Language-Oriented Information Retrieval

David D. Lewis, W. Bruce Croft, Nehru Bhandaru
Computer and Information Science Department
University of Massachusetts

COINS Technical Report 88-36

April 26, 1988

Language-Oriented Information Retrieval

David D. Lewis
W. Bruce Croft
Nehru Bhandaru
Computer and Information Science Department
University of Massachusetts, Amherst, MA 01003

April 26, 1988

Abstract

There is no task that computers regularly perform that is more affected by the nature of human language than the retrieval of texts in response to a human need. Despite this, the techniques actually in use for this task, as well as most of the techniques proposed by information retrieval (IR) researchers, make little use of knowledge about language. In this paper we take the view that IR is an inference task, and that natural language processing (NLP) techniques can produce text representations that enable more accurate inferences about document content. By considering previous work on language-based and knowledge-based techniques from this perspective, some clear lessons are apparent, and we are applying these lessons in the ADRENAL (Augmented Document REtrieval using NAtural Language processing) project. Our initial experiments with hand-coded representations suggest that using NLP-produced representations can result in significant performance increases in IR systems.

1 Introduction

Information retrieval (IR) is concerned with techniques for processing natural language texts in order to satisfy a need for information. We can categorize tasks fitting this description into: *classification* (sorting of texts into a fixed set of categories), *retrieval* (locating texts that are related to an arbitrary expression of interest), *extraction* (recognizing of information in text and representing of this information in a format with known semantics, such as relational database fields), *summarization* (producing a new textual representation with content related to that of the original), and *question-answering* (making use of an understanding of the text content in order to answer questions).

Classification and, particularly, retrieval are the traditional tasks that IR researchers have investigated. Extraction, summarization, and question-answering have been the focus of natural language processing (NLP) researchers. We will argue, however, that techniques developed for solving these latter problems have great relevance to IR.

The input to an IR system, called a *query*, is usually a textual description of the content of documents the user would like to see, such as:

Measurement of plasma temperatures in arc discharge using shock wave techniques.

The system compares the query to a large database of texts and outputs a list of document titles, abstracts, or full texts that the system has determined to be related to the user's interest. Usually this list is ranked in order of presumed relevance to the user.

One way to evaluate the performance of such a system is in terms of *precision* (what proportion of documents retrieved are relevant) and *recall* (what proportion of relevant documents are retrieved). Given a test collection consisting of queries, documents, and a set of judgments as to which documents are relevant to which queries, a *recall-precision curve* can be generated from a ranked list of documents. Such a curve shows how a user can, for instance, achieve greater recall at the cost of lower precision, by looking farther down in the ranking of retrieved documents. The curves in Figure 1 show the results of a classic experiment comparing manual indexing with full-text retrieval [1]. We will discuss these methods more in Section 2.2, but for now this graph serves to demonstrate how far IR techniques are from the ideal of 100% precision and recall.

The central task of IR can be viewed as inference [2], in particular the task of inferring whether concepts referred to in a query are also present in a document. Rather than using explicit inference rules, IR systems typically rely on implicit inferences in the form of *keyword matching* techniques. These techniques compute some measure of the similarity of the words in a query to those appearing in each document. Documents are then ranked according to this measure. Usually only the *content words* in the query and document are used in computing similarity. The words ignored are function words, such as prepositions and articles, which reflect little of the content of a particular document. The set of words, multi-word phrases, and other information that is chosen to represent the document in retrieval processing is called a *document representative*, and the process of producing such representatives (a subject of much research in IR) is called *indexing*.

The most effective of the keyword matching techniques is probabilistic retrieval [3]. In probabilistic techniques, the value assigned to a match between a document word and a query word depends on the word's statistical distribution both within a document and within a collection of documents. Like all keyword techniques, probabilistic retrieval makes relatively little use of the structure of human language. Put another way, probabilistic retrieval could be applied, essentially without change, to recognizing similarities between computer programs, road maps, or strands of DNA.

How can IR be improved by using knowledge of language? Consider one technique already used in IR systems: *stemming*. A stemming algorithm uses morphological information to remove suffixes from words, thus reducing variations on a root word to a single form or *stem*. Suppose we view keyword matching on words as making implicit use of the inferences "All instances of a word refer to related concepts" and "Any two different words refer to different concepts." Then keyword matching on word stems uses the inferences "All instances of a word with the same root word refer to related concepts" and "Any two words with different root words refer to different concepts." The fact that stemming improves performance somewhat means that the latter inferences are more accurate, though they are

still very fallible.

Stemming produces a *representation* of text which makes explicit one aspect of the morphological structure of words, and this allows more accurate inferences to be made. In Section 2 of this paper, we survey a variety of techniques, some using NLP and some not, that produce representations of text . The NLP techniques range from the use of simple syntactic templates for selecting indexing phrases, to natural language *parsers*, which produce detailed syntactic and semantic representations. The non-NLP techniques involve various sorts of content analysis by trained or untrained humans. The common feature of all the above techniques is that they produce representations that make explicit some information about the original text structure, thus allowing more powerful inferences.

The conclusions we draw from our survey of techniques are presented in Section 3. To summarize, we find that language-oriented techniques have so far achieved minimal success in improving IR performance. However, we believe that NLP and knowledge representation (KR) techniques originally developed for the more difficult tasks like extraction and question answering can be applied to retrieval. In particular, the use of natural language parsers producing symbolic representations of text content will allow much more accurate inferences to be drawn by IR systems. In Section 4 we discuss the architecture of the ADRENAL testbed, which we are constructing to test specific hypotheses arising from the above conclusions. In its final form, ADRENAL will allow the combining of traditional IR techniques with text skimming and parsing, and will take advantage of a core technical knowledge base and lexicon, of a large machine-readable dictionary, and of knowledge bases and thesauri for particular technical domains.

Constructing such a system will be a major undertaking, calling for some caution. Other reasons to proceed slowly in language- and knowledge-based IR are questions of efficiency, domain coverage, and the choice of appropriate KR methods. For this reason our initial work has stressed the use of hand-coded text representations and pre-existing parsing and KR tools. Experiments applying these to established test collections enable us to evaluate the performance improvements that can be expected, before we undertake a large scale implementation.

Our work to date is presented in Sections 5 and 6. First, we discuss an existing parser, TRUMP, which we are modifying to parse user queries from a test collection. We then discuss two experiments making use of queries that have been coded by hand into our REST (Representation for Science and Technology) system so as to simulate the eventual output of the parser. The first experiment uses these queries, combined with information from an on-line dictionary, to improve the performance of a probabilistic retrieval technique. The second experiment, which uses hand-coded representations of documents as well as queries, demonstrates that substantial improvements can result from applying simple matching techniques to frame-based text representations. These results support our belief in an NLP- and inference-oriented approach to IR, as well as pointing up the importance of careful consideration of the form of text representation used.

2 Previous Research

In the following section we divide research on applying knowledge of language to IR into four groups. The first two groups consist of NLP work on making explicit the syntactic or semantic structure of real-world text. We concentrate on techniques developed specifically for retrieval, but also examine other NLP research that has addressed real-world texts. A third group of work makes use of weak, nonstandard language analysis methods for concept recognition, combined with a strong emphasis on domain knowledge bases. Finally, we look at work on manual methods for representing text content for retrieval, both those that use traditional indexing languages, and those using KR languages.

2.1 Syntactic Analysis

Many researchers have attempted to make use of syntactic information in IR. The omission of function words (e.g. prepositions, articles, pronouns, etc.) when comparing documents and queries is a simple example of this. Slightly more complex are methods that use simple syntactic templates to recognize groups of words (*phrases*) on which to index a text. An example of such work is the FASIT system [4]. FASIT first tags words with their syntactic classes by using an on-line dictionary. It then indexes the text on those groups of consecutive words that fit one of several templates, such as <JJ NN> (adjective followed by noun). Experiments with a small test collection showed no improvement over a conventional statistical retrieval system, but FASIT's designers felt their technique had more potential for improvement than traditional methods.

Other systems using syntactic templates to choose indexing phrases are LEADER[5], AIR/PHYS [6], and the TMC Indexer [7]. Some researchers have even gone beyond templates to use a full natural language parser in selecting indexing phrases [8,9]. It has not been demonstrated, however, that any of these syntactic indexing techniques significantly improve retrieval performance. For instance, Fuhr and Knorz present results for 300 queries and 15,000 documents showing that AIR/PHYS gives results similar to that of manual indexing. This is not particularly encouraging, as we will see in Section 2.2.

The above work implicitly assumes that phrases with certain syntactic patterns refer to meaningful concepts, so that when such a phrase appears in both query and document, the two refer to some common concept. These schemes are limited by the fact that the phrase must appear in the same form in document and query in order for the concept to be matched. Since the same concept can be expressed by many different syntactic structures (e.g. "information retrieval", "retrieval of information"), these methods miss many matches. However, if a system has knowledge about what syntactic relationships express similar semantic relationships, then it can recognize superficially different occurrences of the same concept. In practice this knowledge is usually used first to select which phrases from text are meaningful, and then to perform *normalization* so that related phrases are all mapped to a single form.

A variant on normalization is to take information about what words in a query are related syntactically, and infer from this that occurrences of the words will be statistically dependent

in relevant documents. This information can then be used to improve the performance of statistical retrieval [10,11].

The most thorough examination of phrasal techniques was done by Fagan [12]. Fagan used both statistical and syntactic methods to produce normalized phrases for documents from several test collections. He found that any sort of phrasal indexing made a relatively small improvement in retrieval performance. His best result were in the range of a 20-25% improvement in average precision, but results were much worse on some test collections. Syntactic means of selecting phrases performed no better than statistical methods, though Fagan points out that there are many ways that his syntactic techniques could be improved. Our conclusion from examining the work by Fagan and others [13,14,15,16,17], is that these methods have limited potential for improving retrieval.

