RECENT DEVELOPMENTS IN NATURAL LANGUAGE TEXT RETRIEVAL

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

ABSTRACT

This paper reports on some recent developments in our natural language text retrieval system. The system uses advanced natural language processing techniques to enhance the effectiveness of term-based document retrieval. The backbone of our system is a traditional statistical engine which builds inverted index files from pre-processed documents, and then 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-2 effort, the total of 550 MBytes of Wall Street Journal articles (ad-hoc queries database) and 300 MBytes of San Jose Mercury articles (routing data) have been processed. In terms of text quantity this represents approximately 130 million words of English. Unlike in TREC-1, we were able to create a single compound index for each database, and therefore avoid merging of results. While the general design of the system has not changed since TREC-1 conference, we nonetheless replaced several components and added a number of new features which are described in the present paper.

INTRODUCTION

A typical 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 a preprocessed user query (or 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.

The simplest word-based representations of content 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 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 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-2 data. The above comments will provide the background for situating our present effort and the state-of-the-art with respect to where we should be in the future.

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".

One of the striking observations made during the course of TREC-2 was to note that removing low-quality terms from the queries is at least as important (and often more so) as adding synonyms and specializations. In some instances (e.g., routing runs) low-quality terms had to be removed (or inhibited) *before* similar terms could be added to the query or else the effect of query expansion was all but drowned out by the increased noise.[1]

After the final query is constructed, the database search follows, and a ranked list of documents is returned. It should be noted 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

[1] We would like to thank Donna Harman for turning our attention to the importance of term weighting schemes, including term deletion.

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. In the runs with approximately 130 million words of TREC's Wall Street Journal and San Jose Mercury texts,[2] the parser's speed averaged between 0.3 and 0.5 seconds per sentence, or up to 70 words per second, on a Sun's SparcStation2. 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). For TREC-2 a number of problems have been corrected in the tagger, including improper tokenization of input and handling of abbreviations.

[2] Approximately 0.85 GBytes of text, over 6 million sentences.

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. [3] 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.

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 in no position to verify their effectiveness in indexing. This was largely because of the tight schedule, but also because of rapidly escalating complexity of the indexing process: even with 2-word phrases, compound terms accounted for nearly 88% of all index entries, in other words, including 2-word phrases increased the index size approximately 8 times.

