Pergamon

Information Processing & Management, Vol. 31, No. 3, pp. 397-417, 1995

0306-4573/95 $9.50 + .00

0306-4573(94)00055-7

NATURAL LANGUAGE INFORMATION RETRIEVAL

TOMEK STRZALKOWSKI
GE Research and Development Center,
1 River Rd., Schenectady, NY 12301, U.S.A.
E-mail: tomek@thuban.crd.ge.com

Abstract—In this paper we describe an information retrieval system in which advanced natural language processing techniques are used to enhance the effectiveness of term-based document retrieval. The backbone of our system is a traditional statistical engine that 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 (a) preprocess the documents in order to extract content-carrying terms, (b) discover inter-term dependencies and build a conceptual hierarchy specific to the database domain, and (c) process the user's natural language requests into effective search queries. During the course of the Text REtrieval Conferences, TREC-1 and TREC-2,* our system has evolved from a scaled-up prototype, originally tested on such collections as CACM-3204 and Cranfield, to its present form, which can be effectively used to process hundreds of millions of words of unrestricted text.

INTRODUCTION

The task of information retrieval is to extract *relevant* documents from a large collection of documents in response to user queries. When the documents contain primarily unrestricted text (e.g., newspaper articles, legal documents, etc.), the relevance of a document is established through 'full-text' retrieval. This has usually been accomplished by identifying key terms in the documents (the process known as 'indexing'), which could then be matched against terms in queries (Salton, 1989). The effectiveness of any such term-based approach is directly related to the accuracy with which a set of terms represents the content of a document, as well as how well it contrasts a given document with respect to other documents. In other words, we are looking for a representation R such that for any text items $D1$ and $D2$, $R(D1) = R(D2)$ iff $meaning(D1) = meaning(D2)$, at an appropriate level of abstraction (which may depend on the types and character of anticipated 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 the database domain. For example, *joint venture* is an important term in the *Wall Street Journal* (WSJ henceforth) database, whereas neither *joint* nor *venture* is important by itself. In fact, in a 800+ MByte database, both *joint* and *venture* would often be dropped from the list of terms by the system because their inverted document frequency (*idf*) weights were too low. In large databases comprising hundreds of thousands of documents, the use of phrasal terms is not just desirable, it becomes necessary.

To illustrate this point, let us consider TREC Topic 104, an information request from which a database search query is to be built. The reader may note various sections of this Topic, with ⟨desc⟩ corresponding to the user's original request, further elaborated in ⟨narr⟩, and ⟨con⟩ consisting of expert-assigned phrases denoting key concepts to be considered.

⟨top⟩
⟨num⟩ Number: 104
⟨dom⟩ Domain: Law and Government

*See Harman (1993) for a detailed description of TREC.

⟨title⟩ Topic: Catastrophic Health Insurance
⟨desc⟩ Description:
Document will enumerate provisions of the U.S. Catastrophic Health Insurance Act of 1988, or the political/legal fallout from that legislation.
⟨narr⟩ Narrative:
A relevant document will detail the content of the U.S. medicare act of 1988 which extended catastrophic illness benefits to the elderly, with particular attention to the financing scheme which led to a firestorm of protest and a Congressional retreat, or a relevant document will detail the political/legal consequences of the catastrophic health insurance imbroglio and subsequent efforts by Congress to provide similar coverages through a less-controversial mechanism.
⟨con⟩ Concept(s):
1. Catastrophic Coverage Act of 1988, Medicare Part B, Health Care Financing Administration
2. catastrophic-health program, catastrophic illness, catastrophic care, acute care, long-term nursing home care
3. American Association of Retired Persons, AARP, senior citizen, National Committee to Preserve Social Security and Medicare
⟨fac⟩ Factor(s):
⟨nat⟩ Nationality: U.S.
⟨/fac⟩
⟨/top⟩

Let us now suppose that we build a search query from the above topic using single-word terms only, that is, using stemmed single words occurring in the topic, except for the stop-words such as articles or prepositions. Using this query to search the database, our system would produce an output as shown in the left column below, which lists the ranks and scores of relevant documents within the top 100 retrieved documents. On the other hand, when we include in the query simple two-word phrases, such as *catastrophic illness*, *acute care*, *home care*, *nursing care*, *senior citizen*, etc., all of which appear in the topic, we can considerably sharpen the outcome of the search as seen in the right column. In both cases we assume that phrasal terms are represented in database documents.

QUERY:104; NO. RELEVANT:21

no phrases

REL DOCUMENT	RANK	SCORE
WSJ890918-0173	2	53448
WSJ891004-0119	7	44423
WSJ870723-0064	8	44227
WSJ870213-0053	10	42477
WSJ880608-0121	14	38913
WSJ891005-0005	15	38173
WSJ891009-0009	35	28252
WSJ890920-0115	39	26660
WSJ890928-0184	40	26058
WSJ880609-0061	53	24511
WSJ891009-0188	73	22389
WSJ880705-0194	97	20932

simple phrases

REL DOCUMENT	RANK	SCORE
WSJ891004-0119	1	38405
WSJ891005-0005	4	29709
WSJ890918-0173	5	29004
WSJ880608-0121	7	27971
WSJ870723-0064	8	27954
WSJ870213-0053	12	25655
WSJ891009-0009	18	23061
WSJ890920-0115	26	21184
WSJ891009-0188	46	18111
WSJ880609-0061	50	17469
WSJ870601-0075	52	17418
WSJ890928-0184	61	16767
WSJ891005-0001	72	15700
WSJ871028-0059	93	14359
WSJ880705-0194	95	14342

We may notice that more relevant documents are retrieved in the phrasal search, and their ranks are generally higher than in the single-word term search. The reason is, as one may expect, that relevant documents have more phrases in common with the query than nonrelevant documents, and that a match on a phrasal term would be a better indicator of relevance than a match on a single-word term, and therefore would command a higher score. Moreover, it will be harder for a nonrelevant document to rack up a comparable score on single-word term matches, which are now worth relatively less than phrasal-term

matches. Consider, for example, the document WSJ891005-0005, which moved from position 15 in the left column to position 4 in the right column. A closer analysis reveals that in addition to the single-word term matches, this document also has the following phrasal terms in common with the query: *catastrophic care*, *senior citizen*, *social security*, *catastrophic illness*, *long-term care*, *nursing home*, and *retired person*. On the other hand, we also notice that the document WSJ890928-0184 actually drops from position 40 to position 61, which is an 18-position slide, if we do not count other relevant documents that moved ahead of it. Again, a closer look at this document shows that there are only two phrasal term matches here, both relatively unspecific and likely to be found in unrelated documents: *social security* and *health care*. Therefore, some relevant documents, namely, those with little vocabulary overlap with the query, may in fact lose out. This problem may be alleviated to some degree by allowing query expansion with related terms from a conceptual domain map. We discuss this issue further in this paper.