2.2 Semantic Analysis

The efficacy of syntactic normalization is limited by the fact that the same words must appear in both query and document to allow a match. To overcome this limitation, some use must be made of the meaning of the words. In IR, such semantic knowledge is usually introduced in the form of a *thesaurus*, which lists words which are related by synonymy, generalization, and similar relationships. This method goes beyond stemming to allow matches on words which have related meanings but unrelated surface forms. Thesauri have been shown to have some effectiveness in improving retrieval [18,19] and the available of large on-line thesauri [20] may increase the usefulness of this technique. A few studies, such as Salton and Lesk [18] and Fuhr and Knorz [6] have looked at combining thesauri with syntactic normalization, but resulting improvements have been marginal.

NLP approaches semantics from a different direction. Rather than representing meaning by some combination of the text's original words, NLP systems usually represent the text's meaning by data structures built using a knowledge representation language. Meaning is expressed in these languages in terms of categories, instances of categories, and relations between those instances and categories. A data structure commonly used by KR languages is the *frame*, a cluster of relations associated with a category or category instance. If we consider frames and other knowledge structures as playing the role in NLP that keywords do in IR, then *knowledge bases* play the role of a thesaurus, by representing the relationships between categories, category instances, and relations. Knowledge base designers typically use a much richer set of relationships between concepts than are used in thesauri. KR languages ease this task by providing support for inheritance, inference rules, error-checking, and other features. Brachman and Levesque [21] is a good introduction to KR languages.

Representations such as frames provide a sort of semantic normalization by mapping linguistic structures with similar meanings to similar representations. By reducing the ambiguity and sheer number of entities that must be referred to, these representations make it more practical to write inference rules which recognize connections between non-identical concepts and which infer the presence of unmentioned concepts. A rare example of applying NLP-produced semantic representations to text retrieval is a paper by Sparck Jones and Tait [22]. This work used a syntactic parser, aided by semantic expectations, to produce a

set of possible semantic frames for a query. Each frame was used to produce a number of rephrasings of concepts from the query, and all these phrases were combined into a single request to a retrieval system. This approach is similar to syntactic normalization, differing in that the semantic representation potentially generates a broader range of rephrasings. The results are encouraging, but based on too small a testset to allow conclusions to be drawn about the usefulness of the method.

The Sparck Jones and Tait work is worth noting for two reasons. First, unlike much work in NLP, their parser made use of a knowledge base designed for general text processing rather than one tailored to the particular domain (physics and electronics) of their texts. This is important since it suggests that large domain-specific knowledge bases are not necessarily needed to support a useful degree of parsing of technical prose. Secondly, this is possibly the only work on parsing into semantic representations which was oriented strictly toward retrieval rather than toward more complex tasks such as extraction, summarization, or question-answering. These latter tasks are less tolerant of inaccuracy and uncertainty than retrieval, forcing researchers to concentrate on narrow domains and unrealistically simple text, and making it difficult to evaluate the usefulness of the techniques to retrieval.

Still, any mapping of text into a form which abstracts away from syntactic structure and word choice, and which makes explicit some of the text's meaning, has great potential to aid retrieval. Work on TOPIC [23,24] has devoted some attention toward parsing for retrieval, though that is not the only focus of this work. The TOPIC parser makes use of expert procedures for handling particular syntactic structures, aided by semantic expectations from a 150 frame knowledge base on computer technology. The output is a set of frames which are linked to each other and to the original text passages that produced them. This representation is used for both summarization and retrieval. The NLP techniques used by TOPIC are relatively sophisticated, and the text handled is realistic and broad (magazine articles on computer technology), but unfortunately no empirical studies measuring the retrieval performance of the system have yet appeared.

Most other work on analyzing real world texts has focused on extraction or question-answering rather than retrieval, and has usually involved text from narrow domains, such as naval ship-to-shore messages [25], wild flower descriptions [26], medical reports [27], banking telexes [28,29], chemical reaction descriptions [30], news stories on corporate takeovers [31], and police reports [32]. A few have handled broader domains such as technical patent abstracts [33] or newswire stories [34]. Some of these systems have achieved impressive accuracy in their extraction of information, suggesting that if they were combined with inference rules making use of the extracted information for comparing queries to documents, their performance as retrieval systems might be impressive. One attempt to do this is FERRET [35], which uses the newswire story skimmer FRUMP [34] to analyze computer network digests. However, no performance data on FERRET have yet been published.

2.3 Knowledge-Based Concept Recognition

In the techniques discussed so far, knowledge about language, and automatic recognition of language structure, are important elements in recognizing when two pieces of text refer

to related concepts. Other methods, however, put their emphasis on the reasoning that is done once a symbolic representation of a concept is obtained, while using language-weak methods to perform concept recognition. Recognition is usually done by a large number of rules, each of which infers the presence of some concept based on the occurrence of word *patterns* in text. These patterns combine sets of related words (such as a thesaurus would have) with simple syntactic templates or specifications that certain words occur within the same sentence, paragraph, or other context. Such rules are less accurate than a full NLP parser, but they do not fail on difficult linguistic constructions, as many parsers do.

The prototypical example of such a system is RUBRIC [36,37]. Users of RUBRIC can define rules for recognizing concepts from word patterns or from the presence of other concepts. One important point made by Tong, et al. [36] about RUBRIC was that allowing rules to refer to other concepts, as well as to text patterns, made them easier to extend and modify. Traditional and nontraditional logical operators can be used in the rules, and research on RUBRIC has emphasized experimentation with inference and scoring methods to manage uncertainty in concept recognition. Very good retrieval performance has been reported on a collection of 730 newspaper stories, but no comparison was done with traditional methods [37]. This makes it hard to objectively evaluate RUBRIC's performance, since retrieval results vary widely between test collections.

Another rule-based retriever is the news story classification system reported by Hayes, et al. [38] This system makes use of more sophisticated linguistic patterns than does RUBRIC, but uses much simpler scoring methods. Its rules are essentially arbitrary LISP code, and must be written by programmers rather than users, so the system can recognize only a small set of categories. A test where 500 newswire stories were classified by the system with respect to 6 categories showed recall and precision of 93%—similar to that of the human categorizers the system was compared to.

Another use that has been made of pattern-matching concept identification rules is to perform an initial classification of a text, so that the correct set of category-specific NLP techniques can be used to extract information from it. Two examples of systems using this technique are FRUMP [34] and TESS [28].

2.4 Knowledge Representation without NLP

Even rules that recognize concepts in terms of word patterns can be complex to produce, so there is a long tradition of IR work which attempts to gain the benefits of associating text with semantic information without using any automated language analysis. As mentioned earlier, one method of doing this is to use a thesaurus, which enables non-identical words with related meanings to be matched. A more complex means of associating semantics with words is manual indexing using a *controlled vocabulary*. In these methods a human indexer produces a document representative for each piece of text. The representative is set of *index terms* (words and phrases), but semantic normalization is provided by requiring that all index terms be drawn from a fixed list. This contrasts with *full text indexing* which, since it uses all content words from text, is easy to automate but does not accomplish any semantic normalization. The more complex manual indexing schemes require the indexer

to indicate relationships between the terms as well. The relationships may be represented as explicit symbols (similar to slots or links in KR languages) or they may be implicitly encoded by requiring the indexer to first list a general subject class for the text, and then follow it by more specialized terms.

A number of studies have cast doubt on the efficacy of manual indexing. Figure 1 is taken from the discussion by Salton [1] of the classic Cranfield study [39] and shows the precision results from full text indexing exceeding those of manual indexing with a controlled vocabulary. Another early study which carefully evaluated manual indexing schemes was by Aitchison and Cleverdon [40]. This work compared the indexing and retrieval of metallurgical documents using the WRU Index Language for Metallurgy with the more general English Electric Faceted Classification for Engineering. The results, based on approximately 100 queries and 1000 documents, showed no advantage for the more domain-specific scheme. This argues against manual indexing as an effective means of using domain knowledge. Furthermore, the absolute performance figures for both systems (approximately 80% recall with 10% precision) were similar to those achieved by full text indexing.

Aitchison and Cleverdon report that many of the failures to retrieve relevant documents resulted from the human indexer's omission of important concepts, or from the system's failure to recognize matches between related concepts. Retrieval of non-relevant documents mostly resulted from matches on concepts that played only a minor role in a document, and from matches on query concepts that were expressed in too general terms. Results such as these have been used to argue that manual content analysis is not superior to full text indexing [1].

Questions about the efficacy of manual indexing schemes have not stopped their development. Some examples include SYNTOL [41], PRECIS [42], MeSH [43] and Relational Indexing [44,45]. Work on Relational Indexing even explored the use of inference rules to infer connections not explicitly represented in text [46], a technique usually found only in sophisticated NLP systems. Unfortunately, no performance evaluation was ever published on this feature.

Recently, work has come out of the AI community on representing text content for retrieval, question answering, and other tasks. These methods use KR languages rather than indexing vocabularies, and allow a much richer representation of the concepts in text. Applications include medical case reports [47], historical information [48], an artificial intelligence reference book [49], and text from Chemical Abstracts [50]. Carbonell, et al. compared KR languages to manual indexing for representing biomedical text content for retrieval, and came out in favor of frame-oriented KR languages [51].

Most of these systems have not been evaluated under controlled conditions. An exception is GRANT [52], which compares frame-based representations of research proposals and funding agencies in order to determine what agencies are likely to fund a proposal. The degree of match between two frames is based on heuristics which makes use of a large semantic net. GRANT achieved a recall rate of 67% at 29% precision (based on experts' judgments of what the appropriate funding agencies for a proposal were). This outperformed the simple keyword-based system it was compared against, and is about the same performance as the

best probabilistic keyword methods. Other encouraging results come from D. E. Lewis' dissertation, where a high correlation was shown between relevance of documents and partial matches on case frame representations of queries and document abstracts [53].