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*

[3] 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.

```
[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.

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 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 067:

> It should NOT be about economically-motivated civil disturbances and NOT be about a civil disturbance directed against a second country.

The corresponding compound terms are:

NOT disturb+civil
NOT country+second
NOT direct+disturb

The particular way of interpreting syntactic contexts was dictated, to some degree at least, by statistical considerations. Our initial experiments were performed on a relatively small collection (CACM-3204), and therefore we combined pairs obtained from different syntactic relations (e.g., verb-object, subject-verb, noun-adjunct, etc.) in order to increase frequencies of some associations. This became largely unnecessary in a large collection such as TIPSTER, but we had no means to test alternative options, and thus decided to stay with the original. It should not be difficult to see that this was a compromise solution, since many important distinctions were potentially lost, and strong associations could be produced where there weren't any. A way to improve things is to consider different syntactic relations independently, perhaps as independent sources of evidence that could lend support (or not) to certain term similarity predictions. We have started investigating this option during TREC-2, however, it has not been sufficiently tested yet.

One difficulty in obtaining head-modifier pairs of highest accuracy is the notorious ambiguity of nominal compounds. For example, the phrase *natural language processing* should generate *language+natural* and *processing+language*, while *dynamic information processing* is expected to yield *processing+dynamic* and *processing+information*. A still another case is *executive vice president* where the association *president+executive* may be stretching things a bit too far. Since our parser has no knowledge about the text domain, and uses no semantic preferences, it does not attempt to guess any internal associations within such phrases. Instead, this task is passed to the pair extractor module which processes ambiguous parse structures in two phases. In phase one, all and only unambiguous head-modifier pairs are extracted, and the frequencies of their occurrences are recorded. In phase two, frequency information about pairs generated in the first pass is used to form associations from ambiguous structures. For example, if *language+natural* has occurred unambiguously a number of times in contexts such as *parser for natural language*, while *processing+natural* has occurred significantly fewer times or perhaps none at all, then we will prefer the former association as valid. In TREC-2 phrase disambiguation was not used, instead we decided to avoid ambiguous phrases altogether. While our *disambig* program worked generally satisfactorily, we could resolve only a small fraction of cases (about 7%) and thus it's impact on the overall system's performance was limited. However, query-level disambiguation may be more important.

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.[4]

For TREC-2 runs we used a similarity formula which, unlike the similarity formula used in TREC-1, produces clusters of related words and phrases, but will not generate uniform term similarity ranking across clusters. This new formula, however, appeared better suited to handle the diverse subject matter of the WSJ database. We used a (revised) variant of weighted Tanimoto'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 1.

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 or SJMN (San Jose Mercury News), we used $M \geq 5$.[5] 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 and frequencies with *aerospace* and

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

[5] 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*.

pharmaceutical, respectively):[6]

firm	GEW=0.58	fx1y=9	fx2y=22
industry	GEW=0.51	fx1y=84	fx2y=56
sector	GEW=0.61	fx1y=5	fx2y=9
concern	GEW=0.50	fx1y=130	fx2y=115
analyst	GEW=0.62	fx1y=23	fx2y=8
division	GEW=0.53	fx1y=36	fx2y=28
giant	GEW=0.62	fx1y=15	fx2y=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.[7]

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).[8] 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 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.[9] 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.[10] For example, the following were obtained

[6] 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, see note 12).

[7] 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).

[8] 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, "metaterms", 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.

[9] 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.

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

from the WSJ training database:

$$
\begin{aligned}
GTS(takeover) &= 0.00145576\\
GTS(merge) &= 0.00094518\\