What follows from the above discussion is that 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 the 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").

For all kinds of terms that can be assigned to the representation of a document (e.g., words, syntactic phrases, fixed phrases, and proper names), various levels of "regularization" are needed to assure that syntactic or lexical variations of input do not obscure underlying semantic uniformity. Without actually doing semantic analysis, this kind of normalization can be achieved through the following processes:

1. morphological stemming (e.g., *retrieving* is reduced to *retriev*);
2. lexicon-based word normalization (e.g., *retrieval* is reduced to *retrieve*);
3. operator-argument representation of phrases (e.g., *information retrieval*, *retrieving of information*, and *retrieve relevant information* are all assigned the same representation, *retrieve+information*);
4. context-based term clustering into synonymy classes and subsumption hierarchies (e.g., *takeover* is a kind of *acquisition* (in business), and *Fortran* is a *programming language*).

We should note here that an effect similar to regularization can be achieved through the generation of alternative variants (lexical, syntactic, etc.), or by retaining ambiguities in the representation (Sparck Jones & Tait, 1984; Metzler *et al.*, 1989; Smeaton & Sheridan, 1991). This approach, though clearly less efficient, can be attractive in situations where only the queries are to be processed, not the database.

Introduction of compound terms complicates the task of discovery of various semantic relationships among them. For example, the term *natural language* can often be considered 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 the 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 the 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.

NLP IN IR: A BRIEF OVERVIEW

Natural language processing has always seemed to offer the key to building an ultimate information retrieval system. Somehow we feel that the "bag-of-words" representations, prevalent among today's information retrieval systems, can hardly do justice to the complexities of free, unprocessed text with which we have to deal. Some of the favorite examples include *Venetian blind vs. blind Venetian*, *Poland is attacked by Germany vs. Germany attacks Poland*, or *car wash vs. automobile detailing*. Natural language processing could provide solutions to at least some of these problems through lexical and syntactic analysis of text (e.g., *Venetian* is used as either an adjective or a noun), through the assignment of logical structures to sentences (e.g., *Germany* is a logical subject of *attack*), or through an advanced semantic analysis that may involve domain knowledge. Other important applications include discourse-level processing, resolution of anaphoric references, proper name identification, and more.

Unfortunately, a direct application of NLP techniques to information retrieval has met some rather severe obstacles, chief among which was a paralyzing lack of robustness and efficiency. Worse yet, the difficulties did not end with the linguistic processing itself, but extended to the representation it produced: it was not at all clear how the complex structures could be effectively compared to determine relevance. A better approach, it seemed, was to use NLP to assist an IR system, whether boolean, statistical, or probabilistic, in automatically selecting important terms, words, and phrases, which could then be used in representing documents for search purposes. This approach provided extra maneuverability for softening any inadequacies of the NLP software without incapacitating the entire system. Efficiency problems still prevented direct online processing of any large amount of text, but NLP could be gradually built into an offline database indexing.

There has been a considerable amount of interest in using NLP in information retrieval research, with specific implementations varying from the word-level morphological analysis to syntactic parsing to conceptual-level semantic analysis. Some interesting insights have been made; however, demonstrating the superiority of these techniques over simple statistical processing has proved harder than expected. One of the more comprehensive evaluations of NLP use in information retrieval was done by Fagan (1987), who used syntactic and 'statistical' phrases to index documents in five different test collections. He showed that phrases could be considerably more effective than single words in representing semantic content of documents, and that phrases derived by statistical means were as effective as those derived through parsing. Still, the overall improvements in retrieval effectiveness were relatively small (2–23%) and inconsistent, perhaps because of the small scale of these experiments. Further experiments with statistical phrases (Lewis & Croft, 1990) generally did not confirm these numbers, showing much lesser improvements (5–8%). Other notable attempts at using compound phrasal terms in indexing or retrieval include Dillon and Gray (1983), Sparck Jones and Tait (1984), Smeaton and van Rijsbergen (1988), Metzler *et al.* (1989), Ruge *et al.* (1991), and Evans *et al.* (1994). Not all of these approaches were properly evaluated, and for those that were evaluated the results were generally discouraging. More advanced techniques aimed at overcoming the limitations of shallow linguistic processing that included semantic and discourse-level processing proved unacceptably expensive and difficult to evaluate on established benchmarks despite some spectacular

results obtained in laboratory tests (e.g., Mauldin, 1991). We must note, however, that in none of these systems was NLP done on an appropriate scale; either it was an advanced NLP system applied to a small-scale task (e.g., Metzler *et al.*'s COP or Mauldin's FERRET), or a relatively shallow NLP applied to a larger task (e.g., Evans *et al.*'s CLARIT). A notable exception is DR-LINK (Liddy & Myaeng, 1994), an advanced conceptual IR system, indeed, in some respects more advanced than our present system. DR-LINK took part in TREC evaluations, but failed to produce any comparable results.

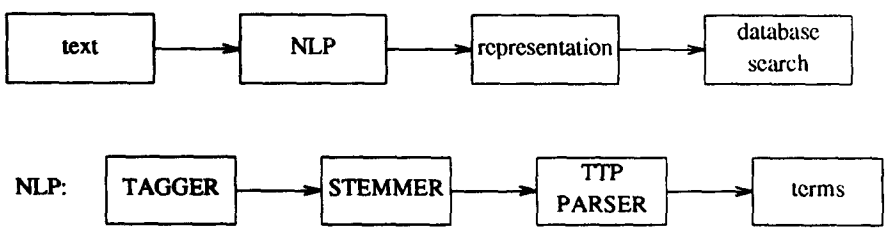
A common theme of the majority of NLP applications to information retrieval is to use linguistically motivated objects (stems, phrases, proper names, fixed terminology, lexical correlations, etc.) as derived from documents and queries, and create a 'value-added' representation, usually in the form of an inverted index. The bulk of this representation may still rest upon statistically weighted single-word terms, whereas additional terms (e.g., phrases) are included on the assumption that they can only make the representation richer, and in effect, improve the effectiveness of subsequent search. For example, if the search finds the phrase *Venetian blind* in a document, we should have more confidence as to the relevance of this document than when *Venetian* and *blind* are found separately. This, however, does not seem to happen in any consistent manner. The problem is that the phrases are not all alike, and their abilities to reflect the content of the text vary greatly with the type of the phrase and its position within the text. Statistical weighting schemes developed for single-word terms, such as *tf.idf*, do not seem to extend on compound terms. Moreover, compound terms derived through statistical means (e.g., co-occurrence, mutual information) tend to behave differently from those derived with a grammar-based parser.