3 Lessons from Previous Research

Our survey provides both discouraging and encouraging evidence for the potential of language-oriented retrieval. On the minus side, NLP techniques for building semantic representations require large amounts of knowledge and have only been demonstrated in narrow domains. More broadly applicable techniques, such as syntactic normalization, provide relatively small performance improvements. Techniques such as concept recognition rules and controlled vocabulary indexing take considerable human effort and achieve uncertain performance. Little of the work on the more advanced language- and knowledge-oriented techniques has used test collections or been compared with established IR techniques, making competing claims hard to sort out.

However, we believe the following positive results have also been established:

- *Representations which abstract away from surface text can improve retrieval performance by making it easier to implement effective inferences.* The traditional IR inference technique—counting of matches between identical items—has been shown to be consistently improved by normalization techniques, such as stemming, thesauri, and phrasal indexing (though improvements vary between techniques). Work by Tong, Cohen, and others points toward the ability to state simple but powerful inference rules given a representation of text's semantic content. Note that one might take the failures of manual indexing to be an argument against semantic representations, since controlled vocabularies provide some of the same features as KR languages. However, since many problems with manual indexing stem from human inconsistencies, we instead conclude, as did Carbonell, et al. [51], that content analysis should be automated, and that richer semantic representations and inferences should be used.

- *Frames are a particularly good representation of document content for retrieval.* Manual indexing usually requires exact matches between indexing terms. Therefore, relevant documents expressing related, but not identical, concepts are missed. Many nonrelevant documents are retrieved, because they contain concepts which occur in the query but do not occur in the proper relationships with other concepts. By explicitly representing relationships between concepts as slot relations, and then grouping these slots in a frame for a concept, it is possible to allow partial matching, as well as matching which takes into account relationships between concepts. Early results on retrieval using partial matching of frame-based text representations have been encouraging [52,53]. Since these experiments made use of matches on only one slot per frame, there is potential (as Cohen and Kjeldsen concluded from their failure analysis) for considerably improving these methods by allowing matches on more than one relation at a time.

- *Current NLP techniques can provide useful analyses of real-world text.* Given some tolerance for error and ambiguity, meaningful semantic analysis can be done in constrained domains, while syntactic analysis is possible in any domain. The work by Sparck Jones and Tait suggests that even imperfect and ambiguous semantic representations have the potential to aid retrieval. The perception that current NLP techniques will necessarily fail on real-world text may apply only to more difficult tasks such as question-answering, and not to text retrieval.

- *The application of NLP to text retrieval is relatively unexplored in comparison with its application to other tasks.* Classification systems such as the one reported by Hayes, et al. use weak NLP techniques and achieve high performance on a relatively easy task. Text extraction systems achieve good performance on a very challenging task by using powerful NLP techniques in constrained domains. But little is known about the potential of powerful NLP techniques to improve text retrieval in broad domains, a potentially less difficult task.

4 ADRENAL

The ADRENAL testbed is our attempt to apply the lessons of the previous section to the retrieval of a wide range of scientific and technical documents. The general architecture of the system is shown in Figure 2 and the flow of processing is as follows: A natural language parser produces a frame-based representation of the user query. ADRENAL uses this representation in generating a keyword query to a traditional retrieval system. The top ranked documents from the keyword-based retrieval are shown to the user for feedback while also being given to a *text skimmer* [34]. The text skimmer uses fast NLP heuristics to scan for concepts from the query, and then reranks the retrieved documents, so that the worst can be discarded and the best shown to the user.

Documents which are not discarded, including those shown to the user, are passed on to a full NLP parser. The purpose of this parser is not to build a complete representation of the remaining documents, but rather to parse enough of each document to confirm or disconfirm the presence of query concepts. The parser will build frame representations for those sentences containing words that caused the document to score well, and a matching component will see to what extent concepts appear in the same relationships as in the user query.

During this entire process, decisions by the user as to which retrieved documents are relevant will result in changes in importance of the concepts being sought. This in turn may result in additional keyword queries being made to the traditional retrieval system, as well as resulting in changes in the concepts sought by the text skimmer and parser. The process will continue until the user is satisfied with the documents retrieved, and the system will compile statistics on its performance for later analysis.

Queries and documents will be parsed into frames drawn from REST (Representation for Science and Technology), our representation system. In its final form, REST will consist

of:

1. An inheritance hierarchy of frames representing events, objects, and relationships common to a range of scientific and technical fields.

2. A lexicon specifying mappings from a general scientific and technical vocabulary to the hierarchy of frames. Morphological, syntactic, and other information needed for parsing will be recorded.

3. A set of inference rules, which will be used to infer the presence of implicit concepts, as well as to determine when related concepts can be matched.

The output of our parser will be a *frame network* of REST frames linked by slot relations. One difference from usual NLP representations is that some of the original words from text will be present in the representation, along with the semantic abstractions the parser builds. This is because REST defines only high level scientific concepts, so words referring to domain-specific concepts must be kept to avoid losing information. Figures 3 and 4 show what REST representations of a typical query and document might look like. The version of REST used in our experiments to date, and in Figures 3 and 4, is described in Section 6 of this paper. The more comprehensive version of REST which is currently under construction is described in Lewis and Croft [54], but part of its inheritance hierarchy is shown in Figure 5.

As for the status of the system as a whole, the probabilistic retriever and the user interface are being adapted from the I^3R system [55]. A preliminary matcher for comparing query and document representations has been built, and an existing parser and on-line dictionary are being modified, as described in Section 5. A more sophisticated matching and inference system, along with a general control framework, are the main components that remain to be built before full-scale testing can begin. We expect to have an operational prototype by mid-1989.

4.1 Initial Hypotheses for the ADRENAL Project

There are a number of questions about processing methods for ADRENAL that we have not yet answered. How much filtering should each step (keyword-based system, text skimmer, NLP parser) do, and how does this interact with hardware speed and with varying user needs? To what extent can new concepts related to the user's interest be extracted from documents he or she judges to be relevant? How can ADRENAL take advantage of domain knowledge bases or user-supplied thesaurus information? The answers to these questions will emerge only after experimentation with a completed system.

Before attacking these larger questions, however, we believe it is important to first establish some more basic results. For instance, before constructing a large lexicon and knowledge base for REST, we want to know more about what characteristics text representations should have in order to aid retrieval. We also want to demonstrate some basic results on the parsing of user queries and on the role of traditional retrieval systems in language-oriented retrieval. In particular, we have the following hypotheses:

1. *Scoring and ranking documents based on similarity of frame-based representations of the document and query will provide a significant improvement over keyword techniques.* Previous research has suggested the effectiveness of inferring the relevance of a document based on comparing frame-based text representations of queries and documents. However, detailed comparisons of the effectiveness of different matching methods has not been done. Measures of similarity must be found which are efficient and can make use of incomplete representations of content.

2. *Representations, such as frames, which make explicit semantic categories and relations, will significantly improve retrieval, even if a only a small set of categories and relations are used. However, choices made about the structure of the representations will strongly affect the extent to which particular matching methods will improve performance.* The role of semantic abstraction in improving retrieval has been well-established, but little is known about how performance changes as knowledge bases grow, about how characteristics of a representation interact with matching and other inference methods, and about what characteristics are most important to text retrieval.

3. *NLP techniques can provide representations of documents and queries that will substantially improve retrieval.* We claim that current NLP techniques can build frame networks for texts that will allow useful, if partial, judgments of the similarity of texts' content. In particular, we believe:

   - Untrained users can be guided to state a query in such a way that a parser can produce a "complete" representation of it. Completeness here means not that the parser will capture all the subtle aspects of meaning, but rather that all the content words in the query will be captured in the frame network which is output.
   - NLP techniques can analyze real-world document texts with sufficient detail and accuracy to enable significant improvements in retrieval performance. A complete analysis is unlikely, due to limitations of both time and grammatical coverage, but we believe that even partial analyses will greatly improve retrieval performance.

4. *Keyword techniques can efficiently screen out many, though not all, of the documents that would be also be rejected by frame matching, and can do so with minimal loss of relevant documents.* If, for instance, a document contains none of the domain-specific words from the user query, as well as no words known to be related to those words, there is no need to subject it to NLP analysis. Such documents can, and should, be pruned by keyword-based methods.

It is important to stress that in proposing these rather conservative hypotheses we do not mean to suggest that more powerful NLP, KR, and reasoning techniques will not improve

performance much more. Our intent is simply to propose certain hypotheses that can be explicitly verified or falsified, and on which further work can be based.

The remainder of this paper describes some of our work testing these hypotheses. The next section describes the NLP techniques which we are investigating, while the last two sections describe initial experiments testing hypotheses 1 and 2.

5 NLP Techniques in ADRENAL

Our emphasis in the ADRENAL project is on adapting existing NLP techniques to the task of retrieval, rather than on developing new NLP techniques. In this way we hope to establish results that are more generally applicable than if we had relied on a specialized parser of our own design. We will also learn a great deal about what would be desirable in a specialized parser.

The parser we are currently working with is TRUMP [56]. TRUMP is a *chart parser* [57], which means that it uses a context-free grammar and has a tabular data structure, the "chart," for representing syntactic structures recognized during a parse. Such parsers are especially suited to dealing with real-world text, as their representation of partial constituents allows some information to be output even when a complete parse fails.

In addition to the basics of syntactic parsing, TRUMP provides a semantic interpretation mechanism which creates frame structures in the KR language KODIAK [58]. TRUMP also has a mechanism for ranking alternative parses by syntactic and semantic criteria, and has extensive support for phrasal lexicons [59]. Implementation of the REST representation system in KODIAK has been straightforward [54].

We are currently testing the ability of TRUMP to parse queries from the NPL collection of physics and electronics abstracts, a standard document retrieval testset [60]. Our current work with TRUMP focuses on extending the grammar to cover additional syntactic constructions, and on entering syntactic and semantic information for the words in the test queries.