GTS(buy-out) &= 0.00272580\\
GTS(acquire) &= 0.00057906
\end{aligned}
$$

with

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

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*.

In TREC-1 the impact of query expansion through term similarities on the system's overall performance was generally disappointing. For TREC-2 we have made a number of changes to the term correlation model, but again time limitations prevented us from properly testing all options. Among the most important changes are:

(1) Exclusion of pairs obtained from SUBJECT--VERB relations: we determined that these contexts are generally of little use as neither subject nor verb subcategorizes well for the other. Moreover we observed that the presence of these pairs was the source of many unwanted term associations.[11]

(2) Automatic pruning of low-content terms from the queries: terms with low idf weights, terms with low information contribution weights that are elements of compound terms, are removed from queries before database search. As we tuned various cutoff thresholds we noted that a significant increase in both recall and precision could be obtained.[12]

[11] Subject-Verb pairs were retained as compound terms, however.

[12] The Information Contribution measure indicates the strength of word pairings, and is defined as $IC(x, [x,y]) = \frac{f_{x,y}}{n_x + d_x - 1}$ where $f_{x,y}$ is the absolute frequency of pair [x,y] in the corpus, n_x is the frequency of term x at the head position, and d_x is a dispersion parameter understood as the number of distinct syntactic contexts in which term x is found.

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

CREATING AN INDEX

Most of the problems we encountered when adapting NIST's PRISE system for use in TREC-2 had to do with the size of the data that had to be indexed.

We had to deal with the restrictions imposed by the resources we had (e.g., only 96 MBytes of virtual memory). The rest of this paragraph signals some of the changes we made to the NIST system in order to deal with our restrictions. The original system would request twice the previously requested amount of memory each time it needed more. As a result of this the system would reach the limit of virtual memory after only a relatively small portion of the total number of documents had been indexed. In our version, the memory requested by the system grows linearly. The increments are estimated in such a way that the system never requests too much memory.

The indexing process became too fragile when the limits of the environment were approached. When a large portion of the virtual memory and of the disk space was being used by the indexing process, crashes became very likely. Unfortunately, it turned out that the process was very difficult to restart after some crashes (e.g., in the *rebuild* phase), thus leading to time consuming repeats.

Indexing also takes too long at present. Given the size of the data to be indexed the whole process takes at least 250 hours if everything goes well, which happens seldom. Given TREC-2's deadlines we could not afford to perform too many experiments: we barely had time to index the corpus once.

Most of the previous problems could be solved by distributing the indexing process to several different machines and performing the indexing in parallel.

We believe that it is possible to create several small indexes instead of a single very large one. If certain rules are followed when creating the distributed index, it should be possible to merge the results of querying the set of small indexes and to obtain a performance (recall and precision) comparable to the results obtained using a single index. The test setup we built in order to perform the experiments required for TREC-2 should allow us to test these hypotheses. The advantages of a distributed index are clear:

(1) The indexing process would be faster.

(2) Each one of the distributed indexing processes would be smaller and less fragile.

(3) Even if one of the distributed processes crashes restarting it would be less expensive.

(4) A distributed system would be much easier to update, i.e., adding a new document would not require to reindex the whole corpus.

(5) A distributed system would be more likely to be useful in order to study the kinds of problems and solutions that are likely to be encountered in a real world situation.

SUMMARY OF RESULTS

We have processed the total of 850 MBytes of text during TREC-2. The first 550 MBytes were articles from the Wall Street Journal which were previously processed for TREC-1; we had to repeat most of the processing to correct early tokenization errors introduced by the tagger. The entire process (tagging, parsing, phrase extraction) took just over 4 weeks on 2 Sun's SparcStations (1 and 2). Building a comprehensive index for the WSJ database took up another 2 weeks. This time we were able to create a single index thanks to the improved indexing software received from NIST. The final index size was 204 MBytes, and included 2,274,775 unique terms (of which about 310,000 were single word terms, and the remaining 1,865,000 were syntactic word pairs) occurring in 173,219 documents, or more than 13 (unique) terms per document. Note that this gives potentially much better coverage of document content than single word terms alone with less than 2 unique terms per document. We say 'potentially' since the process of deriving phrase-level terms from text is still only partially understood, including the complex problem of 'normalization' of representation.

