural language" because "natural" occurs in many unrelated contexts: "natural number", "natural logarithm", "natural approach", etc. At the same time, other terms may be added, namely those which are linked to some query term through admissible similarity relations. For example, "unlawful activity" is added to a query (TREC topic 055) containing the compound term "illegal activity" via a synonymy link between "illegal" and "unlawful". After the final query is constructed, the database search follows, and a ranked list of documents is returned.

There are several deviations from the above scheme in the system that has been actually used in TREC-3, as well as some important changes from TREC-2. First and foremost, we have 'graduated' from the category B (exploratory systems, about 1/4 of text

data) to the category A (full participation) mostly thanks to significant efficiency improvements in the NLP module. In particular, the BBN's part-of-speech tagger, which we use to preprocess the input before parsing, has been redesigned in time for TREC-3 so that it now adds no more than 5% overhead to the parsing time. We have also installed a new, more efficient version of NIST's PRISE system which cut the indexing time from days to hours. In order to keep memory usage within the limits of our resources, as well as to prepare the system to deal with practically unlimited amounts of data in the future, we devised a randomized index splitting mechanism which creates not one but several balanced sub-indexes. These sub-indexes can be searched independently and the results can be merged meaningfully into a single ranking. Finally, while the query expansion via the domain map is an important part of our system, it has not been used in TREC-3 runs. Our analysis of TREC-2 results revealed several problems with the query expansion scheme and we were in process of redesigning it, however, we were unable to test the revised approach in time for this evaluation, and thus decided to leave it out of TREC-3. We plan to have it in place for TREC-4.

Before we proceed to discuss the particulars of our system we would like to note that all the processing steps, those performed by the backbone system, and those performed by the natural language processing components, are fully automated, and no human intervention or manual encoding is required.

FAST PARSING WITH TTP PARSER

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 TREC-1 report (Strzalkowski, 1993a), as well as in other works (Strzalkowski, 1992; Strzalkowski & Scheyen, 1993).

As may be expected, the skip-and-fit strategy will only be effective if the input skipping can be performed with a degree of determinism. This means that most of the lexical level ambiguity must be removed from the input text, prior to parsing. We achieve this using a stochastic parts of speech tagger to preprocess the text (see TREC-1 report for details).

WORD SUFFIX TRIMMER

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.

[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.

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.

HEAD-MODIFIER STRUCTURES

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. In the TREC experiments reported here we extracted head-modifier word and fixed-phrase pairs only. While TREC databases are large enough to warrant generation of larger compounds, we were unable to verify their effectiveness in indexing, mostly because of the tight schedule.

Let us consider a specific example from the WSJ database:

The former Soviet president has been a local hero ever since a Russian tank invaded Wisconsin.

The tagged sentence is given below, followed by the regularized parse structure generated by TTP, given in Figure 1.

The/*dt* former/*jj* Soviet/*jj* president/*nn* has/*vbz* been/*vbn* a/*dt* local/*jj* hero/*nn* ever/*rb* since/*in* a/*dt* Russian/*jj* tank/*nn* invaded/*vbd* Wisconsin/*np* ./*per*

It should be noted that the parser's output is a predicate-argument structure centered around main elements of various phrases. In Figure 1, BE is the main predicate (modified by HAVE) with 2 arguments (*subject*, *object*) and 2 adjuncts (*adv*, *sub_ord*). INVADE is the predicate in the subordinate clause with 2 arguments (*subject*, *object*). The subject of BE is a noun phrase with PRESIDENT as the head element, two modifiers (FORMER, SOVIET) and a determiner (THE). From this structure, we extract head-modifier pairs that become candidates for compound terms. 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

```
[assert
  [[perf [HAVE]]
    [[verb [BE]]
      [subject
        [np
          [n PRESIDENT]
          [t_pos THE]
          [adj [FORMER]]
          [adj [SOVIET]]]]
      [object
        [np
          [n HERO]
          [t_pos A]
          [adj [LOCAL]]]]
      [adv EVER]
      [sub_ord
        [SINCE
          [[verb [INVADE]]
            [subject
              [np
                [n TANK]
                [t_pos A]
                [adj [RUSSIAN]]]]
            [object
              [np
                [name [WISCONSIN]]]]]]]]]
```

Figure 1. Predicate-argument parse structure.

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. For example, the pair *retrieve+information* will be extracted from any of the following fragments: *information retrieval system; retrieval of information from databases*; and *information that can be retrieved by a user-controlled interactive search process*. In the example at hand, the following head-modifier pairs are extracted (pairs containing low-content elements, such as BE and FORMER, or names, such as WISCONSIN, will be later discarded):

PRESIDENT+BE, PRESIDENT+FORMER, PRESIDENT+SOVIET,
BE+HERO, HERO+LOCAL,
TANK+INVADE, TANK+RUSSIAN, INVADE+WISCONSIN

We may note that 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. The representation of this sentence may therefore contain the following terms (along with their inverted document frequency weights):

PRESIDENT	2.623519
SOVIET	5.416102
PRESIDENT+SOVIET	11.556747
PRESIDENT+FORMER	14.594883
HERO	7.896426
HERO+LOCAL	14.314775
INVADE	8.435012
TANK	6.848128
TANK+INVADE	17.402237
TANK+RUSSIAN	16.030809
RUSSIAN	7.383342
WISCONSIN	7.785689

While generating compound terms we took care to identify 'negative' terms, that is, those whose denotations have been explicitly excluded by negation. Even though matching of negative terms was not used in retrieval (nor did we use negative weights), we could easily prevent matching a negative term in a query against its positive counterpart in the database by removing known negative terms from queries. As an example consider the following fragment from topic 192:

> References to the cost of cleanup and number of people and equipment involved without mentioning the method are not relevant.