A retrieval model, which includes a weighting scheme for terms, is a crucial part of any information retrieval system, and a wrong retrieval model can defeat even the most accomplished representation. Nonetheless, most work on NLP in IR concentrated on representation or compound term matching strategies, with relatively little consideration given to term weighting and to scoring of retrieved documents. Some commonly used strategies, where a phrase weight was a function of weights of its components, did not produce uniform results (Fagan, 1987; Lewis & Croft, 1990). In fact, the lack of an established retrieval model that can handle linguistically motivated compound terms may be the most serious obstacle in evaluating the impact and feasibility of natural language processing in information retrieval.

In the remainder of this paper, we discuss particulars of the present system and some of the observations made while processing TREC data, with some emphasis on term weighting issues. 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

We have established the general architecture of an NLP-IR system, depicted schematically below, in which an advanced NLP module is inserted between the textual input (new documents, user queries) and the database search engine (in our case, NIST's PRISE system; Harman & Candela, 1988). This design has already shown some promise in producing a better performance than the base statistical system (Strzalkowski, 1993a; Strzalkowski & Carballo, 1994).



In our system, the database text is first processed with a sequence of programs that include a part-of-speech tagger, a lexicon-based morphological stemmer, and a fast syn-

tactic 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. An important characteristic of this system, not visible from the above scheme, is its scale: NLP with this level of complexity has never before been successfully applied to amounts of text measured in hundreds of millions of words.†

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" may be 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 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 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.

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). It currently encompasses most of the grammar productions and many of the restrictions, but it is by no means complete. Unlike a conventional parser, TTP's output is a regularized representation of each sentence that reflects its logical predicate-argument structure (e.g., logical subject and logical objects are identified depending upon the main verb subcategorization frame). For example, the verb *abide* has, among others, a subcategorization frame in which the object is a prepositional phrase with *by*, as in *he'll abide by the court's decision*, that is,

ABIDE: *subject* NP *object* PREP by NP.

Subcategorization information is read from the online Oxford Advanced Learner's Dictionary (OALD), which TTP uses.

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 severe time pressure. In the runs with approximately 130 million words of TREC's *Wall Street Journal* and *San Jose Mercury* texts, which contain over 6 million sentences or 0.85 GigaBytes, the parser's speed averaged between 0.3 and 0.5 seconds per sentence, or up to 70 words per second, on a Sun SparcStation2. In addition, TTP has been shown to produce parse structures no worse than those generated by full-scale linguistic parsers when compared to hand-coded parse trees obtained from the University of Pennsylvania Treebank Project database. This is not to say that TTP is anywhere close to where we would like it to be in terms of the quality of output it generates, or even in speed. We continue to improve various aspects of the parsing process including its grammar coverage, robustness, and efficiency.

†For the upcoming TREC-3 evaluation, we are processing 3 GBytes of text, or more than half a billion words.

TTP is a full-grammar parser, and initially, it attempts to generate a complete analysis for each sentence. However, it also 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 attempts to reduce incomplete constituents forcibly, possibly skipping portions of input in order to restart processing at a next unattempted constituent; in other words, the parser favors reduction to backtracking. 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 scanner 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 and TREC-2 reports (Strzalkowski, 1993a; Strzalkowski & Carballo, 1994), 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 part of speech tagger to preprocess the text (see TREC-1 report for details). A part-of-speech tagger assigns a part of speech label to each word in a text depending on the labels assigned to the preceding words. Often, more than one part-of-speech tag is assigned to a single word, presumably reflecting some kind of ambiguity in the input. In order to control the amount of uncertainty of the input due to multiple tags, we may limit the number of tags the parser will actually see. In the best-tag-only option, which we used in our experiments, the parser looks only at the top-ranked tag for each word. Although alternative (but also less likely) readings of ambiguous sentences may be lost this way, the gain in speed and robustness of the parser greatly outweights any lack of completeness.

In order to streamline the processing, we also perform morphological normalization of words on the tagged text, but before parsing. This is possible because of the part-of-speech tags retain the information about each word's original form. Thus, the sentence *The Soviets have been notified* is transformed into *the/dt soviet/nps have/vbp be/vbn notify/vbn* before parsing commences. The tags are read as follows: *dt* is determiner, *nps* is proper name, *vbp* is tensed plural verb, and *vbn* is past participle.

Morphological normalization of words, or stemming, has been an effective way of improving document recall, since it reduces the number of word forms, 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 form. A stemmer usually cuts words down to their morphological root, thus replacing words with shorter strings that are not necessarily words by themselves (in the lexical sense). In our system we replaced a traditional morphological stemmer with a conservative dictionary-assisted suffix trimmer. 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). At present we do not analyze prefixes: 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).

HEAD-MODIFIER STRUCTURES

Syntactic phrases extracted from TTP parse structures are represented as head-modifier pairs. The head in such a pair is a central element of a phrase (main verb, main noun, etc.), and the modifier is one of the adjuncts or arguments of the head. In the TREC experiments

reported here we extracted head-modifier word pairs only (i.e., nested pairs were not used even though this was warranted by the size of the database).‡

Figure 1 shows all stages of the initial linguistic analysis of a sample sentence from the WSJ database. The reader may note that the parser's output is a predicate-argument structure centered around the main elements of various phrases. For example, BE is the main predicate (modified by HAVE) with two arguments (*subject*, *object*) and two adjuncts (*adv*, *sub_ord*). INVADE is the predicate in the subordinate clause with two 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. In general, the following types of pairs are considered: (1) a head noun of a noun phrase 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*. We should note here that longer phrases or nested pairs may be more appropriate in some cases. For example, when the phrase *former Soviet president* is broken into *former president* and *Soviet president*, we get something potentially quite different from what the original phrase is intended to refer to, and this may have a negative effect on retrieval precision. Although the longer phrases were not used in TREC-2, we plan to include them in future evaluations. We also attempted to identify and remove any terms that were explicitly negated in order to prevent matches against their positive counterparts, either in the database or in the queries. The reader may note that the representation of phrasal terms is kept as simple as possible, so that ordinary string matching routines used in most IR systems are straightforwardly applicable. This places an extra burden on the NLP module, which needs to produce structurally unambiguous representations of phrases (i.e., either *dynamic (information processing)* or *(dynamic information) processing*). An alternative would be to retain such ambiguities in the representation (e.g., COP system; Metzler *et al.*, 1989), and use complex structure matching procedures during retrieval; however, such approaches have thus far proved untenable in larger-scale tasks.

Nonetheless, the notorious ambiguity of nominal compounds remains a serious difficulty in obtaining head-modifier pairs of highest accuracy. In order to cope with this,

‡As the database grows in size, the frequency of some compound terms increases to the point where their indexing value can no longer be ignored.