Within the next few months we will establish how easily TRUMP can parse user queries. It will be particularly important to discover to what extent users must supply semantic information on domain-specific words in order to enable accurate parsing. Our hope is that very broad semantic judgments by the user will suffice, since we cannot expect untrained users to make fine-grained distinctions between REST categories.

One piece of future work will involve building an interface for TRUMP to make use of the on-line Longman Dictionary of Contemporary English (LDOCE). Once the large vocabulary from LDOCE is available, we will begin testing the parser's ability to produce partial representations of document abstracts from NPL and other test collections.

6 Experiments With REST

Our criterion for evaluating a parser is its ability to produce frame representations that improve text retrieval. So, before investing effort in building parsers, lexicons, and knowledge

bases, we would like to learn as much as possible about the characteristics of representations that are most important to retrieval performance. One way to do this is to hand-code representations of queries and documents similar to those that parsers might build, and then to perform retrieval experiments which vary the ways in which the representations are used. In this section we discuss two such experiments.

Both experiments make use of a standard test collection consisting of 3204 articles (represented by their abstracts and titles) from *Communications of the ACM*, along with a set of queries on computer science topics, and a set of judgments as to which articles are relevant to which queries. To maintain consistency and minimize bias introduced by human coders, a set of written guidelines was used when coding documents and queries [61]. Queries were coded by the first author (Lewis) and documents by the third author (Bhandaru). It is important to stress that we do *not* view hand coding of frame representations as a practical technique for information retrieval—it is merely a way to gain some insight into the kind of performance that an NLP analysis will eventually produce.

Both experiments used representations hand-coded in REST 0.3, an early version of REST using 8 categories, 13 types of relations between frames, and no inference rules. Figures 3 and 4 show the REST 0.3 frame networks for a hypothetical query and document. We say that any subset of frames from a frame network is a *subnetwork* if all the frames are connected by slots. If a subnetwork has a single outermost frame then that frame is the *root frame*.

REST 0.3 allows the following kinds of slots in frames:

- The *category* is a symbol drawn from the taxonomy (see Figure 6 and the Appendix) of REST 0.3 semantic classes.

- A frame can have any number of *relation* slots drawn from the list in the Appendix. Each represents a binary relation between the concept represented by the *containing frame* that the slot appears in, and the concept represented by the *filler frame*, a pointer to which is the slot filler. Each relation slot has an inverse so in Figure 4 we see the idea of one thing affecting another expressed by both AFF and AFF-BY, depending on which frame happens to end up as the containing frame and which as the filler frame.

- The *textname* slot indicates the syntactic head (loosely defined) of the linguistic structure that corresponds to the concept the frame represents. This could be one word or several, since compound nominals and similar structures were not broken up unless one of the words was in the REST vocabulary. There was also a special slot for acronyms, but for these experiments this was treated as another textname slot.

6.1 Enhancing Initial Probabilistic Ranking

One of our hypotheses is that keyword-based retrieval methods can save work for the NLP components of a language-based retrieval system by giving a preliminary indication of relevant documents, and by screening out obviously nonrelevant ones. The more accurate the

initial ranking produced by the keyword-based component, the more quickly the system will be able to present the user with relevant documents. The experiments in this section were designed to test the ability of information from REST representations of queries to improve such an initial document ranking.

In the strategy developed by Croft [11,62] from the probabilistic model of IR, an estimate is made of the probability that a document is relevant to a query. This estimate is based on the occurrence of keywords in both query and document, the prior probability of keywords appearing in relevant and nonrelevant documents, and the extent to which the occurrence of keywords in relevant documents is not statistically independent. Documents are then ranked according to this estimate. There are three ways that a REST representation of a query might improve such an estimate, without adding terms to the query or requiring REST representations of documents:

- The query used by the probabilistic strategy could be formed only from keywords that appear in the REST query, rather than from all the content words in the textual query. For instance, the word "information" would be present in the REST representation of "articles on information theory" but would not be present in the representation of "give me information on parallel processing." In a normal probabilistic retrieval "information" would be used in both cases.

- Vocabulary information could be used to estimate the probability that a query keyword occurs among the relevant documents. Usually this value is set to an arbitrary constant, but knowing which words were in our REST vocabulary or in a large on-line dictionary could enable better estimates of this probability.

- Finally, the correlation coefficient for groups of dependent terms in relevant documents can be estimated more accurately by using knowledge of which terms occur in the same frame subnetworks in the query.

To test these methods, REST 0.3 representations for 32 CACM queries were hand-coded and the information to use the above three methods was automatically generated from these representations. This information was generated by hand, without building REST representations, for another 18 queries. Table 1 shows the resulting recall and precision scores for this set of 50 queries, which were also used in the experiments in Croft (1986). The first column shows the scores from a probabilistic version of $tf \times idf$, one of the better keyword-based methods [12]. The next three columns show the effects of first eliminating terms not in the REST representation, then using vocabulary information, and finally using term dependency information. The average improvement in precision gained from combining all methods was 15.1%.

These experiments, which are described in more detail in Lewis, et al. [61], suggest the moderate, but useful, improvements that can be obtained by NLP analysis of queries alone. Since we explored, with some thoroughness, ways of making use of the above three types of information, we feel that the best that can be expected from these three methods is a

15-20% increase in average precision. We propose this upper bound only for these three particular methods, of course, and not for any of the others described in this paper.

There are more powerful methods which we have not yet tested for using a REST query representation to enhance a probabilistic retrieval. One is to produce multiple probabilistic queries from a single user query, one for each frame or frame subnetwork in the query representation. The resulting document rankings could be averaged or otherwise combined. This would address the problem of documents which receive high scores based on repeated matches with only a small part of a query. The different rankings could also be used to guide the NLP components' search for concepts. Another possibility is to use the REST hierarchy as a thesaurus in order to add additional REST vocabulary words to the query keywords. In general, query expansion using hierarchically related terms has not been shown to significantly improve retrieval [63], but it might be made more effective when combined with multiple querying.

Such methods are worth pursuing, since the better the results of the initial retrieval, the better the raw material for the NLP techniques . However, to achieve truly significant improvements, it will be necessary to build representations of document content as well, and match these against query representations. We turn to this issue in the next section.

6.2 Frame Network Matching

The primary purpose of the experiments in this section was to demonstrate the way in which decisions about how to represent text and how to compare representations of text interact with each other and with retrieval performance. A secondary goal was to show that certain techniques can in fact significantly increase performance, and to gain insight into why this is so and how these techniques can be extended.

Discovering these things required that we had representations for some set of relevant and nonrelevant documents. Hand-coding such representations for a full test collection of thousands of documents was out of the question, of course. Instead, for 28 of the 32 queries used in the experiments of Section 6.1, we hand-coded the top 10 documents retrieved by the $tf \times idf$ retrieval. The top 10 documents for the other 4 queries were not coded because either all 10 documents were relevant or all 10 were nonrelevant. Since the techniques we tested only altered the ranks of the top 10 documents, there was no way they could alter the retrieval performance of these 4 queries.

The remaining 280 documents still constituted a substantial amount of text (approx. 30,000 words), so the human coder attempted to observe the following limitations:

- Each keyword K in the text that matched some keyword in the query was potentially the source of a frame, F, and associated subnetworks.
- The coder made a tentative evaluation of what the document subnetworks containing F would look like. The only subnetworks that actually got coded were those that contained at least one other frame related to the query, and in particular satisfied one of the following two conditions:

1. If there were one or more other keywords from the query in the same document sentence as K, then the coder found the smallest linguistic structure such as a noun phrase, verb phrase, or sentence, that contained all the query keywords, and coded the document subnetwork for that structure.
2. Any single frame with slot filler F, which had the same category as a query frame that itself had a filler frame containing K was coded.

Figure 7 shows the representation that would be formed for the document in Figure 4 if the above rules were followed. Real documents typically had much more of their content omitted. As a measure of the amount of content omitted, 2791 keywords appeared in the 280 document frame networks, while 17,856 words appeared in the normal keyword representations of the same documents.

It should be pointed out that the selective representation used by our human coder was not just a sacrifice made for practicality. We would desire the same behavior from a parser—it should build representations of document text only to the extent that they are useful in evaluating how well the document's content matches that of the query. This means that decisions about what sentences or parts of sentences to parse should be based on knowledge of the matching method to be used in comparing document and query representations. In our case we planned to use matching methods based on evaluating the extent to which pairs of concepts were in the same relationships in document and query, and out of this came the rules for coding. The asymmetry of coding frames that matched only on category if they had a filler frame that matched on a keyword, but not vice versa, resulted from our intuition that general concepts should be matched only if they were modified in some way by a specific concept mentioned in the query.

6.3 Evaluation Criteria

Since the methods tested in this section all involved reranking only the 10 documents that were coded for each query, these methods made little change in the recall-precision curves, except at very high precision levels. Therefore, it was desirable to find other measures to compare methods on.

Since all the methods totaled up matches of some sort between query and document frame networks, the ratio of matches in relevant documents to matches in non-relevant documents was of interest was one point of comparison. We computed this ratio for each of the 28 queries for which documents were coded, and then took the median of these 28 values as a rough measure of how well the technique discriminated between relevant and non-relevant documents. (The median was used rather than the mean, since the denominator of some ratios was zero.) One disadvantage to this measure is that it isn't directly applicable to the $tf \times idf$ variants.

Another measure was to compute the Coefficient of Ranking Effectiveness [64] (CRE) for each set of 10 documents, and take the mean, MCRE, of this value across the 32 queries. CRE is defined as

$$\frac{m_r - \bar{R}}{m_r - m_p}$$

where m_r is the expected mean rank of relevant documents in a randomly ordered list, m_p is the expected mean rank of relevant documents in a perfectly ordered list, and $\bar{R}$ is the observed mean rank of relevant documents. CRE ranges between 1 (all relevant documents at the top of the ranking) and -1 (all relevant documents at the bottom of the ranking) and measures the deviation from randomness of a ranking. A high value means that relevant documents are being pushed higher in the set of 10 documents. The CRE of the top 10 documents is a somewhat deceptive measure when applied to the $tf \times idf$ retrievals, since, for instance, it is higher when no relevant documents appear in the top 10 than when several appear in the top 10, but at very low ranks. This measure does, however, give a sense of how well the varying methods improved the ranking of the top 10 documents over $tf \times idf$.