The remaining 300 MBytes were articles from the San Jose Mercury News, which were contained in TIPSTER disk-3. Processing of this part, and creating an index for routing purposes took about 3 weeks. While natural language processing required 2 weeks to complete (at approximately the same speed as WSJ database), we were able to cut indexing time in half by using a faster in-memory version of the NIST system. This new version reduces the time required by the first phase of indexing from days to hours, however the second phase remains slow (days) and fragile (we had to redo it 3 times). The final size of the SJMN index was 101 MBytes, with 1,535,971 unique terms occurring in 86,270 documents (nearly 18 unique terms per document).[13]

Two types of retrieval have been done: (1) new topics 101-150 were run in the ad-hoc mode against WSJ database, and (2) topics 51-100, previously used in TREC-1, were run in the routing mode against SJMN database. In each category several runs were attempted

[13] It has to be noted that the ratios at which new terms are generated are nearly identical in both databases; at 86,319 documents (or about half way through WSJ database) 1,335,622 unique terms had been recorded.

with a different combination of fields from the topics used to create search queries. A typical topic is shown below:

<top>
<head> Tipster Topic Description
<num> Number: 107
<dom> Domain: International Economics
<title> Topic: Japanese Regulation of Insider Trading
<desc> Description:
Document will inform on Japan's regulation of insider trading.
<narr> Narrative:
A relevant document will provide data on Japanese laws, regulations, and/or practices which help the foreigner understand how Japan controls, or does not control, stock market practices which could be labeled as insider trading.
<con> Concept(s):
1. insider trading
2. Japan
3. Ministry of Finance, Securities and Exchange Council, Osaka Securities Exchange, Tokyo Stock Exchange
4. Securities and Exchange Law, Article 58, law, legislation, guidelines, self-regulation
5. Nikko Securities, Yamaichi Securities, Nomura Securities, Daiwa Securities, Big Four brokerage firms
<fac> Factor(s):
<nat> Nationality: Japan
</fac>
</top>

This topic actually consists of two different statements of the same query: the natural language specification consisting of <desc> and <narr> fields, and an expert-selected list of key terms which are often far more informative than the narrative part (in some cases these terms were selected via feedback in actual retrieval attempts). The table below shows the search query obtained from fields <desc> and <narr> of topic 107, after an expansion with similar terms and deleting low-content terms.

Query 107

standard+trade	idf=16.81	weight=0.38
standard+trade	idf=16.81	weight=0.38
regulate+japanese	idf=15.40	weight=1.00
standard+japanese	idf=14.08	weight=0.38
regulate+trade	idf=12.84	weight=1.00
regulate+trade	idf=12.84	weight=1.00
controls	idf=9.97	weight=1.00
labele	idf=9.20	weight=1.00
trade+inside	idf=8.62	weight=1.00
trade+inside	idf=8.62	weight=1.00
inside	idf=6.49	weight=1.00
inside	idf=6.49	weight=1.00
inside	idf=6.49	weight=1.00
regulate	idf=5.66	weight=1.00
regulate	idf=5.66	weight=1.00
regulate	idf=5.66	weight=1.00
practice	idf=5.46	weight=1.00
practice	idf=5.46	weight=1.00
data	idf=4.91	weight=1.00
data	idf=4.91	weight=0.51
data	idf=4.91	weight=0.26
japanese	idf=4.84	weight=1.00
japanese	idf=4.84	weight=1.00
standard	idf=4.81	weight=0.38
standard	idf=4.81	weight=0.38
standard	idf=4.81	weight=0.38
inform	idf=4.71	weight=1.00
inform	idf=4.71	weight=0.26
inform	idf=4.71	weight=0.51
protect	idf=4.69	weight=0.41

Note that many 'function' words have been removed from the query, e.g., *provide*, *understand*, 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., *data* and *inform*, because these could not be uniformly considered as 'common' across all queries.

Results obtained for queries using text fields only and those based primarily on keyword fields are reported separately. The purpose of this distinction was to demonstrate that (or whether) an intensive natural language processing can make an imprecise and frequently convoluted narrative into a better query that an expert would create.[14]

The ad-hoc category runs were done as follows (these are the official TREC-2 results):

(1) *nyuir1*: An automatic run of topics 101-150 against the WSJ database with the following fields used: <title>, <desc>, and <narr> only. Both syntactic phrases and term similarities were included.

(2) *nyuir2*: An automatic run of topics 101-150 against the WSJ database with the following fields used: <title>, <desc>, <con> and <fac> only. Both syntactic phrases and term similarities were included.