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

TAGGED SENTENCE
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*

TAGGED & STEMMED SENTENCE
the/*dt* former/*jj* soviet/*jj* president/*nn* have/*vbz* be/*vbn* a/*dt* local/*jj* hero/*nn* ever/*rb* since/*in* a/*dt* russian/*jj* tank/*nn* invade/*vbd* wisconsin/*np* ./*per*

PARSED SENTENCE

```
[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]]]]]]]]]
```

EXTRACTED TERMS & 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

Fig. 1. Stages of sentence processing.

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, whereas 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. In addition, phrases with a significant number of occurrences across different documents, including those for which no clear disambiguation into pairs can be obtained, are included as a third level of index (beside single-word terms, and pairs).

COMPUTING LEXICAL CORRELATIONS BETWEEN TERMS

Head-modifier pairs also serve as occurrence contexts for terms included in them: whether they are single words (as shown in Fig. 1) or nested pairs (e.g., *country+*[*world+third*]). If two terms tend to be modified with a number of common modifiers, or appear as modifiers of common head terms, but otherwise they 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: in general, high-content terms appearing in multiple identical contexts will be considered strong candidates for lexical similarity, provided that these contexts are not too commonplace; for instance, it would not be appropriate to predict similarity between *language* and *logarithm* on the basis of their co-occurrence with *natural*. Figure 2 shows examples of terms sharing a number of common contexts along with frequencies of occurrence in a subset of *Wall Street Journal* database. A *head context* is where two distinct modifiers are attached to the same head element; a *mod context* is where the same term modifies two distinct heads.

To compute term similarities, we used a variant of weighted Jaccard's measure described in, for example, Grefenstette (1992):

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

$term_1$	$term_2$	*head-ctxt*	f_1	f_2	$term_1$	$term_2$	*head-ctxt*	*mod-ctxt*	f_1	f_2
vice	deputy	president	9295	29	man	boy	story		9	3
		chairman	1007	146			club		6	4
		director	6	158			age		18	3
		minister	37	17			mother		4	5
		premier	7	8				bad	4	4
								young	258	12
								older	18	2

Fig. 2. Example pairs of related terms.

where *att* is the attribute of the term x_i, which is the head context for mod-position terms and the mod context for head-position terms. The weight W of the pair $[x,y]$ is determined by calculating the global entropy of the context term y:

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

where

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

In the above, $f_{x,y}$ stands for absolute frequency of pair $[x,y]$ in the corpus, n_y, is the frequency of term y, and N is the number of single-word terms.

In TREC-1 we used a Mutual Information (MI) based classification formula (e.g., Church & Hanks, 1990; Hindle, 1990), but we found it less effective for diverse databases, such as WSJ (Strzalkowski & Vauthey, 1992; Strzalkowski, 1993a). MI was more useful in dealing with narrow subject domains, for example, Computer Science abstracts from the CACM-3204 collection.

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 $[x_2,y]$ (*mod context*), or $[y,x_1]$ and $[y,x_2]$ (*head context*), such that they each occurred at least K 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 are $M-1$ other such common contexts (there were none in the WSJ database). For smaller or narrow domain databases $M = 2$, $K = 2$ is usually sufficient (e.g., for the CACM-3204 database of computer science abstracts). For large databases covering a diverse subject matter, like WSJ or SJMN (*San Jose Mercury News*), we used $M \geq 5$, $K \geq 3$. This helped to reject clustering of *banana* and *Dominican*, which were found to have two common contexts: *republic* and *plant*, even though this second context occurred in apparently different senses in *Dominican plant* and *banana plant*. However, this simple context counting turned out not to be sufficient. We would still generate fairly strong similarity links between terms such as *aerospace* and *pharmaceutical* where six and more common contexts were found, as shown in Figure 3.§ Note that whereas some of the GEW 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. To filter out such cases, we set minimums for admissible values of GEW and disregarded contexts with entropy weights falling below the threshold. In our recent experiments we found that 0.6 is a good threshold for clusters generated from *Wall Street Journal* text; at present we have no evidence whether it would be suitable for other texts. We also observed that clustering head terms using their modifiers as contexts converges faster and gives generally more reliable links than clustering mod terms using heads as context (e.g., Fig. 3). In our experiments with the WSJ database, we found that an occurrence of a common head context needs to be considered as contributing less to the total context count than an occurrence of a common mod context: we used 0.6 and 1, respectively. Using this formula, terms *man* and *boy* in Fig. 2 share 5.4 contexts (4 head contexts and 3 mod contexts). In Fig. 3, after eliminating contexts with GEW falling below 0.6, we are left with 3 head contexts, for the total context count of just 1.8.

Initially, term similarities are organized into clusters around a centroid term. Figure 4 shows the top 10 elements (sorted by similarity value) of the cluster for *president*. Note that in this case the similarity value drops suddenly after the second element of the cluster. Changes in SIM value are used to determine cut-off points for clusters. In Fig. 4, GTS stands for the Global Term Specificity factor, which is explained in the following section.

§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).

CONTEXT	GEW	freq with *aerospace*	freq with *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

Fig. 3. Common (head) contexts for *aerospace* and *pharmaceutical*.

Sample clusters (after cutoffs) obtained from approx. 250 MByte (42 million word) subset of WSJ (years 1990–1992) are given in Table 1.

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 nonlinguistic clustering (e.g., Sparck Jones & Barber, 1971; Crouch, 1988; Lewis & 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. We must note here that syntactic analysis of text is not necessary for lexical clustering as local context can be established statistically based on word co-occurrences in fixed-size text windows. This has some obvious advantages; for example, proximity-based contexts cross sentence boundaries with no fuss, a result that is difficult to come by with a parser. This property makes proximity-based clustering a helpful tool to deal with short, succinct documents (such as abstracts, or brief summaries, e.g., CACM-3204 collection), but less so with longer texts, where it tends to produce excessively noisy results; see also Grishman *et al.* (1986). Some notable works on word-level and syntactic clustering include Church and Hanks (1990), Ruge (1991), and Grefenstette (1992). Grefenstette, in particular, uses the lexical clusters to expand search queries, and he reports some slight improvements in recall and precision on a medical abstracts database, but stops short of any subcategorization of term-to-term relations.

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

CENTROID	TERM	SIM	GTS
president			0.0011
	director	0.2481	0.0017
	chairman	0.2449	0.0028
	office	0.1689	0.0010
	manage	0.1656	0.0007
	executive	0.1626	0.0012
	official	0.1612	0.0008
	head	0.1564	0.0018
	member	0.1506	0.0014
	lead	0.1311	0.0009

Fig. 4. A cluster for *president*.