The corresponding compound terms are:

> NOT cost cleanup
> NOT number equip
> NOT number people

Note that while this statement is negated, the negation is conditioned with the *without mentioning* ... phrase. Our NLP module is not able to represent such fine distinctions at this time.

NOMINAL COMPOUNDS

The notorious ambiguity of nominal compounds remains a serious difficulty in obtaining head-modifier pairs of highest accuracy. In order to cope with this, 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 disambiguation 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 algorithm proceeds in three steps, as follows:

(1) *Assign scores* to each of the candidate pairs x_i+x_j where $i>j$ from the ambiguous noun phrase $x_1 \cdots x_n$. The score assigned to a candidate pair is the sum of the scores for each occurrence of this pair in any compound nominal within the training corpus. For each occurrence, the score is maximum when the words x_i and x_j are the only words in the phrase, i.e., we have unambiguous nominal $x_j x_i$, in which case the score is 1. For longer phrases, for non-adjacent words, and for pairs anchored at words toward the left of the compound, the score decreases proportionately.

(2) For each set $X_j=\{x_i+x_j \mid \text{for } i>j\}$ of candidate pairs *rank* alternative pairs by their scores.

(3) *Disambiguate* by selecting the top choice from each set such that its score is above an empirically established global threshold, it is significantly higher than the second best choice from the set, and it is not significantly lower than the scores of pairs selected from other sets X_i.

The effectiveness of this algorithm can be measured in terms of recall (the proportion of all valid head+modifier pairs extracted from ambiguous nominals), and precision (the proportion of valid pairs among those extracted). The evaluation was done on a small sample of randomly selected phrases, and the algorithm performance was compared to manually selected correct pairs. The following numbers were recorded: recall 66% to 71%; precision 88% to 91%, depending on the size of the training sample. In terms of the total number of pairs extracted unambiguously from the parsed text (i.e., those obtained by the procedure described in the previous section), the disambiguation step recovers an additional 10% to 15% of pairs, all of which were previously thrown out as unrecoverable. A sample set of ambiguous phrases and extracted head+modifier pairs is shown in Table 1.

Ambiguous nominal	Extracted pairs
oil import fee	oil import import fee
croatian wartime cabinet	croatian cabinet wartime cabinet
national enviromental watchdog group	national group enviromental group watchdog group
current export subsidy program	current program export subsidy subsidy program
gas operating and maintaining expenses	**gas operating operating expenses maintaining expenses

Table 1. Ambiguous nominals and extracted pairs.

EXTRACTING PROPER NAMES

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*). A more advanced recognizer is planned for TREC-4 evaluation. In the TREC-3 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
mcdonnell+douglas+corp+brad+beaver
soviet+georgia
rebel+leader+savimbi
plo+leader+arafat
suzuki+samurai+soft_top+4wd
honda+civic
richard+j+rosebery
mr+rosebery
international+business+machine+corp
cytomegalovirus+retinitis
ids+financial+service+analyst+g+michael+kennedy
senate+judiciary+committee
first+fidelity+bank+n.a+south+jersey
eastern+u.s
federal+national+mortgage+association
canadian+airline+international

TERM CORRELATIONS FROM TEXT

Head-modifier pairs form compound terms used in database indexing. They also serve as occurrence contexts for smaller terms, including single-word terms. If two terms tend to be modified with a number of common modifiers and otherwise appear in few distinct contexts, we assign them a similarity coefficient, a real number between 0 and 1. The similarity is determined by comparing distribution characteristics for both terms within the corpus: how much information content do they carry, do their information contribution over contexts vary greatly, are the common contexts in which these terms occur specific enough? In general we will credit high-content terms appearing in identical contexts, especially

if these contexts are not too commonplace.[2]

To cluster terms into similarity classes, we used a (revised) variant of weighted Jaccard's measure described in (Grefenstette, 1992):

$$SIM(x_1,x_2) = \frac{\sum_{att} MIN(W([x,att]), W([y,att])}{\sum_{att} MAX(W([x,att]), W([y,att])}$$

with

$$W([x,y]) = GEW(x) * log(f_{x,y})$$

$$GEW(x) = 1 + \sum_{y} \left(\frac{\frac{f_{x,y}}{n_y} * \log\left(\frac{f_{x,y}}{n_y} \right)}{log(N)} \right)$$

In the above, $f_{x,y}$ stands for absolute frequency of pair $[x,y]$, n_y is the frequency of term y, and N is the number of single-word terms. Sample clusters obtained from approx. 250 MByte (42 million words) subset of WSJ (years 1990-1992) are given in Table 2.

In order to generate better similarities, we require that words x_1 and x_2 appear in at least *M distinct* common contexts, where a common context is a couple of pairs $[x_1,y]$ and $[x_2,y]$, or $[y,x_1]$ and $[y,x_2]$ such that they each occurred at least three times. Thus, *banana* and *Baltic* will not be considered for similarity relation on the basis of their occurrences in the common context of *republic*, no matter how frequent, unless there is another such common context comparably frequent (there wasn't any in TREC's WSJ database). For smaller or narrow domain databases M=2 is usually sufficient. For large databases covering a rather diverse subject matter, like WSJ, we used $M \geq 5$.[3] This, however, turned out not to be sufficient. We would still generate fairly strong similarity links between terms such as *aerospace* and *pharmaceutical* where 6 and more common contexts were found. In the example at hand the following common contexts were located, all occurring at the head (left) position of a pair (at right are their global entropy weights (GEW) and frequencies with *aerospace* and *pharmaceutical*, respectively):[4]

CONTEXT	GEW	freq/*aerospace*	freq/*pharmaceutical*
firm	0.58	9	22
industry	0.51	84	56
sector	0.61	5	9
concern	0.50	130	115
analyst	0.62	23	8
division	0.53	36	28
giant	0.62	15	12

Note that while some of these weights are quite low (less than 0.6 — GEW takes values between 0 and 1), thus indicating a low importance context, the frequencies with which these contexts occurred with both terms were high and balanced on both sides (e.g., *concern*), thus adding to the strength of association. We are now considering additional thresholds to bar low importance contexts from being used in similarity calculation.