Finally, in order to give a rough sense of how the changes measured by the CRE translate into what a user sees, we computed the precision for the top 5 documents after reranking the top 10, and averaged this value over the 32 queries. The 4 queries where reranking could not be done were included in this computation, as they were in computing MCRE, in order to give a more accurate impression of how effective the method would be in a real retrieval setting, where the top 10 documents of some rankings might be too good or too bad to improve. However, since the precision of the top 5 documents is even more affected by chance fluctuations than the measures based on the top 10 documents, it should be interpreted with caution. (For instance, this measure shows $tf \times idf$ and the enhanced $tf \times idf$ scoring equally well in Table 2, while their precision figures in Table 1, based on hundreds of documents, show significant differences.)

6.4 Relation Triples

In general, methods for comparing the similarity of frame representations involve use of graph matching and other complex techniques. For these experiments, however, we wanted to explore more simple measures of comparison, which look only at small components of a frame network. The simplest component we can break a frame network into, such that information about relationships between concepts is preserved, is a *relation triple*. These have the form (S, containing-frame, filler-frame) where S is the name of the relation slot and the containing and filler frames are as defined earlier. We generated the complete set of such triples automatically from each frame network for a query or documents, and defined measures of similarity between frame networks based on their number of matching triples.

Each similarity measure required defining what constituted a match between two relation triples. By changing this definition we were able to test the effects of various representation choices. The basic matching strategy defined two triples as matching if 1) their containing frames had the same category, 2) there was a textname match, as defined below, between their filler frames, and 3) their slot types matched. Slot types were considered to match if they were identical, and in addition REL and ARG were allowed to match each other (since

both were ways of specifying that a relation was not covered by our set of slots), and SUB and HAS-TEXT were allowed to match any other slot since they had very general semantic content. A textname match between two frames simply meant that the textname slots of the two frames had some (stemmed) word in common.

Ranking the 10 documents for each query on the number of the matches they had with relation triples in the queries, and breaking ties according to the $tf \times idf$ rankings, gave the results shown under Basic Match in Table 2. We see a considerable improvement over both the $tf \times idf$ and enhanced $tf \times idf$ measures. Clearly some aspect of representing document content aided retrieval performance.

We investigated several variations on the basic matching scheme. One was to recognize that users are likely to prefer documents that match several of the concepts mentioned in their query over ones that match the same concept multiple times. This was easily tested by allowing each component of the query network to be matched only once. Query Coverage, in Table 2, shows the results for this approach, which surprisingly are essentially the same as those for the basic method, which does count multiple matches.

This does not necessarily mean that our theory about user preferences is wrong. The total number of matches on triples under Basic Match was 427, while the total number when allowing each query triple to be matched only once was 296, giving only 131 more matches, distributed rather unevenly over 280 documents, for the basic method. This means the two methods assigned essentially the same scores to most documents. This is not really strange, considering the relatively small amount of text coded in comparison to the very large number of triples that could potentially be formed. The theory about user preferences could only be adequately tested by using much larger pieces of text, or by using thesauri or knowledge bases that allow non-identical keywords to match.

We can also vary not just how matches are counted, but what triples are considered to match. The line for All Slots Match in Table 2 shows the results of altering the basic method by no longer requiring slot types to match. The fact that this reduces performance considerably on all measures suggests that the slot types are in fact capturing meaningful differences between the various relations expressed in text. This is evidence in favor of using representations, such as frames, that make these relations explicit.

A similar experiment to test the importance of frame categories gave quite a different result. Hierarchical Match in Table 2 shows the results if categories of containing frames are allowed to match their child or parent categories, and All Categories Match shows the results if any two containing frame categories are allowed to match. Comparing this with the basic method, which requires the containing frames to have identical categories, shows essentially no difference.

Should we take this as evidence that distinguishing between categories for frames is not important? Probably not, since for most documents all the slot matches in the Any Category case actually had identical containing frame categories. The human coder's guidelines meant that most document networks produced had this property, the exceptions being those that contained two keywords from the query. So again, the representation we built did not allow adequate testing of a theory.

Another manifestation of the influence of coding decisions was found by changing which matching criterion was required for which frame. Our baseline method requires a category match on the containing frame and a textname match on the filler frame. We could equally well do things the other way around, requiring a category match on the filler frame and a textname match on the containing one, or we could allow the match to work either way, as long as there was at least one of each on different frames. These two alternatives are presented as Reversed Match and Symmetric Match in the table.

By now it should be no surprise that the results in the Reversed case are quite bad. The reason, of course, is that there were very few Reversed matches in either relevant or non-relevant documents, since our coder was instructed not to represent networks with this structure unless two keywords were present. Symmetric Match comes out slightly better than Basic Match, providing further evidence that the problem with Reversed Matches was less that they were ineffective, than that they were infrequent. This is what we would hope, if our coder was using each relation slot S in such a way that triple (S,A,B) had the same meaning as (inverse-S,B,A).

Just how important the influence of selective coding was is shown by the performance of Textname Match in Table 2. This method counts the number of matches between frames rather than triples, simply allowing two frames to match if their textname slots have a stemmed word in common. This is essentially a simple form of keyword matching making some use of phrasal items. Despite the well-established fact that this version of keyword matching is inferior to methods using probabilistic information, such as the variants of $tf \times idf$ tried above, we see Textname Match outperforming these methods and performing essentially the same as the methods which match on triples.

This suggests that much of the effectiveness of both relational triple matching and textname matching came less from the particular matching scheme used, than from the selectivity of document coding. Recall that only about a sixth as many keywords appeared in the frame networks as appeared in the normal keyword representations used for the $tf \times idf$ retrievals. Most of the keywords discarded did not occur in the query, and so would not have been taken into account under $tf \times idf$ anyway. Others, however, did occur in both document and query, but were discarded because they did not have one of the required connections, in that document, with another keyword or category from the query. The result was that when keywords matched under the Textname Match scheme they were much more likely to have the same meaning, in query and document,, than when they matched under the various $tf \times idf$ schemes.

Our original rationale for selective parsing (and selective human coding) was as an efficiency measure—there was no reason to produce a representation of a particular document text fragment if the parser could tell in advance that that representation would not match any part of the query representation. The effectiveness of Textname Matching suggests an additional possible benefit. For simple matching schemes such as used here, it may be possible to achieve most of the performance that the matching scheme would have on full document representations by instead using the matching scheme to limit parsing, and then doing keyword matching. However, it seems likely that AI frame matching methods which

take into account larger pieces of frame structure would not be well approximated by this method.

6.5 Discussion of Results

The above experiments succeeded at two things. The obvious one was to provide an initial test of our hypotheses that retrieval performance can be improved by using explicit semantic representations, and in particular by techniques that measure the similarity of frame-based representations of queries and documents. The benefits of semantic representation were demonstrated by the various ways in which REST representations were used to enhance $tf \times idf$ retrievals. The superiority of restricting relation triple matches according to slot type also provides evidence for the importance of distinguishing between semantic classes, at least for relations. Both textname matching and relation triple matching improved retrieval performance, showing the usefulness of ranking documents based on frame similarity measures. All these improvements were demonstrated using representations of queries and documents similar to those we might expect to get from a natural language parser. These experiments, however, are based on a quite small set of hand-coded documents, and can only be verified by using automated analysis of larger collections.

It must be strongly emphasized that these results do *not* in any sense provide an upper limit on the performance of language-oriented retrieval using document analysis, just because the representations were hand-coded. The set of frame categories in REST is currently being greatly expanded beyond those used in these experiments, inference rules are being added, and the particular choices of slot types are being experimented with. A robust parser will be able to analyze many more documents per retrieval than we were able to analyze by hand. The above experiments did not use domain knowledge bases or lexicons, which will undoubtedly enhance performance, as will the use of full-text documents rather than abstracts. More sophisticated frame matching methods, already developed by AI researchers, need to be tried. Finally, methods of plausible inference should be found to combine evidence provided by frame matching with that produced by probabilistic retrieval and other methods. So the results in this paper give only a hint of the potential of language-oriented retrieval.

The second thing we wanted to do with these experiments was to demonstrate the strong influence that representation choices have on what retrieval methods are appropriate and on what performance they achieve. These choices range from whether documents are represented by abstracts or full text, to what parts of a text are translated into a KR language, to what slot and frame categories are used. In particular, we saw how choices made in human coding of the documents strongly interacted with what matching techniques were appropriate for the resulting frame network representations. There was also evidence that anticipating our simple matching technique (relation triple matching) in selecting the parts of text to represent allowed an even simpler technique (keyword matching) to be improved.

The interactions between text analysis and matching techniques will not go away when a parser is used rather than a human coder. For the sake of efficiency it will be desirable to

parse as little of a document as possible, and we will have to take into account the matching techniques used in deciding what text can be skipped. In addition to this explicit selectivity, any parser will implicitly introduce biases into the representations it builds, based on the parsing technique it uses, the coverage of its grammar, and the design of its knowledge representation language. These biases will interact with matching in unexpected ways, just as the explicit biases we gave our human coder did. These interactions are likely to be exacerbated with the use of inference techniques more complicated than matching, such as ones which infer unrepresented concepts.

There are two lessons to draw from this. The first is that the interaction of representation choices and parser strategies with matching and other inference methods needs to be considered early in the design of a system. The second lesson is that language-oriented retrieval makes the IR tradition of evaluating techniques on realistic testsets more important than ever.