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

Word	Cluster	Word	Cluster
takeover	*merge, buy-out, acquire, bid*	*benefit*	*insurance, compensate, aid, payment*
capital	*cash, fund, money*	*complete*	*approve, begin*
attract	*lure, draw*	*charge*	*case, allege, loss, claim, cost*
speculate	*rumor*	*president*	*director, chairman*
sanction	*embargo, restrict, ban, penalty*	*inflate*	*growth, demand*
law	*policy, legislate, regulate, rule*	*earnings*	*profit, revenue, result*
portfolio	*asset, invest, hold*	*outlook*	*forecast, prospect, trend, picture*
industry	*concern, firm, division*	*lawyer*	*attorney*
manage	*executive, director, president, office*	*environ*	*climate*
debt	*loan, secure, bond*	*growth*	*gain, decline, demand, rise, increase, recovery*
counsel	*attorney, administrator, secretary*	*create*	*establish*
competitor	*rival, competition*	*slowdown*	*downturn, slump, decrease, setback*
big	*large, major, huge, small*	*aircraft*	*plane, jet*
enter	*start, begin, join*	*shareholder*	*creditor, customer, bondholder, investor, stockholder, hold*

(adding specializations). 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 such as from *aerospace* to *industry* may actually harm a 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 *merger*, *buy-out*, or *acquisition*. We noted than an average set of similarities generated from a text corpus contains about as may "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.‖ Terms with higher GTS values are generally considered more specific, but the specificity comparison is only meaningful for terms already known to be similar. The new function is calculated according to the following formula:

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

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

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

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

‖We believe that measuring term specificity over document-size contexts (e.g., Sparck Jones, 1972) may not be appropriate in all types of databases. In particular, syntax-based contexts allow for processing texts without any internal document structure.

In the above, d_w is *dispersion* of term w understood as the number of distinct contexts in which w is found. For any two terms w_1 and w_2, and a constant $\delta_1 > 1$, if $GTS(w_2) \geq \delta_1 \cdot GTS(w_1)$, then w_2 is considered more specific than w_1. In addition, if $SIM_{norm}(w_1,w_2) = \sigma > \theta_1$, where θ_1 is an empirically established threshold, then w_2 can be added to the query containing term w_1 with weight $\sigma \cdot \omega$,¶ where ω is the weight w_2 would have if it were present in the query. Similarly, if $GTS(w_2) \leq \delta_2 \cdot GTS(w_1)$ and $GTS(w_1) \leq \delta_2 \cdot GTS(w_2)$, and $SIM_{norm}(w_1,w_2) = \sigma > \theta_2$ (with $1 < \delta_2 < \delta_1$ and $\theta_1 < \theta_2$), then we may consider w_2 as synonymous to w_1. All other relations are discarded. For example, the following were obtained from the WSJ training database:

$$GTS(takeover) = 0.001455,$$
$$GTS(merge) = 0.000945,$$
$$GTS(buy\text{-}out) = 0.002725,$$
$$GTS(acquire) = 0.000579,$$

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* may be either discarded or accepted as synonymous. At this time we are unable to tell synonymous or near synonymous relationships from those that are primarily complementary (e.g., *complete* and *begin*). One possible way to tell synonymous from complementary is to use document or paragraph size contexts in addition to syntactic contexts. For example, we may expect that synonyms will not often appear in the same document, even though they would appear repeatedly in the same syntactic context.

Filtered similarity relations create a domain map of terms. At present it may contain only two types of links: equivalence (synonymy and near-synonymy) and subsumption (specification). Figure 5 shows a small fragment of the map derived from lexical relations computed from WSJ database. The domain map is used to expand user queries with related terms automatically, but it also can be used in a feedback mode where the user is allowed to select additional terms from an appropriate part of the map.

An effective use of the domain map for query expansion is not nearly as easy as it may appear from the preceding paragraphs. For one thing, in order to generate high-quality clusters and relations we have to tighten the criteria of term membership. When these criteria get too tight (which happens easily), few clusters are generated and their impact is thus sidelined. Moreover, an effective query expansion is frequently restricted by a larger context, and inserting wrong specializations of "ambiguous" terms may throw the query completely off balance. In one such case, a query about research on U.S. space *missile* program ("star wars") was expanded with *scud* (a specific type of missile used in the Persian Gulf war).

Query expansion (in the sense considered here, though not necessarily in the same way) has been used widely in information retrieval research (e.g., Sparck Jones & Tait, 1984; Harman, 1988), but the results were usually mixed at best. An interesting alternative is to use term clusters to create new terms, "metaterms," and use them to index the database

¶*For TREC-2 we used several values for* $\theta_{1,2}$ from 0.1 to 0.2; δ varied between 10 and 100.

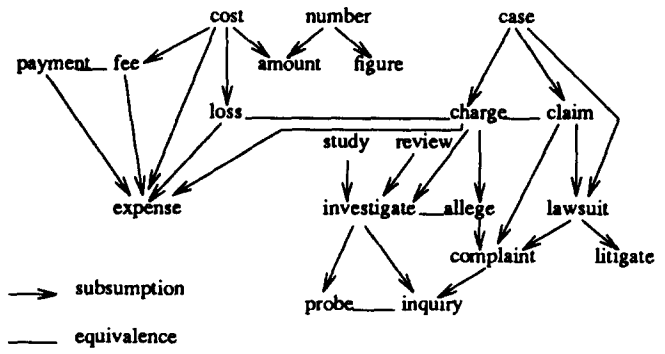


Fig. 5. A fragment of the domain map network. Note the emerging senses of 'charge' as 'expense' and 'allege'.

instead (e.g., Crouch, 1988; Lewis & 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. In fact, a more common use of query expansion is in the relevance feedback, where the original search query is modified based on the content of the documents retrieved in the first run (e.g., Evans *et al.*, 1994).

SUMMARY OF RESULTS

We have processed a total of 850 MBytes of text during TREC-1 and TREC-2.# The first 550 MBytes were articles from the *Wall Street Journal*, from which we created one database. Linguistic processing of this part (tagging, stemming, parsing, phrase extraction) took just over four weeks on two Sun SparcStations (1 and 2). 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 two unique terms per document. We say 'potentially' because 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*, from which the routing database was built. Processing of this part took about two weeks. 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). However, 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 halfway through the WSJ database), 1,335,622 unique terms had been recorded.