It may be worth pointing out that the similarities are calculated using term co-occurrences in syntactic rather than in document-size contexts, the latter being the usual practice in non-linguistic clustering (e.g., Sparck Jones and Barber, 1971; Crouch, 1988; Lewis and Croft, 1990). Although the two methods of term clustering may be considered mutually complementary in certain situations, we believe that more and stronger associations can be obtained through syntactic-context clustering, given sufficient amount of data and a reasonably accurate syntactic parser.[5]

QUERY EXPANSION

Similarity relations are used to expand user queries with new terms, in an attempt to make the final search query more comprehensive (adding synonyms) and/or more pointed (adding specializations).[6] It follows that not all similarity relations will be equally useful in query expansion, for instance, complementary and antonymous relations like the one between *Australian* and *Canadian*, *accept* and *reject*, or even generalizations like from *aerospace* to *industry* may actually harm system's performance, since we may end up retrieving many

[2] It would not be appropriate to predict similarity between *language* and *logarithm* on the basis of their co-occurrence with *natural*.

[3] For example *banana* and *Dominican* were found to have two common contexts: *republic* and *plant*, although this second occurred in apparently different senses in *Dominican plant* and *banana plant*.

[4] Other common contexts, such as *company* or *market*, have already been rejected because they were paired with too many different words (a high dispersion ratio).

[5] Non-syntactic contexts cross sentence boundaries with no fuss, which is helpful with short, succinct documents (such as CACM abstracts), but less so with longer texts; see also (Grishman et al., 1986).

[6] Query expansion (in the sense considered here, though not quite in the same way) has been used in information retrieval research before (e.g., Sparck Jones and Tait, 1984; Harman, 1988), usually with mixed results. An alternative is to use term clusters to create new terms, "meta-terms", and use them to index the database instead (e.g., Crouch, 1988; Lewis and Croft, 1990). We found that the query expansion approach gives the system more flexibility, for instance, by making room for hypertext-style topic exploration via user feedback.

irrelevant documents. On the other hand, database search is likely to miss relevant documents if we overlook the fact that *vice director* can also be *deputy director*, or that *takeover* can also be *merge*, *buy-out*, or *acquisition*. We noted that an average set of similarities generated from a text corpus contains about as many "good" relations (synonymy, specialization) as "bad" relations (antonymy, complementation, generalization), as seen from the query expansion viewpoint. Therefore any attempt to separate these two classes and to increase the proportion of "good" relations should result in improved retrieval. This has indeed been confirmed in our experiments where a relatively crude filter has visibly increased retrieval precision.

In order to create an appropriate filter, we devised a global term specificity measure (GTS) which is calculated for each term across all contexts in which it occurs. The general philosophy here is that a more specific word/phrase would have a more limited use, i.e., a more specific term would appear in fewer *distinct* contexts. In this respect, GTS is similar to the standard *inverted document frequency (idf)* measure except that term frequency is measured over syntactic units rather than document size units.[7] Terms with higher GTS values are generally considered more specific, but the specificity comparison is only meaningful for terms which are already known to be similar. The new function is calculated according to the following formula:

$$GTS(w) = \begin{cases} IC_L(w) * IC_R(w) & \text{if } \textit{both exist} \\ IC_R(w) & \text{if } \textit{only } IC_R(w) \textit{ exists} \\ IC_L(w) & \textit{otherwise} \end{cases}$$

where (with n_w, $d_w > 0$):

$$IC_L(w) = IC([w,\_]) = \frac{n_w}{d_w(n_w+d_w-1)}$$
$$IC_R(w) = IC([\_,w]) = \frac{n_w}{d_w(n_w+d_w-1)}$$

For any two terms w_1 and w_2, and a constant $\delta > 1$, if $GTS(w_2) \geq \delta * GTS(w_1)$ then w_2 is considered more specific than w_1. In addition, if $SIM_{norm}(w_1,w_2) = \sigma > \theta$, where θ is an empirically established threshold, then w_2 can be added to the query containing term w_1 with weight σ.[8] For example, the following were obtained from the WSJ training database:

$GTS(takeover)$ = 0.00145576
$GTS(merge)$ = 0.00094518
$GTS(buy\text{-}out)$ = 0.00272580
$GTS(acquire)$ = 0.00057906

with

$SIM(takeover, merge)$ = 0.190444
$SIM(takeover, buy\text{-}out)$ = 0.157410
$SIM(takeover, acquire)$ = 0.139497
$SIM(merge, buy\text{-}out)$ = 0.133800
$SIM(merge, acquire)$ = 0.263772
$SIM(buy\text{-}out, acquire)$ = 0.109106

Therefore both *takeover* and *buy-out* can be used to specialize *merge* or *acquire*. With this filter, the relationships between *takeover* and *buy-out* and between *merge* and *acquire* are either both discarded or accepted as synonymous. At this time we are unable to tell synonymous or near synonymous relationships from those which are primarily complementary, e.g., *man* and *woman*.