---

[14] Some results on the impact of different fields in TREC topics on the final recall/precision results were reported by Broglio and Croft (1993) at the ARPA HLT workshop, although text-only runs were not included. One of the most striking observations they have made is that the narrative field is entirely disposable, and moreover that its inclusion in the query actually hurts the system's performance. Croft (personal communication, 1992) has suggested that excluding all expert-made fields (i.e., <con> and <fac>) would make the queries quite ineffective. Broglio (personal communication, 1993) confirms this showing that text-only retrieval (i.e., with <desc> and <narr>) shows an average precision at more than 30% below that of <con>-based retrieval.

(3) *nyuir3*: A run of manually pruned topics 101-150 against the WSJ database with the following fields used: <title>, <desc>, <con> and <fac> only. Both syntactic phrases and term similarities were included. Manual intervention involved removing some terms from queries before database search.

Summary statistics for these runs are shown in Table 2. In addition, the 'base' column reports the system's performance on text fields with no language preprocessing, and no phrase terms or similarities used. We must note, however, that in all cases the topics have been processed with our suffix-trimmer, which means some NLP has been done already (tagging + lexicon), and therefore what we do not see here a performance of 'pure' statistical system.

In the routing category only automatic runs were done (again, these are the official TREC-2 results):

(1) *nyuir1*: An automatic run of topics 51-100 against the SJMN database with the following fields used: <title>, <desc>, and <narr> only. Both syntactic phrases and term similarities were included.

(2) *nyuir2*: An automatic run of topics 51-100 against the SJMN database with the following fields used: <title>, <desc>, <con> and <fac> only. Both syntactic phrases and term similarities were included.

A (simulated) routing mode run means that queries (i.e., terms and their weights) were derived with respect to a (different) training database (here WSJ), and were subsequently run against the new database (here SJMN). In particular, this means that the terms and their relative importance (reflected primarily through idf weights) were those of WSJ database rather than SJMN database.

Routing runs are summarized in Table 3. Again a column 'base' is added to show the system's performance without NLP module. We may note that the routing results are generally well below the ad-hoc results, both because the base system performance is inferior and because query processing has a different effect on the final statistics. The last column is a post-TREC run.[15]

[15] It should be noted that in category B runs, three topics (63, 65, and 88) had no relevant documents in SJMN database. Unfortunately, the evaluation program counts those as if there were relevant documents but none had been found, thus underestimating the system's performance by 5 to 8%. Excluding these three topics from consideration we obtain, in the last column, the average precision of 0.2624 and the R-precision of 0.3000.

Run Name	base ad-hoc	nyuir1 ad-hoc	nyuir2 ad-hoc	nyuir3 ad-hoc
Queries	50	50	50	50
Tot number of docs over all queries				
Ret	49887	49884	49876	49877
Rel	3929	3929	3929	3929
RelRet	2740	2983	3274	3281
Recall	(interp) Precision Averages			
0.00	0.7038	0.7013	0.7528	0.7528
0.10	0.4531	0.4874	0.5567	0.5574
0.20	0.3708	0.4326	0.4721	0.4724
0.30	0.3028	0.3531	0.4060	0.4076
0.40	0.2550	0.3076	0.3617	0.3621
0.50	0.2059	0.2637	0.3135	0.3142
0.60	0.1641	0.2175	0.2703	0.2711
0.70	0.1180	0.1617	0.2231	0.2237
0.80	0.0766	0.1176	0.1667	0.1697
0.90	0.0417	0.0684	0.0915	0.0916
1.00	0.0085	0.0102	0.0154	0.0160
Average precision over all rel docs				
Avg	0.2224	0.2649	0.3111	0.3118
Precision at				
5 docs	0.4640	0.4920	0.5360	0.5360
10 docs	0.4140	0.4420	0.4880	0.4880
15 docs	0.3867	0.4240	0.4693	0.4707
20 docs	0.3670	0.4050	0.4390	0.4410
30 docs	0.3253	0.3640	0.4067	0.4080
100 docs	0.2304	0.2720	0.3094	0.3094
200 docs	0.1626	0.1886	0.2139	0.2140
500 docs	0.0911	0.1026	0.1137	0.1140
1000 docs	0.0548	0.0597	0.0655	0.0656
R-Precision (after RelRet)				
Exact	0.2605	0.3003	0.3320	0.3321

Table 2. Automatic ad-hoc run statistics for queries 101-150 against WSJ database: (1) *base* - statistical terms only with <desc> and <narr> fields; (2) *nyuir1* - using syntactic phrases and similarities with <desc> and <narr> fields only; (3) *nyuir2* - same as 2 but with <desc>, <con>, and <fac> fields only; and (4) *nyuir3* - same as 3 but queries manually pruned before search.

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	50000
Rel	2064	2064	2064	2064
RelRet	1349	1390	1610	1623
Recall	(interp) Precision Averages			
0.00	0.5276	0.5400	0.6435	0.6458
0.10	0.3685	0.3937	0.4610	0.5021
0.20	0.3054	0.3423	0.3705	0.4151
0.30	0.2373	0.2572	0.3031	0.3185
0.40	0.2039	0.2263	0.2637	0.2720
0.50	0.1824	0.2032	0.2282	0.2379
0.60	0.1596	0.1674	0.1934	0.1899