Results obtained for queries using text fields only (⟨*desc*⟩, ⟨*narr*⟩) and those based primarily on keyword fields (⟨*con*⟩, ⟨*fac*⟩) are reported separately. The purpose of this distinction was to demonstrate that (or whether) intensive natural language processing can make an imprecise and frequently convoluted narrative into a better query that an expert would create (please consult the example query 104 given at the beginning of this paper for the contents of the different fields). Some results on the performance of different fields in TREC topics on the final recall/precision results were reported by Broglio and Croft (1993). They noted that the narrative field was entirely disposable and, moreover, that its inclusion in the query actually hurt the system's performance. Croft (personal communi-

#In TREC-1 and TREC-2, our group participated in Category B, working with about a quarter of the TIPSTER database of approx. 3 Gigabytes. For the upcoming TREC-3 evaluation we are processing the full database (Category A). Thanks to the improved text processing components, including a much faster tagger, we were able to cut in half the total processing time from text to terms, to approximately 45 minutes per MByte. Further improvements are expected as we optimize the parser through corpus-based training.

cation, 1992) has suggested that excluding all expert-made fields (i.e., ⟨con⟩ and ⟨fac⟩) would make the queries quite ineffective. This has been confirmed by Broglio (personal communication, 1993) who showed that text-only retrieval (i.e., with ⟨desc⟩ and ⟨narr⟩ fields only) had an average precision more than 30% below that of ⟨con⟩-based retrieval.

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.
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 3. In addition, the column 'txt1' 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, we do not see here a performance of a 'pure' statistical system.

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

1. *nyuir1r:* 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. *nyuir2r:* 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 the WSJ database rather than the SJMN database.

Routing runs are summarized in Table 4. Again a column 'txt1' is added to show the system's performance without the 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.*

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-

*In category B runs, three topics (63, 65, and 88) had no relevant documents in the SJMN database; however, the evaluation program counted 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.

document frequency factor is usually normalized by the document length, that is, it is more significant for a term to occur in a short 100-word abstract than in a 5000-word article.†

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 two or more words. Specifically, the system used *lnc-ntc* combination of weights, which is already one of the most effective options of tf.idf (Buckley, 1993). 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 may be inappropriate for mixed term sets, consisting of ordinary concepts, proper names, and 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 were 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 are often decisive for 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, whereas 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.

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

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

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.

Table 2 illustrates the effect of differential weighting of phrasal terms using topic 101 and a relevant document (WSJ870226-0091) as an example. Note that although most of the affected terms have their weights increased, sometimes substantially, for some (e.g., *space+base*) the weight actually decreases. 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 led to an overall increase of precision of more than 20% over our official TREC-2 ad hoc results. Table 3 includes statistics of these new runs for queries 101–150 against the WSJ database. The gap between the precision levels in columns *txt2* and *con* reflects the difference in the quality of the queries obtained directly from user's input (*txt2*) and those obtained primarily from expert's formulation (*con*). The column *txt2+nlp* represents the improvement of user queries thanks to NLP, with as much as 70% of the gap closed. 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

†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.

Table 2. The effect of differential term weighting.

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

Topic 101 matches WSJ870226-0091.
Duplicate terms not shown.

Table 3. Run statistics for ad hoc queries 101-150 against WSJ database with 1000 docs per query.

	Run							
	txt1	nyuir1	txt2	txt2+nlp	nyuir2	nyuir3	con	con+nlp
Tot number of docs over all queries								
Ret	50000	49884	50000	50000	49876	49877	49999	50000
Rel	3929	3929	3929	3929	3929	3929	3929	3929
RelRet	2736	2983	3025	3108	3274	3281	3332	3401
%chg		+9.0	+10.5	+14.7	+19.5	+20.0	+21.8	+24.3
(interp) Precision Averages at Recall								
0.00	0.6874	0.7013	0.7318	0.7201	0.7528	0.7528	0.7469	0.8063
0.10	0.4677	0.4874	0.5293	0.5239	0.5567	0.5574	0.5726	0.6198
0.20	0.3785	0.4326	0.4532	0.4751	0.4721	0.4724	0.4970	0.5566
0.30	0.3060	0.3531	0.3707	0.4122	0.4060	0.4076	0.4193	0.4786
0.40	0.2675	0.3076	0.3276	0.3541	0.3617	0.3621	0.3747	0.4257
0.50	0.2211	0.2637	0.2815	0.3126	0.3135	0.3142	0.3271	0.3828
0.60	0.1765	0.2175	0.2406	0.2752	0.2703	0.2711	0.2783	0.3380
0.70	0.1313	0.1617	0.1783	0.2142	0.2231	0.2237	0.2267	0.2817
0.80	0.0828	0.1176	0.1337	0.1605	0.1667	0.1697	0.1670	0.2164
0.90	0.0451	0.0684	0.0818	0.1014	0.0915	0.0916	0.0959	0.1471
1.00	0.0094	0.0102	0.0159	0.0194	0.0154	0.0160	0.0168	0.0474
Average precision over all rel docs								
Avg	0.2309	0.2649	0.2835	0.3070	0.3111	0.3118	0.3210	0.3759
%chg		+15.0	+22.8	+33.0	+35.0	+35.0	+39.0	+62.8
Precision at								
5 docs	0.5000	0.4920	0.5240	0.5200	0.5360	0.5360	0.5600	0.6040
10 docs	0.4080	0.4420	0.4600	0.4900	0.4880	0.4880	0.5020	0.5580
15 docs	0.3813	0.4240	0.4427	0.4653	0.4693	0.4707	0.4773	0.5253
20 docs	0.3640	0.4050	0.4150	0.4420	0.4390	0.4410	0.4560	0.4980
30 docs	0.3353	0.3640	0.3740	0.3993	0.4067	0.4080	0.4100	0.4607
100 docs	0.2380	0.2720	0.2790	0.2914	0.3094	0.3094	0.3084	0.3346
200 docs	0.1656	0.1886	0.1969	0.2064	0.2139	0.2140	0.2156	0.2325
500 docs	0.0912	0.1026	0.1052	0.1103	0.1137	0.1140	0.1162	0.1229
1000 docs	0.0547	0.0597	0.0605	0.0622	0.0655	0.0656	0.0666	0.0680
R-Precision (after Rel)								
Exact	0.2671	0.3003	0.3053	0.3332	0.3320	0.3321	0.3455	0.3950
%chg		+12.4	+14.3	+24.7	+24.3	+24.3	+29.3	+47.9

(1) txt1 = all single terms of ⟨narr⟩ and ⟨desc⟩ fields—this is the base run; (2) nyuir1 = the official TREC-2 run with ⟨narr⟩ and ⟨desc⟩ only; (3) txt2 = a run with ⟨narr⟩ and ⟨desc⟩ fields with low weight terms removed; (4) txt2+nlp = ⟨narr⟩ and ⟨desc⟩ fields including syntactic phrase terms using the new weighting scheme; (5) nyuir2 = the official TREC-2 run with ⟨desc⟩ and ⟨con⟩ fields; (6) con = ⟨desc⟩ and ⟨con⟩ fields with low weight terms removed but with no NLP; and (7) con+nlp = ⟨desc⟩ and ⟨con⟩ fields including phrases with the new weighting scheme. In all cases documents preprocessed with the lexicon-based suffix-trimmer.

Table 4. Automatic routing run statistics for queries 51-100 against SJMN database.