Query expansion is an important part of our system, but it wasn't used in TREC-3, mostly because we were in process of redesigning it. Following TREC-2 we found various problems with both the term clustering procedure as well as with the way the cluster were used to add new terms to queries. Details of these finding are discussed in TREC-2 final report.

CREATING AN INDEX

The limited amount of resources that we had available for indexing forced us to devise a method that splits the collection randomly and produces several sub-indexes. This method would allow us now to index even larger collections in reasonable times. The preliminary tests that we carried out in order to compare the performance of systems where the collection is split into N sub-indexes, for different values of N, suggest that a collection can be split into at least 7 sub-indexes without seeing any degradation in the performance. Given the results that we obtained from such tests as well as the fact that the tests were carried out using relatively small collections (about 150 Megabytes) we intend to perform more extensive testing as soon as possible.

One of the problems we had to face for TREC-1 and TREC-2 was that we did not have enough real memory to index the complete collection (category A) in a reasonable time . Even indexing only the collection for category B (550 megabytes for the ad-hoc experiments) used to take 2 weeks, or about 330 hours. This was more slow than the times that could be obtained by other versions of the PRISE system that were already available by that time. We used a slower version because we did not have then enough main memory to use the faster one. The faster version grows the word frequency tree in main

[7] We believe that measuring term specificity over document-size contexts (e.g., Sparck Jones, 1972) may not be appropriate in this case. In particular, syntax-based contexts allow for processing texts without any internal document structure.

[8] For TREC-2 we used $\sigma = 0.2$; δ varied between 10 and 100.

word	cluster
takeover	*merge, buy-out, acquire, bid*
benefit	*compensate, aid, expense*
capital	*cash, fund, money*
staff	*personnel, employee, force*
attract	*lure, draw, woo*
sensitive	*crucial, difficult, critical*
speculate	*rumor, uncertainty, tension*
president	*director, executive, chairman*
vice	*deputy*
outlook	*forecast, prospect, trend*
law	*rule, policy, legislate, bill*
earnings	*profit, revenue, income*
portfolio	*asset, invest, loan*
inflate	*growth, demand, earnings*
industry	*business, company, market*
growth	*increase, rise, gain*
firm	*bank, concern, group, unit*
environ	*climate, condition, situation*
debt	*loan, secure, bond*
lawyer	*attorney*
counsel	*attorney, administrator, secretary*
compute	*machine, software, equipment*
competitor	*rival, competition, buyer*
alliance	*partnership, venture, consortium*
big	*large, major, huge, significant*
fight	*battle, attack, war, challenge*
base	*facile, source, reserve, support*
shareholder	*creditor, customer, client investor, stockholder*

Table 2. Selected clusters obtained from syntactic contexts, derived from approx. 40 million words of WSJ text, with weighted Tanimoto formula.

memory, and it is the physical memory that matters here, not the virtual memory, since a tree larger than the size of the real memory causes so many page faults that performance becomes unacceptably slow.

The version of the PRISE system that we used for TREC-3 is much faster than previous versions. According to the on-line documentation provided by NIST the old system would take about 67 hours to index 276 Megabytes of WSJ material while the new system takes less than 2 hours to index the same material. Still, we did not have enough main memory to use the new system to index the complete collection. Our solution to this problem was to split the collection into N sets of almost equal number of documents and create a separate sub-index for each set. In order to keep the N sub-indexes balanced with respect to each other (so that the term idfs are comparable across sub-indexes, for example) we split the collection randomly into N sets. This is done by assigning each document to one of the N sets selected at random. Our goal was to build N sets that would be as homogeneous as possible. At retrieval time the same query is submitted to each one of the sub-indexes and a separate list of ranked documents is obtained for each index. Since we expect idfs to be comparable across sub-indexes, it makes sense to compare the scores of documents belonging to different sub-indexes. The result of the query is then the set of documents with the highest scores chosen from the union of all lists of ranked documents.

In order to evaluate this technique we ran a series of experiments involving about 50000 records. We split that collection into N sets for several values of N (from 1 to 7) and made some measurements of parameters that we expected to be indicators of the degree of homogeneity (e.g., standard deviation of the total number of terms per index, standard deviation of the maximum idf, standard deviation of the number of unique terms, and others). As expected, these indicators showed a decreasing level of homogeneity as N grows larger. This information is summarized in Table 3.

For each value of N, we evaluated the performance of the system using a series of queries for which NIST had provided relevance judgments. For the weighting scheme we were using, and the small collection used for these preliminary experiments, we observed that the performance actually peaks at N = 4 (the average precision when N was 4 was about 7% better than when N was 1). We thought that these results were promising enough to justify the use of the technique described in order to index the complete collection but we intend to perform a much more careful and complete series of experiments as soon as the time and the resources are available. Table 4 summarizes the system's performance at various levels of index split with a subset of AP subcollection.

No. of indexes	Max-Mem MB	Max-idf %std	Uniq.terms Mean	Uniq.terms %std
1	81.9	0.000	921253	0.000
2	61.2	0.424	600869	1.006
3	54.7	0.902	438992	11.678
4	48.2	0.555	373249	3.095
5	46.0	0.986	314986	6.356
6	44.1	1.080	279261	7.318
7	46.8	2.432	247606	16.475

Table 3. Statistics of index splitting performed on a subset of Tipster AP88 subcollection consisting of 48,770 records (about 230 MBytes).

No. of indexes	Avg Prec %change	R-Prec %change	Recall %change
1	0.00	0.00	0.00
2	+4.04	+1.85	+1.11
3	+4.63	+0.72	+0.81
4	+7.04	+4.53	+2.59
5	+1.68	+4.08	+3.92
6	+5.68	+2.75	+4.29
7	+4.18	+4.45	+4.36

Table 4. Performance statistics for split index performed on a subset of Tipster AP88 subcollection consisting of 48,770 records (about 230 MBytes).