7 Conclusion

We have argued in this paper that there is considerable potential for enhancing the performance of text retrieval systems by making use of developments in natural language processing and knowledge representation. When ADRENAL is completed it will provide an example of how traditional IR techniques can combined with NLP to achieve increased performance. The results in this paper demonstrate the potential of such a system, and also argue for the careful testing, on realistic data, of decisions on representation and parsing. We encourage and look forward to more research of this kind.

Acknowledgments

This research was sponsored in part by the Office of Naval Research under University Research Initiative Grant N0014-86-K-0764, by NSF grant IST-8414486, and by a NSF Graduate Fellowship. We would also like to thank Scott Anderson, Bob Krovetz, and Kishore Swaminathan for reading previous drafts of this paper. All responsibility for remaining errors lies with the authors.

References

[1] G. Salton. Another look at automatic text-retrieval systems. *Communications of the ACM*, 29(7):648–656, July 1986.

[2] C. J. van Rijsbergen. A non-classical logic for information retrieval. *The Computer Journal*, 29(6):481–485, 1986.

[3] C. J. van Rijsbergen. *Information Retrieval.* Butterworths, London, second edition, 1979.

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

[5] D. J. Hillman and A. J. Kasarda. The LEADER retrieval system. In *Spring Joint Computer Conference*, pages 447–455, AFIPS, 1969.

[6] N. Fuhr and G. E. Knorz. Retrieval test evaluation of a rule based automatic indexing (AIR/PHYS). In C. J. van Rijsbergen, editor, *Research and Development in Information Retrieval: Proceedings of the Third Joint BCS and ACM Symposium*, pages 391–408, Cambridge University Press, Cambridge, July 2-6 1984.

[7] P. M. Mott, D. L. Waltz, H. L. Resnikoff, and G. G. Robertson. *Automatic Indexing of Text.* Technical Report 86-1, Thinking Machines Corporation, January 1986.

[8] L. L. Earl. Experiments in automatic extracting and indexing. *Information Storage and Retrieval*, 6:313–324, 1970.

[9] D. DeJaco and G. Garbolino. An information retrieval system based on artificial intelligence techniques. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 214–220, 1986.

[10] A. F. Smeaton. Incorporating syntactic information into a document retrieval strategy: an investigation. In *ACMSIGIR Conference on Research and Development in Information Retrieval*, pages 103–113, 1986.

[11] W. B. Croft. Boolean queries and term dependencies in probabilistic retrieval models. *Journal of the American Society for Information Science*, 71–77, 1986.

[12] J. L. Fagan. *Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and Non-Syntactic Methods.* PhD thesis, Department of Computer Science, Cornell University, September 1987.

[13] G. Salton. *Automatic Information Organization and Retrieval.* McGraw-Hill Book Company, New York, 1968.

[14] Y. Chiaramella and B. Defude. A prototype of an intelligent system for information retrieval: IOTA. *Information Processing and Management*, 23(4):285–303, 1987. Special Issue on Artificial Intelligence and Information Retrieval.

[15] M. Bruandet. Outline of a knowledge base model for an intelligent information retrieval system. In C. T. Yu and C. J. van Rijsbergen, editors, *Tenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 33–43, June 1987.

[16] C. Berrut and P. Palmer. Solving grammatical ambiguities within a surface syntactical parser for automatic indexing. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 123–130, 1986.

[17] G. Thurmair. A common architecture for different text processing techniques in an information retrieval environment. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 138–143, 1986.

[18] G. Salton and M. E. Lesk. Computer evaluation of indexing and text processing. *Journal of the Association for Computing Machinery*, 15(1):8–36, 1968.

[19] E. A. Fox. Lexical relations: enhancing effectiveness of information retrieval systems. *ACM SIGIR FORUM*, XV(3):5–36, 1980.

[20] E. A. Fox, J. T. Nutter, T. Ahlswede, M. Evens, and J. Markowitz. Building a large thesaurus for information retrieval. In *Second Conference on Applied Natural Language Processing*, pages 101–108, February 1988.

[21] R. J. Brachman and H. J. Levesque, editors. *Readings in Knowledge Representation.* Morgan Kaufmann Publishers, Inc., Los Altos, California, 1985.

[22] K. Sparck Jones and J. I. Tait. Automatic search term variant generation. *Journal of Documentation*, 40(1):50–66, March 1984.

[23] U. Hahn and U. Reimer. *TOPIC Essentials.* Technical Report TOPIC-19/86, Universitat Konstanz, Konstanz, April 1986.

[24] U. Thiel and R. Hammwohner. Informational zooming: an interaction model for the graphical access to text knowledge bases. In C. T. Yu and C. J. van Rijsbergen, editors, *Tenth Annual International ACMSIGIR Conference on Research and Development in Information Retrieval*, pages 45–56, june 1987.

[25] R. H. Granger, C. J. Staros, G. B.Taylor, and R. Yoshii. Scruffy text understanding: design and implementation of the nomad system. In *Conference on Applied Natural Language Processing*, pages 104–106, February 1987.

[26] J. R. Cowie. Automatic analysis of descriptive texts. In *Conference on Applied Natural Language Processing*, pages 117–123, February 1983.

[27] N. Sager. *Natural Language Information Processing: A Computer Grammar of English and Its Applications.* Addison-Wesley, Reading, Massachusetts, 1981.

[28] S. R. Young and P. J.Hayes. Automatic classification and summarization of banking telexes. In *The Second Conference on Artificial Intelligence Applications*, pages 402–408, IEEE Computer Society, December 1985.

[29] S. Lytinen and A. Gershman. ATRANS: automatic processing of money transfer messages. In *AAAI-86*, pages 1089–1093, 1986.

[30] L. H. Reeker, E. M. Zamora, and P. E. Blower. Specialized information extraction: automatic chemical reaction coding from english descriptions. In *Conference on Applied Natural Language Processing*, pages 109–116, 1983.

[31] L. F. Rau and P. S. Jacobs. Integrating top-down and bottom-up strategies in a text processing system. In *Second Conference on Applied Natural Language Processing*, pages 129–135, February 1988.

[32] D. Allport. The TICC: parsing interesting text. In *Second Conference on Applied Natural Language Processing*, pages 211–218, February 1988.

[33] F. Nishida and S. Takamatsu. Structured-information extraction from patent-claim sentences. *Information Processing and Management*, 18(1):1–13, 1982.

[34] G. DeJong. *An Overview of the FRUMP System*, pages 149–176. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1982.

[35] M. Mauldin, J. Carbonell, and R. Thomason. Beyond the keyword barrier: knowledge-based information retrieval. In *29th Annual Conference of the National Federation of Abstracting and Information Services*, Elsevier Press, 1987.

[36] R. M. Tong, L. A. Appelbaum, V. N. Askman, and J. F. Cunningham. Conceptual information retrieval using RUBRIC. In C. T. Yu and C. J. van Rijsbergen, editors, *Tenth Annual International ACMSIGIR Conference on Research and Development in Information Retrieval*, pages 247–253, june 1987.

[37] R. M. Tong and L. A. Appelbaum. Conceptual information retrieval from full-text. In *RIAO 88 : User-Oriented Content-Based Text and Image Handling*, pages 899–909, 1988.

[38] P. J. Hayes, L. E. Knecht, and M. J. Cellio. A news story categorization system. In *Second Conference on Applied Natural Language Processing*, pages 9–17, February 1988.

[39] C. W. Cleverdon and E. M. Keen. *Aslib-Cranfield Research Project. Vol. 2, Test Results.* Technical Report, Cranfield Institute of Technology, Cranfield, England, 1966.

[40] J. Aitchison and C. Cleverdon. *A Report on the Test of the Index of Metallurgical Literature of Western Reserve University.* Technical Report, The College of Aeronautics, Cranfield, Cranfield, England, October 1963.

[41] J. C. Gardin. *SYNTOL.* Volume II of *Rutgers Series on Systems for the Intellectual Organization of Information*, Graduate School of Library Service, Rutgers, 1965.

[42] D. Austin. Precis in a multilingual context. *Libri*, 26(1):1–37, 1976.

[43] D. B. McCarn. Medline: an introduction to on-line searching. *Journal of the American Society for Information Science*, 31(3):181–192, May 1980.

[44] J. Farradane. Relational indexing. part I. *Journal of Information Science*, 1:267–276, 1980.

[45] J. Farradane. Relational indexing. part II. *Journal of Information Science*, 1:313–324, 1980.

[46] J. Farradane, J. M. Russell, and P. A. Yates-Mercer. Problems in information retrieval: logical jumps in the expression of information. *Information Storage and Retrieval*, 9:65–77, 1973.

[47] G. D. Rennels, E. H. Shortliffe, F. E. Stockdale, and P. L. Miller. *Reasoning from the Clinical Literature: The Roundsman System.* Memo KSL-86-5, Medical Computer Science Group, Stanford University School of Medicine, January 1986.

[48] G. P. Zarri. *Some Remarks about the Inference Techniques of RESEDA, An "Intelligent" Information Retrieval System*, pages 281–300. Cambridge University Press, Cambridge, July 1984.

[49] R. F. Simmons. A text knowledge base from the AI handbook. *Information Processing and Management*, 23(4):321–339, 1987. Special Issue on Artificial Intelligence and Information Retrieval.

[50] D. Krawczak, P. J. Smith, and S. J. Shute. EP-X: a demonstration of semantically-based search of bibliographic databases. In *Tenth Annual International ACMSIGIR Conference on Research and Development in Information Retrieval*, pages 263–271, 1987.

[51] J. G. Carbonell, D. A. Evans, D. S. Scott, and R. H. Thomason. On the design of biomedical knowledge bases. In R. Salamon, B. Blum, and M. Jorgenson, editors, *Medinfo86: Proceedings of the Fifth Conference on Medical Informatics*, pages 37–41, Elsevier Science Publishers, Amsterdam, 1986.