0.70	0.1167	0.1295	0.1542	0.1571
0.80	0.0854	0.0905	0.1002	0.1163
0.90	0.0368	0.0442	0.0456	0.0434
1.00	0.0228	0.0284	0.0186	0.0158
Average precision over all rel docs				
Avg	0.1884	0.2038	0.2337	0.2466
Precision at				
5 docs	0.3160	0.3360	0.4280	0.4440
10 docs	0.3100	0.3240	0.4000	0.4180
15 docs	0.2813	0.2933	0.3613	0.3800
20 docs	0.2670	0.2790	0.3260	0.3530
30 docs	0.2240	0.2404	0.2760	0.2993
100 docs	0.1306	0.1412	0.1708	0.1698
200 docs	0.0865	0.0939	0.1078	0.1107
500 docs	0.0464	0.0489	0.0575	0.0570
1000 docs	0.0270	0.0278	0.0322	0.0325
R-Precision (after Rel)				
Exact	0.2196	0.2267	0.2513	0.2820

Table 3. Automatic routing run statistics for queries 51-100 against SJMN database: (1) *base* - statistical terms only with <desc> and <narr> fields; (2) *nyuir1* - using syntactic phrases and similarities with <desc> and <narr> fields only; (3) *nyuir2* - same as 2 but with <desc>, <con>, and <fac> fields only; and (4) *nyuir2a* - run *nyuir2* repeated with new weighting for phrases.

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 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.[16]

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.

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.

[16] 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.

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. Table 4 summarizes these new runs for queries 101-150 against WSJ database. Similar improvements have been obtained for queries 51-100.

The results of the routing runs against SJMN database are somewhat more troubling. Applying the new weighting scheme we did see the average precision increase by some 5 to 12% (see column 4 in Table 3), but the results remain far below those for the ad-hoc runs. Direct runs of queries 51-100 against SJMN database produce results that are about the same as in the routing runs (which may indicate that our routing scheme works fine), however the same queries run against WSJ database have retrieval precision some 25% above SJMN runs. This may indicate some problems with SJMN database or the relevance judgements for it.

Run Name	nyuir1 ad-hoc	nyuir1a ad-hoc	nyuir2 ad-hoc	nyuir2a ad-hoc
Queries	50	50	50	50
Tot number of docs over all queries				
Ret	49884	50000	49876	50000
Rel	3929	3929	3929	3929
RelRet	2983	3108	3274	3401
Recall	(interp) Precision Averages			
0.00	0.7013	0.7201	0.7528	0.8063
0.10	0.4874	0.5239	0.5567	0.6198
0.20	0.4326	0.4751	0.4721	0.5566
0.30	0.3531	0.4122	0.4060	0.4786
0.40	0.3076	0.3541	0.3617	0.4257
0.50	0.2637	0.3126	0.3135	0.3828
0.60	0.2175	0.2752	0.2703	0.3380
0.70	0.1617	0.2142	0.2231	0.2817
0.80	0.1176	0.1605	0.1667	0.2164
0.90	0.0684	0.1014	0.0915	0.1471
1.00	0.0102	0.0194	0.0154	0.0474
Average precision over all rel docs				
Avg	0.2649	0.3070	0.3111	0.3759
Precision at				
5 docs	0.4920	0.5200	0.5360	0.6040
10 docs	0.4420	0.4900	0.4880	0.5580
15 docs	0.4240	0.4653	0.4693	0.5253
20 docs	0.4050	0.4420	0.4390	0.4980
30 docs	0.3640	0.3993	0.4067	0.4607
100 docs	0.2720	0.2914	0.3094	0.3346
200 docs	0.1886	0.2064	0.2139	0.2325
500 docs	0.1026	0.1103	0.1137	0.1229
1000 docs	0.0597	0.0622	0.0655	0.0680
R-Precision (after Rel)				
Exact	0.3003	0.3332	0.3320	0.3950

Table 4. Automatic ad-hoc run statistics for queries 101-150 against WSJ database: (1) *nyuir1* - TREC-2 official run with <desc> and <narr> fields only; (2) *nyuir1a* - revised term weighting run; (3) *nyuir2* - official TREC-2 run with <desc>, <con>, and <fac> fields only; and (4) *nyuir2a* - revised weighting run.

'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:

Full 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).

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 can match those of traditional statistical programs such as key-word indexing or statistical phrase extraction. 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.

On the other hand, we must be aware of the limits of NLP technologies at our disposal. While part-of-speech tagging, lexicon-based stemming, and parsing can be done on large amounts of text (hundreds of millions of words and more), other, more advanced processing involving conceptual structuring, logical forms, etc., is still beyond reach, computationally. It may be assumed that these super-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 (e.g., Mauldin, 1991).

ACKNOWLEDGEMENTS

We would like to thank Donna Harman of NIST for making her PRISE system available to us. We would also like to thank Ralph Weischedel and Heidi Fox 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 Project Agency under Contract N00014-90-J-1851 from the Office of Naval Research, under Contract N00600-88-D-3717 from PRC

Inc., and the National Science Foundation under Grant IRI-93-02615. We also acknowledge support from the Canadian Institute for Robotics and Intelligent Systems (IRIS).

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.

Mauldin, Michael. 1991. ‘‘Retrieval Performance in Ferret: A Conceptual Information Retrieval System.’’ Proceedings of ACM SIGIR-91, pp. 347-355.

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 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.