For TREC-3 we used 4 sub-indexes for the ad-hoc experiments (2200 Megabytes) and 2 for the routing part (1100 Megabytes). We chose these numbers because, in each case, it was the smallest number of sub-indexes that we could handle given our resources. A nice side-effect of this technique is that each index can be created in parallel on a different machine, making the total time required even shorter. The parameters of the 4-way split used in indexing the TREC-3 ad-hoc database are listed in Table 5.

TERM WEIGHTING ISSUES

Finding a proper term weighting scheme is critical in term-based retrieval since the rank of a document is determined by the weights of the terms it shares with the query. One popular term weighting scheme, known as tf.idf, weights terms proportionately to their inverted document frequency scores and to their in-document

Index No.	Postings MB	Dict. MB	Max-idf	Records	Uniq.terms
1	128.82	72.91	18.509	186557	2928737
2	129.34	72.26	18.492	184460	2909249
3	128.99	72.82	18.499	185297	2931791
4	128.27	71.26	18.498	185114	2874007

Table 5. Statistics of the 4-way split index created for ad-hoc database from Tipster Disks 1 and 2 (about 2 GBytes).

frequencies (tf). The in-document frequency factor is usually normalized by the document length, that is, it is more significant for a term to occur 5 times in a short 20-word document, than to occur 10 times in a 1000-word article.[9]

In our official TREC runs we used the normalized tf.idf weights for all terms alike: single 'ordinary-word' terms, proper names, as well as phrasal terms consisting of 2 or more words. Whenever phrases were included in the term set of a document, the length of this document was increased accordingly. This had the effect of decreasing tf factors for 'regular' single word terms.

A standard tf.idf weighting scheme (and we suspect any other uniform scheme based on frequencies) is inappropriate for mixed term sets (ordinary concepts, proper names, phrases) because:

(1) It favors terms that occur fairly frequently in a document, which supports only general-type queries (e.g., "all you know about 'star wars'"). Such queries are not typical in TREC.

(2) It attaches low weights to infrequent, highly specific terms, such as names and phrases, whose only occurrences in a document often decide of relevance. Note that such terms cannot be reliably distinguished using their distribution in the database as the sole factor, and therefore syntactic and lexical information is required.

(3) It does not address the problem of inter-term dependencies arising when phrasal terms and their component single-word terms are all included in a document representation, i.e., *launch+satellite* and *satellite* are not independent, and it is unclear whether they should be counted as two terms.

[9] This is not always true, for example when all occurrences of a term are concentrated in a single section or a paragraph rather than spread around the article. See the following section for more discussion.

In our post-TREC-2 experiments we considered (1) and (2) only. We changed the weighting scheme so that the phrases (but not the names which we did not distinguish in TREC-2) were more heavily weighted by their idf scores while the in-document frequency scores were replaced by logarithms multiplied by sufficiently large constants. In addition, the top N highest-idf matching terms (simple or compound) were counted more toward the document score than the remaining terms. This 'hot-spot' retrieval option is discussed in the next section.

Schematically, these new weights for phrasal and highly specific terms are obtained using the following formula, while weights for most of the single-word terms remain unchanged:

$$weight(T_i)=(C_1*log(tf)+C_2*\alpha(N,i))*idf$$

In the above, $\alpha(N,i)$ is 1 for $i<N$ and is 0 otherwise. The selection of a weighting formula was partly constrained by the fact that document-length-normalized *tf* weights were precomputed at the indexing stage and could not be altered without re-indexing of the entire database. The intuitive interpretation of the $\alpha(N,i)$ factor is given in the following section.

The table below illustrates the problem of weighting phrasal terms using topic 101 and a relevant document (WSJ870226-0091).

Topic 101 matches WSJ870226-0091
duplicate terms not shown

TERM	*TF.IDF*	*NEW WEIGHT*
sdi	1750	1750
eris	3175	3175
star	1072	1072
wars	1670	1670
laser	1456	1456
weapon	1639	1639
missile	872	872
space+base	2641	2105
interceptor	2075	2075
exoatmospheric	1879	3480
system+defense	2846	2219
reentry+vehicle	1879	3480
initiative+defense	1646	2032
system+interceptor	2526	3118
DOC RANK	30	10

Changing the weighting scheme for compound terms, along with other minor improvements (such as expanding the stopword list for topics, or correcting a few parsing bugs) has lead to the overall increase of precision of nearly 20% over our official TREC-2 ad-hoc results. This weighting scheme was again used in TREC-3 runs.

'HOT SPOT' RETRIEVAL

Another difficulty with frequency-based term weighting arises when a long document needs to be retrieved on the basis of a few short relevant passages. If the bulk of the document is not directly relevant to the query, then there is a strong possibility that the document will score low in the final ranking, despite some strongly relevant material in it. This problem can be dealt with by subdividing long documents at paragraph breaks, or into approximately equal length fragments and indexing the database with respect to these (e.g., Kwok 1993). While such approaches are effective, they also tend to be costly because of increased index size and more complicated access methods.