[52] P. R. Cohen and R. Kjeldsen. Information retrieval by constrained spreading activation in semantic networks. *Information Processing and Management*, 23(4):255–268, 1987. Special Issue on Artificial Intelligence and Information Retrieval.

[53] D. E. Lewis. *Case Grammar and Functional Relations in Aboutness Recognition and Relevance Decision-Making in The Bibliographic Retrieval Environment.* PhD thesis, School of Library and Information Science, Faculty of Graduate Studies, The University of Western Ontario, July 1984.

[54] D. D. Lewis and W. B. Croft. Frame-based representation for information retrieval. In preparation.

[55] W. B. Croft and R. Thompson. I^3R: a new approach to the design of document retrieval systems. *Journal of the American Society for Information Science*, 38(6), 1987.

[56] P. S. Jacobs. Language analysis in not-so-limited domains. In *Fall Joint Computer Conference*, Dallas, TX, 1986.

[57] J. Allen. *Natural Language Understanding.* The Benjamin/Cummings Publishing Company, Inc., Menlo Park, California, 1987.

[58] R. Wilensky. *Some Problems and Proposals for Knowledge Representation.* Technical Report UCB/CSD87/351, EECS, University of California, Berkeley, may 1987.

[59] D. J. Besemer and P. S. Jacobs. FLUSH: a flexible lexicon design. In *25th Annual Meeting of the Association for Computational Linguistics*, pages 186–192, 1987.

[60] K. Sparck Jones and C. J. van Rijsbergen. Information retrieval test collections. *Journal of Documentation*, 32(1):59–75, 1976.

[61] D. D. Lewis, N. Bhandaru, and W. B. Croft. *Experiments with Frame-Based Text Retrieval.* Technical Report, Computer and Information Science Department, University of Massachusetts at Amherst, 1988.

[62] W. B. Croft. Document representation in probabilistic models of information retrieval. *Journal of the American Society for Information Science*, 451–457, November 1981.

[63] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill Book Company, New York, 1983.

[64] M. McGill, M. Koll, and T. Noreault. *An Evaluation of Factors Affecting Document Ranking by Information Retrieval Systems.* Technical Report, School of Information Studies, Syracuse University, Syracuse, New York, October 1979.

Appendix: REST 0.3 Categories and Relation Slots.

There were 8 categories of frames in REST 0.3, the version used for the experiments in the paper. As can be seen, the distinctions between them are both broad and vague:

1. ACTION: All actions, but actually used only to represent those not covered by ALTER, BUILD, or GET-INFO.
2. ALTER: An ACTION whose main purpose is to change something else.
3. BUILD: An ACTION whose main purpose is to bring something into existence.
4. GET-INFO: An ACTION whose main purpose is to produce information about something.
5. FIELD: A field of study, like "engineering," "biochemistry," etc. A subclass of STRUCTURE.
6. METHOD: A body of information used in controlling actions. A subclass of STRUCTURE.

7. RELATIONSHIP: A connection of any sort between other frames. A RELATIONSHIP frame was used when the actual words used to indicate the relationship were domain words that should be kept in the representation. Otherwise, one of the slot relations, which did not allow keeping the original words, was used.

8. STRUCTURE: Essentially anything that wasn't an ACTION or a RELATIONSHIP. Both solid stuff and abstractions, but not used if FIELD or METHOD applied.

The 13 relation slots and their inverses used in REST 0.3 are listed below. When reference is made to slot fillers or containing frames, it is with respect to the first slot in each pair.

1. ACTOR/ACTOR-IN: Used only in ACTION and its children. The filler is an entity, usually a STRUCTURE, which plays some active role in making the ACTION happen.

2. AFF/AFF-BY: Indicates that some entity is affected by another.

3. ARG/ARG-OF: Used only in RELATIONSHIP frames. Merely indicates that the filler takes part in the RELATIONSHIP.

4. HAS-PART/PART-OF: Used to represent that an entity (usually a STRUCTURE) is composed of other entities.

5. HAS-TEXT/TEXT-OF: Indicates that the filler frame has a TEXTNAME or ACRONYM slot, all of whose words are found in a TEXTNAME or ACRONYM slot of the containing frame. Not used if there was any other connection between the frames.

6. IN-CONTEXT/CONTEXT-OF: A rather general slot used when there's a hierarchical relationship too complicated for SUB, HAS-PART, or USES to apply. Includes things like being used by an unmentioned part of a complex structure, being a piece of knowledge in a scientific field, etc.

7. INTEREST/INTEREST-OF: Used only with GET-INFO frames. Similar to STUDIED; the distinction is that INTEREST is what you really want to know, while STUDIED is what you look at to find that out.

8. REL/REL: A very general slot used when some relationship between two items was specified in text, but none of the other slots applied.

9. RESULT/RESULT-OF: Things brought into existence by ACTIONs.

10. STUDIED/INV-STUDIED: Used only with GET-INFO frames. Indicates the entity that information is being produced on.

11. SUB/SUPER: More specific/more general. Our name for "ISA," "AKO," etc.

12. SAME/SAME : Used when two frames represent concepts with identical meanings. This most often occurred because a piece of text introduced a term and gave a definition for it.

13. USES/USED-BY: Relationships that might reasonably be expressed by the English word "use".

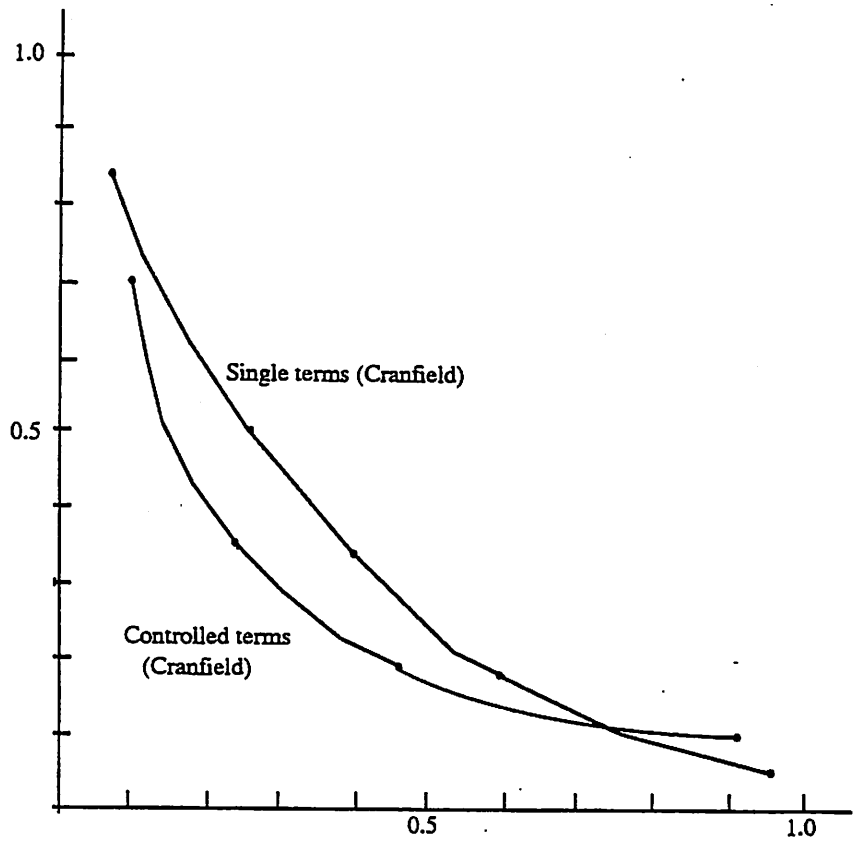

Figure 1: Recall-precision results from the Cranfield study.