	Run					
	txt1	txt2	nyuir1r	con	nyuir2r	con+nlp
Tot number of docs over all queries						
Ret	50000	50000	50000	50000	50000	50000
Rel	2064	2064	2064	2064	2064	2064
RelRet	1413	1349	1390	1554	1610	1623
%chg		−5.2	−1.6	+10.0	+13.9	+14.9
(interp) Precision Averages at Recall						
0.00	0.5686	0.5276	0.5400	0.6438	0.6435	0.6458
0.10	0.4061	0.3685	0.3937	0.4453	0.4610	0.5021
0.20	0.3296	0.3054	0.3423	0.3567	0.3705	0.4151
0.30	0.2548	0.2373	0.2572	0.2935	0.3031	0.3185
0.40	0.2196	0.2039	0.2263	0.2480	0.2637	0.2720
0.50	0.1918	0.1824	0.2032	0.2189	0.2282	0.2379
0.60	0.1565	0.1596	0.1674	0.1834	0.1934	0.1899
0.70	0.1149	0.1167	0.1295	0.1444	0.1542	0.1571
0.80	0.0787	0.0854	0.0905	0.0949	0.1002	0.1163
0.90	0.0336	0.0368	0.0442	0.0338	0.0456	0.0434
1.00	0.0203	0.0228	0.0284	0.0154	0.0186	0.0158
Average precision over all rel docs						
Avg	0.1989	0.1884	0.2038	0.2238	0.2337	0.2466
%chg		−5.3	+2.5	+12.5	+17.5	+24.0
Precision at						
5 docs	0.3600	0.3160	0.3360	0.4040	0.4280	0.4440
10 docs	0.3120	0.3100	0.3240	0.3780	0.4000	0.4180
15 docs	0.2987	0.2813	0.2933	0.3333	0.3613	0.3800
20 docs	0.2680	0.2670	0.2790	0.3020	0.3260	0.3530
30 docs	0.2347	0.2240	0.2404	0.2560	0.2760	0.2993
100 docs	0.1356	0.1306	0.1412	0.1598	0.1708	0.1698
200 docs	0.0915	0.0865	0.0939	0.1051	0.1078	0.1107
500 docs	0.0488	0.0464	0.0489	0.0552	0.0575	0.0570
1000 docs	0.0283	0.0270	0.0278	0.0311	0.0322	0.0325
R-Precision (after Rel)						
Exact	0.2176	0.2196	0.2276	0.2435	0.2513	0.2820
%chg		+1.0	+4.2	+11.9	+15.5	+29.6

(1) txt1 = statistical terms only with ⟨desc⟩ and ⟨narr⟩ fields; (2) txt2 = same as txt1 but low-idf terms deleted from queries; (3) nyuir1r = txt2 plus syntactic phrases and similarities with ⟨desc⟩ and ⟨narr⟩ fields only; (4) con = statistical terms only using fields ⟨desc⟩ and ⟨con⟩; (5) nyuir2r = same as nyuir1r using fields ⟨desc⟩ and ⟨con⟩; and (6) con + nlp = nyuir2r repeated with the new weighting for phases.

to 12% (column 6 *con+nlp* in Table 4), but the results remain far below those for the ad hoc runs. Direct runs of queries 51-100 against the SJMN database produce results about the same as in the routing runs (which may indicate that our routing scheme works fine); however, the same queries run against the WSJ database have retrieval precision some 25% above SJMN runs. At this time we are unable to explain this discrepancy.‡

'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 *et al.*, 1993). Although such approaches are effective, they also tend to be costly because of increased index size and more complicated access methods.

‡Some problems with the SJMN database and the relevance judgments for it have been reported following TREC-2 evaluations.

Efficiency considerations have 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 the N highest-weight terms, where N can be the same for all queries or 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 having highest weights, will be considered when computing the document score. Moreover, only the global importance weights for terms are considered (such as *idf*), whereas local in-document frequency (e.g., *tf*) is suppressed by either taking a log or replacing it with a constant. The effect of this 'hot spot' retrieval is shown in Table 5 in the ranking of relevant documents within the top 30 retrieved documents for topic 72.

The final ranking is obtained by adding the scores of documents in 'regular' tf.idf ranking and in the hot-spot ranking. Although some of the recall may be sacrificed ('hot spot' retrieval has often lower recall than full query retrieval, and this becomes the lower bound on recall for the combined ranking), the combined ranking precision has been con-

Table 5. Ranks of the relevant documents in hot-spot retrieval and merged ranking for Topic 72

Documented ID	Rank	Score
Full log-*tf.idf* retrieval		
WSJ901228-0063	2	15957
WSJ910619-0153	3	15843
WSJ910322-0041	4	15063
WSJ880118-0090	7	13816
WSJ910102-0058	11	12803
WSJ870324-0083	12	12720
WSJ910916-0109	17	11014
WSJ910208-0191	18	10912
WSJ871013-0105	19	10745
WSJ910419-0071	21	10540
WSJ901227-0001	27	9928
WSJ900904-0093	28	9685
WSJ910215-0054	30	9609
Hot-spot idf-dominated with $N = 20$		
WSJ910916-0109	1	11822
WSJ910322-0041	2	11822
WSJ920226-0151	4	10016
WSJ901228-0063	6	9917
WSJ901227-0001	11	8704
WSJ870324-0083	12	8704
WSJ880127-0086	13	8704
WSJ910227-0107	14	7571
WSJ901227-0005	48	6754
WSJ900524-0125	51	6754
WSJ880118-0090	59	6754
WSJ911218-0028	61	6754
WSJ910719-0067	67	6754
Merged rankings		
WSJ910322-0041	1	15975
WSJ901228-0063	2	15060
WSJ910916-0109	3	13951
WSJ910619-0153	4	12745
WSJ870324-0083	6	12577
WSJ880118-0090	9	11732
WSJ920226-0151	11	11518
WSJ910102-0058	13	11225
WSJ901227-0001	16	11181
WSJ880127-0086	18	10871
WSJ910227-0107	23	9821
WSJ910419-0071	24	9811
WSJ871006-0091	37	8768

sistently better than in either of the original rankings: an average improvement is 10–12% above the tf.idf run precision (which is often the stronger of the two). The 'hot spot' weighting is represented with the α factor in the term weighting formula given in the previous section.

CONCLUSIONS

We presented in some detail our natural language information retrieval system consisting of an advanced NLP module and a 'pure' statistical core engine. Although 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 a 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. Will Rogers provided valuable assistance in installing updated versions of PRISE at NYU. We would also like to thank Ralph Weischedel, Heidi Fox, and Constantine Papageorgiou of BBN for providing and assisting in the use of the part-of-speech tagger. Most of the implementation of the indexing and search modules and their integration with the NLP module was done by Jose Perez Carballo. Mihnea Marinescu created updated versions of phrase extractor and structural disambiguator. Thanks to Ralph Grishman for his comments on an earlier version of this paper. Additional thanks to Alan Smeaton for detailed suggestions that helped to turn this into a more comprehensive article. 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 Contract N00600-88-D-3717 from PRC Inc., under ARPA's Tipster Phase-2 Contract 94-FI57900-000, and the National Science Foundation under Grant IRI-93-02615. We also acknowledge an early support from the Canadian Institute for Robotics and Intelligent Systems (IRIS).