Efficiency considerations has led us to investigate an alternative approach to the *hot spot* retrieval which would not require re-indexing of the existing database or any changes in document access. In our approach, the maximum number of terms on which a query is permitted to match a document is limited to N highest weight terms, where N can be the same for all queries of may vary from one query to another. Note that this is not the same as simply taking the N top terms from each query. Rather, for each document for which there are M matching terms with the query, only min(M,N) of them, namely those which have highest weights, will be considered when computing the document score. Moreover, only the global importance weights for terms are considered (such as idf), while local in-document frequency (eg., tf) is suppressed by either taking a log or replacing it with a constant. The effect of this 'hot spot' retrieval is shown below in the ranking of relevant documents within the top 1000 retrieved documents for topic 65:

Log(tf).idf retrieval

DOCUMENT ID	RANK	SCORE
WSJ870304-0091	4	12228
WSJ891017-0156	7	9771
WSJ920226-0034	14	8921
WSJ870429-0078	26	7570
WSJ870205-0078	33	6972
WSJ880712-0033	34	6834
WSJ920116-0002	37	6580
WSJ910328-0013	74	4872
WSJ910830-0140	80	4701
WSJ890804-0138	102	4134
WSJ911212-0022	104	4065
WSJ870825-0026	113	3922
WSJ880712-0023	135	3654
WSJ871202-0145	153	3519

Hot-spot idf-dominated with N=20

DOCUMENT ID	RANK	SCORE
WSJ920226-0034	1	11955
WSJ870304-0091	3	11565
WSJ870429-0078	5	9997
WSJ920116-0002	7	9997
WSJ910830-0140	11	8792
WSJ870205-0078	20	8402
WSJ910328-0013	29	8402
WSJ880712-0033	71	6834
WSJ880712-0023	72	6834
WSJ891017-0156	87	6834
WSJ890804-0138	92	6834
WSJ911212-0022	111	6834
WSJ871202-0145	124	6834

The final ranking is obtained by merging the two rankings by score. While some of the recall may be sacrificed ('hot spot' retrieval has, understandably, lower recall than full query retrieval, and this becomes the lower bound on recall for the combined ranking) the combined ranking precision has been consistently better than in either of the original rankings: an average improvement is 10-12% above the tf.idf run precision (which is often stronger of the two). The 'hot spot' weighting is represented with the α factor in the term weighting formula given in the previous section.

SUMMARY OF RESULTS

We have processed the total of 3334 MBytes of text during TREC-3. The first 2162 MBytes were data from the Tipster/TREC disks 1 and 2 of which 550 Mbytes (Wall Street Journal subcollection) were previously processed for TREC-2; however, even this portion had to be partially reprocessed. The entire process (tagging, parsing, phrase and name extraction) took about 45 minutes per Megabyte, or just over 2 months on a Sun's SparcStations 10 (at times using an additional Sparc-2). Building a 4-way split index took about 0.6 minutes per Megabyte, or about 21 hours on the Sparc10. The final index size, including postings files and term dictionaries was 804 MBytes, and included approximately 2.9 million unique terms in each sub-index (that's single-word terms, syntactic word pairs and compound names) or nearly 16 (unique) terms per document.

The remaining 1172 MBytes were documents from the Tipster/TREC disk 3 of which about 300 Mbytes of San Jose Mercury articles were previously processed for TREC-2. This portion of the corpus was used to create the routing database. Natural language processing of this part required about 4 weeks on SparcStation 10, and about 10 hours of indexing time. The final size of the index was 428 MBytes, split into 2 sub-indexes of about 214 MBytes each. Each sub-index contained about 3.2 million unique terms, or more than 19 unique terms per record.

Note that in both cases the index size was at 37% of the initial size of the corpus. Given that the natural language processing has added an average 30% to the size of the input (i.e., for each Megabyte of text we obtained about 1.3 Megabytes of terms), the indexer compression ratio was actually 29%.[10]

Two types of retrieval have been done: (1) new topics 151-200 were run in the ad-hoc mode against the Disk-1&2 database, and (2) topics 101-150, previously used in TREC-2, were run in the routing mode against the Disk-3 database. In each category 2 official runs were performed, all fully automatic, with different set up of system's parameters. These runs were labeled *nyuir1* and *nyuir2*. The second run in the routing category includes an experimental use an automatic feedback program which uses the known relevance judgements for topics 101-150 with respect TREC-2 database, to automatically expand the search queries. Summary statistics for these runs are shown in Tables 6 and 7. We note that there is a significant (20%) improvement in precision, as well as a visible increase in recall over the base statistical run when phrasal terms are used. The increase is smaller for routing runs because the routing queries already contained manually prepared concepts fields (<*con*>). We also note the robust improvement of routing results when massive query expansion is performed based on the known relevance judgements for these queries with respect to the training database.

An example ad-hoc topic is shown below:

<top>
<num> Number: 189

<title> Topic: Real Motives for Murder

<desc> Description:
Document must identify a murderer's motive for killing a person or persons in a true case.

<narr> Narrative:
Most relevant would be a description of an intentional murder with a statement of the murderer's motive. An unintentional murder, such as in a charge of second-degree homicide, would be relevant if a motive is stated for an action which clearly led to the victim's death.
</top>

[10] This may be somewhat misleading, since many of the compound terms added by NLP were singletons which take little index space. The unprocessed text compression ratio may in fact be closer to 37%.

The reader familiar with previous TREC evaluations may notice that this query lacks the manually derived <con> field which listed important concepts relevant to the topic, some of which had not even occurred in the actual query. Performance comparison in database search performed during TREC-2 showed that search queries built from the concepts fields outperformed the queries based on narrative sections of the topics by as much as 25% to 30% in precision and up to 10% in recall. Nonetheless, it was felt that the results obtained with the use of the <con> field in the queries did not reflect accurately the capabilities of automatic IR systems in dealing with unprocessed input, therefore the field was dropped in TREC-3.

The table below shows the search query obtained from Topic 189 above with respect to one of the 4 sub-indexes making up the ad-hoc database. Note that the terms extracted from <desc> field are weighted doubly.