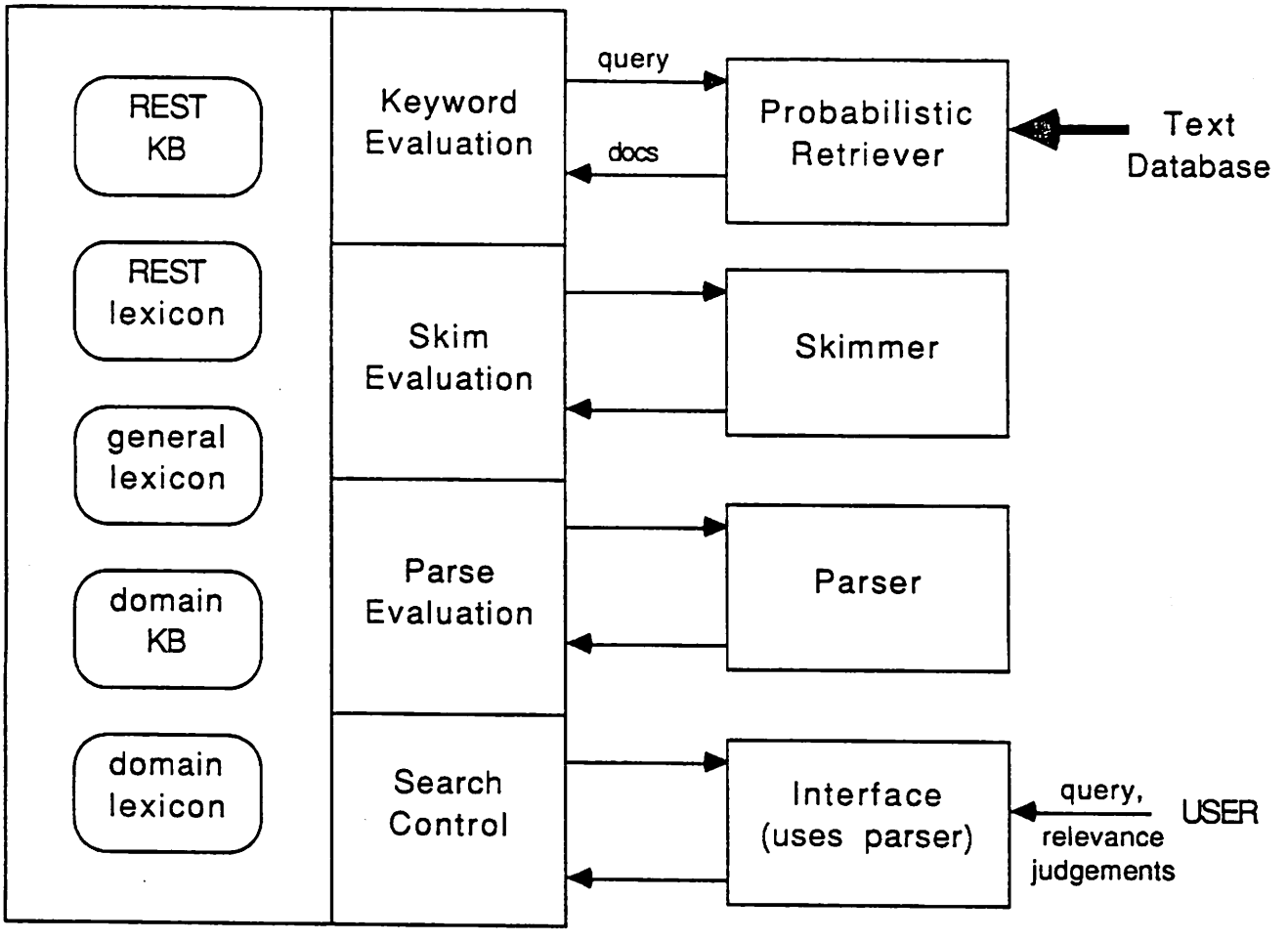

Figure 2: Architecture of the ADRENAL testbed.

```
GET-INFO
  TEXTNAME: research
  STUDIED: ALTER
                TEXTNAME: failures
                AFF: STRUCTURE
                        TEXTNAME: microprocessors
                        HAS-PART: STRUCTURE
                                        TEXTNAME: multipliers
                ACTOR: STRUCTURE
                          TEXTNAME: cosmic rays
                          ARG-OF: RELATIONSHIP
                                          TEXTNAME: high energy
                AFF: STRUCTURE
                        TEXTNAME: memory chips
```

Figure 3: REST 0.3 representation for the query: "*I'm interested in research on failures in memory chips caused by high energy cosmic rays. Also in microprocessors containing multipliers.*"

```
GET-INFO
  TEXTNAME: study
  STUDIED: STRUCTURE
                TEXTNAME: sensors
                ARG-OF: RELATIONSHIP
                              TEXTNAME: optical
                USING: STRUCTURE
                              TEXTNAME: microprocessors
                              HAS-PART: STRUCTURE
                                            TEXTNAME: silicon
                AFF-BY: ALTER
                              TEXTNAME: failures
                IN-CONTEXT: ACTION
                                            TEXTNAME: energetic reaction
BUILD
  TEXTNAME: developed
  AFF: METHOD
          TEXTNAME: protective coating
          ARG-OF: RELATIONSHIP
                        TEXTNAME: highly effective
```

Figure 4: REST 0.3 representation for the document abstract: "*This paper presents a study of optical sensors using silicon microprocessors which failed in the presence of an energetic reaction. A highly effective protective coating we have developed is also discussed.*"

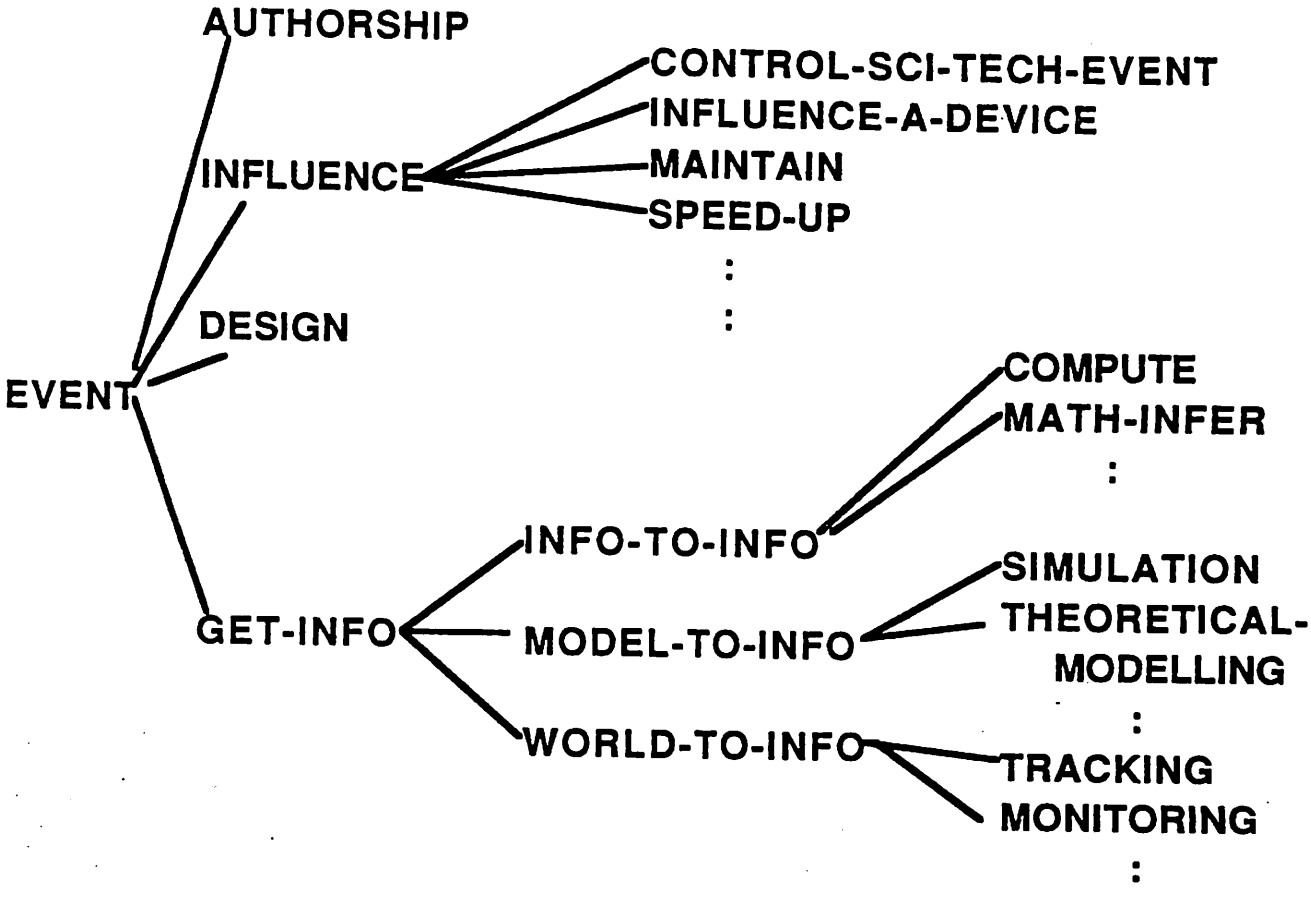

Figure 5: Parts of current REST knowledge base.

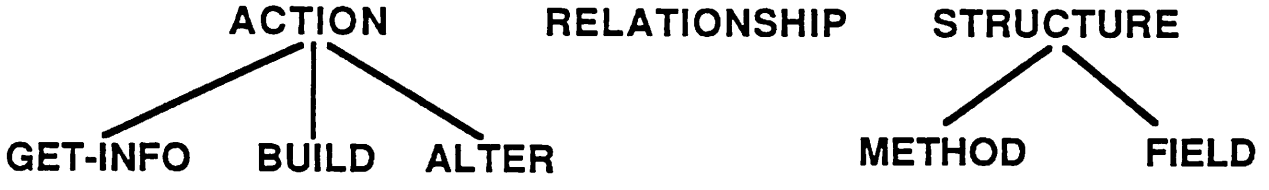

Figure 6: REST 0.3 frame categories.

STRUCTURE
TEXTNAME: sensors
ARG-OF: RELATIONSHIP
TEXTNAME: optical
USING: STRUCTURE
TEXTNAME: microprocessors
HAS-PART: STRUCTURE
TEXTNAME: silicon
AFF-BY: ALTER
TEXTNAME: failures
IN-CONTEXT: ACTION
TEXTNAME: energetic reaction

METHOD
TEXTNAME: protective coating
ARG-OF: RELATIONSHIP
TEXTNAME: highly effective

Figure 7: Parts of representation for document in Figure 4 that would be coded under our human coding rules.

Recall (%)	Precision (%)			
	$tf \times idf$	Plus REST term set	Plus dict. weights	Plus REST dependencies
10	57.5	62.2	59.6	63.7
20	47.2	48.0	51.7	51.4
30	34.8	38.2	40.6	41.2
40	31.0	33.2	35.2	35.2
50	26.4	28.5	30.7	31.1
60	21.7	23.7	24.6	25.9
70	13.5	15.7	16.1	17.4
80	10.3	11.1	12.2	13.5
90	7.6	8.4	8.0	9.0
100	5.6	6.0	5.7	6.3

Table 1: Enhancing probabilistic retrieval.

Method	Median Match Ratio (28Q × 10D)	Mean CRE (32Q × 10D)	Mean Precision (32Q × 5D)
$tf \times idf$ [Table 1]	—	.28	37%
Enhanced [Col 5, Table 1]	—	.34	37%
Matching Triples			
Basic Match	3.8	.53	41%
Query Coverage	2.5	.53	41%
All Slots Match	3.8	.53	40%
Hierarchical Match	3.7	.55	41%
All Categories Match	2.8	.56	41%
Reversed Match	1.4	.40	39%
Symmetric Match	4.3	.61	43%
Textname Match	2.5	.57	44%

Table 2: Reranking top 10 documents.