REFERENCES

Broglio, J., & Croft, W.B. (1993). Query processing for retrieval from large text bases. *Proceedings of ARPA HLT Workshop*, March 21–24, Plainsboro, NJ.

Buckley, C. (1993). The importance of proper weighting methods. *Human language technology, Proceedings of the workshop*, pp. 349–352. Princeton, NJ: Morgan-Kaufmann.

Church, K.W., & Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics, 16*(1), 22–29.

Crouch, C.J. (1988). A cluster-based approach to thesaurus construction. *Proceedings of ACM SIGIR-88*, pp. 309–320.

Dillon, M., & Gray, A.S. (1983). FASIT: A fully automatic syntactically-based indexing system. *Journal of the American Society for Information Science, 34*(2), 99–108.

Evans, D., & Lefferts, R.G. (1994). Design and evaluation of the CLARIT-TREC-2 system. *Proceedings of the second Text REtrieval Conference (TREC-2)*, NIST Special Publication 500-215 (pp. 137–150), National Institute of Standards and Technology, Gaithersburg, MD.

Fagan, J.L. (1987). *Experiments in automated phrase indexing for document retrieval: A comparison of syntactic and non-syntactic methods*. Ph.D. Thesis, Department of Computer Science, Cornell University, Ithaca, NY.

Grefenstette, G. (1992). Use of syntactic context to produce term association lists for text retrieval. *Proceedings of SIGIR-92*, Copenhagen, Denmark, pp. 89–97.

Grishman, R., Hirschman, L., & Nhan, N.T. (1986). Discovery procedures for sublanguage selectional patterns: Initial experiments. *Computational Linguistics, 12*(3), 205–215.

Harman, D.K. (Ed.) (1993). *The first Text REtrieval Conference (TREC-1)*. NIST Special Publication 500-207, National Institute of Standards and Technology, Gaithersburg, MD.

Harman, D.K. (Ed.) (1994). *The second Text REtrieval Conference (TREC-2)*. NIST Special Publication 500-215, National Institute of Standards and Technology, Gaithersburg, MD.

Harman, D. (1988). Towards interactive query expansion. *Proceedings of ACM SIGIR-88*, pp. 321-331.

Harman, D., & Candela, G. (1989). Retrieving records from a gigabyte of text on a minicomputer using statistical ranking. *Journal of the American Society for Information Science*, *41*(8), 581-589.

Hindle, D. (1990). Noun classification from predicate-argument structures. *Proc. 28 Meeting of the ACL*, Pittsburgh, PA, pp. 268-275.

Kwok, K.L., Papadopoulos, L., & Kwan, K.Y.Y. (1993). Retrieval experiments with a large collection using PIRCS. *Proceedings of TREC-1 conference*, NIST Special Publication 500-207, pp. 153-172.

Lewis, D.D., & Croft, W.B. (1990). Term clustering of syntactic phrases. *Proceedings of ACM SIGIR-90*, pp. 385-405.

Liddy, E.D., & Myaeng, S.H. (1994). DR-LINK: A system update for TREC-2. *Proceedings of the second Text REtrieval Conference (TREC-2)*, NIST Special Publication 500-215, National Institute of Standards and Technology, Gaithersburg, MD, pp. 85-99.

Mauldin, M. (1991). Retrieval performance in ferret: A conceptual information retrieval system. *Proceedings of ACM SIGIR-91*, pp. 347-355.

Meteer, M., Schwartz, R., & Weischedel, R. (1991). Studies in part of speech labeling. *Proceedings of the 4th DARPA Speech and Natural Language Workshop* (pp. 331-336). San Mateo, CA: Morgan-Kaufmann.

Metzler, D.P., Haas, S.W., Cosic, C.L., & Wheeler, L.H. (1989). Constituent object parsing for information retrieval and similar text processing problems. *Journal of the ASIS*, *40*(6), 398-423.

Ruge, G. (1991). Experiments on linguistically based term associations. *Proceedings of RIAO'91*, Barcelona, Spain, pp. 528-545.

Ruge, G., Schwarz, C., & Warner, A.J. (1991). Effectiveness and efficiency in natural language processing for large amounts of text. *Journal of the ASIS*, *42*(6), 450-456.

Sager, N. (1981). *Natural language information processing*. Reading, MA: Addison-Wesley.

Salton, G. (1989). *Automatic text processing: The transformation, analysis, and retrieval of information by computer*. Reading, MA: Addison-Wesley.

Smeaton, A.F., & van Rijsbergen, C.J. (1988). Experiments on incorporating syntactic processing of user queries into a document retrieval strategy. *Proceedings of ACM SIGIR-88*, pp. 31-51.

Smeaton, A.F., & Sheridan, P. (1991). Using morpho-syntactic language analysis in phrase matching. *Proceedings of RIAO'91*, Barcelona, Spain, pp. 414-430.

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

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

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

Strzalkowski, T., & Vauthey, B. (1991). Fast text processing for information retrieval. *Proceedings of the 4th DARPA Speech and Natural Language Workshop* (pp. 346-351). Morgan-Kaufmann.

Strzalkowski, T., & Vauthey, B. (1992). Information retrieval using robust natural language processing. *Proc. of the 30th ACL Meeting*, Newark, DE, June-July, pp. 104-111.

Strzalkowski, T. (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, T. (1993a). Natural language processing in large-scale text retrieval tasks. *Proceedings of the first Text REtrieval Conference (TREC-1)*, NIST Special Publication 500-207, National Institute of Standards and Technology, Gaithersburg, MD, pp. 173-187.

Strzalkowski, T. (1993b). Robust text processing in automated information retrieval. *Proc. of ACL-sponsored workshop on Very Large Corpora*. Ohio State Univ., Columbus, June 22.

Strzalkowski, T., & Scheyen, P. (1993). Evaluation of TTP parser: A preliminary report. *Proceedings of International Workshop on Parsing Technologies (IWPT-93)*, Tilburg, Netherlands, and Durbuy, Belgium, August 10-13.

Strzalkowski, T., & Carballo, J.P. (1994). Recent developments in natural language text retrieval. *Proceedings of the first Text REtrieval Conference (TREC-2)*, NIST Special Publication 500-215, National Institute of Standards and Technology, Gaithersburg, MD, pp. 123-136.