Query 189

term = motive+murder	idf = 18.509256	weight = 2
term = motive+murder	idf = 18.509256	weight = 1
term = murder+intentional	idf = 18.509256	weight = 1
term = state+motive	idf = 17.509256	weight = 1
term = death+victim	idf = 15.509257	weight = 1
term = motive+real	idf = 15.509257	weight = 1
term = motive+real	idf = 15.509257	weight = 1
term = charge+homicide	idf = 14.339332	weight = 1
term = unintentional	idf = 12.727898	weight = 1
term = kill+person	idf = 12.033524	weight = 2
term = kill+person	idf = 12.033524	weight = 1
term = second+degree	idf = 11.299804	weight = 1
term = homicide	idf = 10.531977	weight = 1
term = intentional	idf = 9.724622	weight = 1
term = motive	idf = 8.484118	weight = 1
term = motive	idf = 8.484118	weight = 1
term = motive	idf = 8.484118	weight = 1
term = murder	idf = 7.294331	weight = 2
term = murder	idf = 7.294331	weight = 1
term = murder	idf = 7.294331	weight = 1
term = murder	idf = 7.294331	weight = 1
term = victim	idf = 6.775394	weight = 1
term = true	idf = 6.400079	weight = 1
term = degree	idf = 5.974225	weight = 1
term = death	idf = 5.846589	weight = 1

Note that many ‘function’ words have been removed from the query, e.g., *must*, *identify*, as well as other ‘common words’ such as *document* and *relevant* (this is in addition to our regular list of ‘stopwords’). Some still remain, however, e.g., *true* and *degree*, because these could not be uniformly considered as ‘common’ across all queries.

Run Name	base ad-hoc	nyuir1 ad-hoc	nyuir2 ad-hoc
Queries	50	50	50
Tot number of docs over all queries			
Ret	50000	50000	50000
Rel	9805	9805	9805
RelRet	5398	5978	5978
%chg		+11.0	+11.0
Recall			
0.00	0.6710	0.7653	0.7639
0.10	0.4444	0.5420	0.5429
0.20	0.3784	0.4465	0.4523
0.30	0.3298	0.3763	0.3814
0.40	0.2821	0.3271	0.3281
0.50	0.2274	0.2608	0.2613
0.60	0.1684	0.2031	0.2033
0.70	0.1112	0.1522	0.1519
0.80	0.0699	0.0990	0.0989
0.90	0.0147	0.0339	0.0332
1.00	0.0000	0.0029	0.0029
Average precision over all rel docs			
Avg	0.2271	0.2722	0.2735
%chg		+20.0	+20.0
Precision at			
5 docs	0.5160	0.5960	0.5880
10 docs	0.4680	0.5480	0.5580
15 docs	0.4427	0.5280	0.5253
20 docs	0.4280	0.5060	0.5070
30 docs	0.4113	0.4793	0.4827
100 docs	0.3138	0.3650	0.3644
200 docs	0.2489	0.2902	0.2907
500 docs	0.1623	0.1832	0.1832
1000 docs	0.1080	0.1196	0.1196
R-Precision (after RelRet)			
Exact	0.2807	0.3232	0.3231
%chg		+15.0	+15.0

Table 6. Automatic ad-hoc run statistics for queries 151-200 against Tipster Disks-1&2 database: (1) *base* - statistical terms only; (2) *nyuir1* - using syntactic phrases, names, and the new weighting scheme; (3) *nyuir2* - same as 2 but different parameters on the weighting scheme.

Run Name	base routing	nyuir1 routing	nyuir2 routing	nyuir2a routing
Queries	50	50	50	50
Tot number of docs over all queries				
Ret	50000	50000	50000	5000
Rel	9353	9353	9353	9353
RelRet	6011	6350	7203	7345
%chg		+5.6	+19.8	+21.2
Recall	(interp) Precision Averages			
0.00	0.7551	0.7401	0.7711	0.7881
0.10	0.4871	0.5003	0.5957	0.6243
0.20	0.4059	0.4255	0.4857	0.5117
0.30	0.3482	0.3719	0.4337	0.4492
0.40	0.2952	0.3330	0.3758	0.4005
0.50	0.2557	0.2723	0.3321	0.3531
0.60	0.2116	0.2327	0.2762	0.2922
0.70	0.1582	0.1765	0.2213	0.2411
0.80	0.0953	0.1094	0.1480	0.1541
0.90	0.0651	0.0691	0.0816	0.0918
1.00	0.0048	0.0077	0.0069	0.0117
Average precision over all rel docs				
Avg	0.2578	0.2743	0.3244	0.3422
%chg		+6.4	+25.8	+32.7
Precision at				
5 docs	0.5080	0.5280	0.6000	0.6080
10 docs	0.4520	0.4800	0.5560	0.5760
15 docs	0.4533	0.4600	0.5333	0.5560
20 docs	0.4410	0.4390	0.5200	0.5370
30 docs	0.4273	0.4273	0.4940	0.5133
100 docs	0.3418	0.3584	0.4098	0.4270
200 docs	0.2838	0.3063	0.3380	0.3484
500 docs	0.1874	0.2008	0.2260	0.2310
1000 docs	0.1202	0.1270	0.1441	0.1469
R-Precision (after Rel)				
Exact	0.3062	0.3135	0.3510	0.3635
%chg		+2.4	+14.6	+18.7

Table 7. Automatic routing run statistics for queries 101-150 against Tipster Disk-3 database: (1) *base* - statistical terms only; (2) *nyuir1* - using syntactic phrases, names, and the new weighting scheme; (3) *nyuir2* - same as 2 with the automatic relevance feedback, but some queries not fully expanded due to an error; (4) *nyuir2a* - nyuir2 rerun after TREC-3 with full feedback.

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. We suggest, with some caution until more experiments are run, that natural language processing can be very effective in creating appropriate search queries out of user's initial specifications which can be frequently imprecise or vague.

At the same time it is important to keep in mind that the NLP techniques that meet our performance requirements (or at least are believed to be approaching these requirements) are still fairly unsophisticated in their ability to handle natural language text. In particular, advanced processing involving conceptual structuring, logical forms, etc., is still beyond reach, computationally. It may be assumed that these advanced techniques will prove even more effective, since they address the problem of representation-level limits; however the experimental evidence is sparse and necessarily limited to rather small scale tests.

ACKNOWLEDGEMENTS

We would like to thank Donna Harman of NIST for making her PRISE system available to us. Will Rogers provided valuable assistance in installing updated versions of PRISE at NYU. We would also like to thank Ralph Weischedel and Constantine Papageorgiou of BBN for providing and assisting in the use of the part of speech tagger. This paper is based upon work supported by the Advanced Research Projects Agency under Contract N00014-90-J-1851 from the Office of Naval Research, under ARPA's Tipster Phase-2 Contract 94-FI57900-000, and the National Science Foundation under Grant IRI-93-02615.

REFERENCES

Broglio, John and W. Bruce Croft. 1993. ''Query Processing for Retrieval from Large Text Bases.'' Proceedings of ARPA HLT Workshop, March 21-24, Plainsboro, NJ.

Church, Kenneth Ward and Hanks, Patrick. 1990. ''Word association norms, mutual information, and lexicography.'' *Computational Linguistics*, 16(1), MIT Press, pp. 22-29.

Crouch, Carolyn J. 1988. ‘‘A cluster-based approach to thesaurus construction.’’ Proceedings of ACM SIGIR-88, pp. 309-320.

Grefenstette, Gregory. 1992. ‘‘Use of Syntactic Context To Produce Term Association Lists for Text Retrieval.’’ Proceedings of SIGIR-92, Copenhagen, Denmark. pp. 89-97.

Grishman, Ralph, Lynette Hirschman, and Ngo T. Nhan. 1986. ‘‘Discovery procedures for sublanguage selectional patterns: initial experiments’’. *Computational Linguistics*, 12(3), pp. 205-215.

Grishman, Ralph and Tomek Strzalkowski. 1991. ‘‘Information Retrieval and Natural Language Processing.’’ Position paper at the workshop on Future Directions in Natural Language Processing in Information Retrieval, Chicago.

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

Harman, Donna and Gerald Candela. 1989. ‘‘Retrieving Records from a Gigabyte of text on a Minicomputer Using Statistical Ranking.’’ *Journal of the American Society for Information Science*, 41(8), pp. 581-589.

Hindle, Donald. 1990. ‘‘Noun classification from predicate-argument structures.’’ Proc. 28 Meeting of the ACL, Pittsburgh, PA, pp. 268-275.

Kwok, K.L., L. Papadopoulos and Kathy Y.Y. Kwan. 1993. ‘‘Retrieval Experiments with a Large Collection using PIRCS.’’ Proceedings of TREC-1 conference, NIST special publication 500-207, pp. 153-172.

Lewis, David D. and W. Bruce Croft. 1990. ‘‘Term Clustering of Syntactic Phrases’’. Proceedings of ACM SIGIR-90, pp. 385-405.

Meteer, Marie, Richard Schwartz, and Ralph Weischedel. 1991. "Studies in Part of Speech Labeling." Proceedings of the 4th DARPA Speech and Natural Language Workshop, Morgan-Kaufman, San Mateo, CA. pp. 331-336.

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

Sparck Jones, Karen. 1972. ‘‘Statistical interpretation of term specificity and its application in retrieval.’’ *Journal of Documentation*, 28(1), pp. 11-20.

Sparck Jones, K. and E. O. Barber. 1971. ‘‘What makes automatic keyword classification effective?’’ *Journal of the American Society for Information Science*, May-June, pp. 166-175.

Sparck Jones, K. and J. I. Tait. 1984. ‘‘Automatic search term variant generation.’’ *Journal of Documentation*, 40(1), pp. 50-66.

Strzalkowski, Tomek and Barbara Vauthey. 1991. ‘‘Fast Text Processing for Information Retrieval.’’ Proceedings of the 4th DARPA Speech and Natural Language Workshop, Morgan-Kaufman, pp. 346-351.

Strzalkowski, Tomek and Barbara Vauthey. 1992. ‘‘Information Retrieval Using Robust Natural Language Processing.’’ Proc. of the 30th ACL Meeting, Newark, DE, June-July. pp. 104-111.

Strzalkowski, Tomek. 1992. ‘‘TTP: A Fast and Robust Parser for Natural Language.’’ Proceedings of the 14th International Conference on Computational Linguistics (COLING), Nantes, France, July 1992. pp. 198-204.

Strzalkowski, Tomek. 1993. ‘‘Natural Language Processing in Large-Scale Text Retrieval Tasks.’’ Proceedings of the First Text REtrieval Conference (TREC-1), NIST Special Publication 500-207, pp. 173-187.

Strzalkowski, Tomek. 1993. ‘‘Robust Text Processing in Automated Information Retrieval.’’ Proc. of ACL-sponsored workshop on Very Large Corpora. Ohio State Univ. Columbus, June 22.

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

Strzalkowski, Tomek. 1994. ‘‘Natural Language Information Retrieval’’ To appear in **Information Processing and Management**.

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.