A Knowledge–Based System for Composite Document Analysis and Retrieval: Design Issues in the CODER Project

by
Edward A. Fox
Robert K. France

March 1986

TR–86–6

A Knowledge-Based System for Composite Document Analysis and Retrieval: Design Issues in the CODER Project†

Edward A. Fox
Robert K. France

Department of Computer Science
Virginia Tech, Blacksburg VA 24061

ABSTRACT

The CODER (COmposite Document Expert/Extended/Effective Retrieval) Project aims at applying a variety of methods developed in the realm of artificial intelligence to improve the performance of information retrieval systems. A prototype CODER system is being developed that will serve as a testbed for future research in this area. Initial experimentation will take place on a collection of more than three years of issues of the AIList ARPANET Digest.

CODER is being developed in MU-Prolog and C++ as a collection of experts communicating through central blackboards using UNIX™ pipes and the TCP/IP protocol. This distributed system can be divided up across several machines, to best utilize special display devices, storage facilities, and processors. There is a central spine, including document text and document knowledge representations, and a large lexicon being constructed from two machine readable English dictionaries. An entry/analysis subsystem carries out detailed analysis of composite documents, determining the structure and type of the whole and of each part. An access/retrieval subsystem has models of each user, can accomodate a variety of query languages, and supports browsing, searching, and immediate feedback.

Many issues must be dealt with in the design of such a system, including issues of knowledge representation, natural language processing, storage management and support environments. This paper gives background, describes related work, explains the design principles and architecture, and closes with future plans.

CR Categories and Subject Descriptors: D.2.6 **[Software Engineering]:** Programming Environments; E.2 **[Data Storage Representation]**; H.3.1 **[Information Storage and Retrieval]:** Content Analysis and Indexing; H.3.2 **[Information Storage and Retrieval]:** Information and Storage; H.3.3 **[Information Storage and Retrieval]:** Information on search and retrieval; H.3.4 **[Information Storage and Retrieval]:** Systems and Software; I.2.1 **[Artificial Intelligence]:** Applications and Expert Systems; I.2.4 **[Artificial Intelligence]:** Knowledge Representation Formalisms and Methods; I.2.7 **[Artificial Intelligence]:** Natural Language Processing

General Terms: Design

Additional Keywords and Phrases: abstract data types, blackboard, composite documents, expert systems, frames, interpretations, knowledge base, message passing, natural language processing, PROLOG, relational lexicon, relations, test collections.

†Project funded in part by grants from the National Science Foundation (IST-8418877) and the Virginia Center for Information Technology (INF-85-016).

1. Introduction

The CODER (COmposite Document Expert/Extended/Effective Retrieval) Project is an investigation of the applicability of artificial intelligence (AI) methods to the problems of information storage and retrieval (ISR). The CODER System is being developed as a flexible testbed on which to efficiently carry out these studies [FOXE 85a]. The initial experimentation with CODER will be conducted on a fitting but challenging collection of composite documents - the accumulation of AIList Digest issues distributed over the ARPANET since March 1983. It is expected that CODER will more effectively retrieve relevant passages and messages from this collection than would systems which solely employ Boolean, extended Boolean, vector, or probabilistic approaches.

This paper provides an overview of the CODER Project from the standpoint of design. First, it gives background information and surveys related research. Next, it outlines the design principles for CODER as a prelude to describing system architecture. Finally, a brief description of the system architecture is followed by a report on progress to date.

2. Background

Though visionaries have described their hopes for intelligent information retrieval systems since the mid-1940's (or even before) [BUSH 45], that goal has still not been reached. Many preliminary steps have been taken, though, and by integrating the results of some of the most productive efforts, it is hoped that significant progress can be made during the 1980's and beyond.

2.1 Retrieval Approaches

Information retrieval can be carried out in many ways, depending on the hardware, access methods, and retrieval model employed. In recent years there has been a revolution in storage media and devices that may allow relatively inexpensive systems to handle large databases [FUJI 84]. The advances in laser disks have been particularly impressive [GOLD 84]. It is hoped that experimental retrieval methods developed in the past few decades will soon be adapted to newly-available CD-ROM based systems [FOXE 86b]. Special purpose hardware for information retrieval processing *has* been available for some time [HASK 80]. Laboratory testing has become more intensive [HOLL 85] and will hopefully lead to lower cost as well as higher efficiency.

Much more effort has been focused, though, on improved methods and software. Approximate matching algorithms [HALL 80] can help avoid spelling errors or other errors caused by lapses in memory or lack of awareness of an author's means of expression. Sophisticated access methods aim at employing special data structures to save space and/or processing time [FALO 85]. The use of feedback information to help in the construction of improved queries was demonstrated by Rocchio [71], developed further in terms of probabilistic estimation by Robertson and others [ROBE 76], and explained more completely in [VANR 79] and [SALT 83a]. It was applied to large collections through an intelligent front-end system as well [MORR 83]. Clustering has been considered numerous times; the most recent thorough study is [VOOR 85].

Fuzzy set theory has been used to build a number of ISR models. Bookstein suggested including the ability to consider user-supplied term weights [BOOK 80]. Paice carried out some small-scale experiments to demonstrate the value of "soft" Boolean evaluation [PAIC 84]. The p-norm model, generalizing both of these approaches, was explained and validated [SALT 83c], applied to automatic query construction [SALT 83b], and adapted for (extended) Boolean feedback [SALT 85]. A recent study explored the p-norm model further and found it more effective than that proposed by Paice [FOXE 86a].

Further improvements have been sought by utilizing other information besides terms. The value of bibliographic information was established early on [SALT 63]. Bibliographic coupling [KESS 63], citations [GARF 78], and cocitations [SMAL 73] were all shown to help determine how closely pairs of documents relate. Relevance feedback techniques were developed to incorporate some of the bibliographic data [MICH 71] in searches. Since the results of different searches carried out for a single query tend to have small overlap [KATZ 82], it seems wise to combine a variety of types of information. Bichteler and Eaton found this to be of value with bibliographic coupling and cocitations [BICH 80]; a mix of those and other information does give demonstrable improvements [FOXE 83a]. Indeed, part of the appeal of using rule-based retrieval in CODER is to better handle the combining of the wide variety of available (indexing) information that describes most documents.

2.2 Retrieval Systems

In connection with the various models developed for ISR, there are many different systems to demonstrate their behavior. Most common are systems supporting Boolean queries and implemented through the use of an inverted file; a good overview of such systems and appoaches to search with them can be found in [MEAD 81].

Numerous experimental systems have been developed, including SIRE™ (marketed by KNM Inc.) and SMART [SALT 83a]. While SMART began in the 1960's, flexible ways to implement a new design were considered in the late 1970's [FOXE 81]. A UNIX version was completed several years later [FOXE 83c] and later tuned [BUCK 85].

Recently, many databases have been made available in full-text form [TENO 84]. In order to keep precision relatively high, the Responsa Project added a sophisticated morphological analyzer, and a richer set of metrical operators [CHOE 80]. Special "local" feedback was also shown to be of value [ATTA 77].

With the advent of low cost, high resolution bitmap displays, it became possible to display full pages of text and to rapidly browse through full text collections. The CALIBAN system incorporated browsing and search to become more responsive to users [FREI 83], and fully exploited the 2-D display to go beyond the normal 1-D retrieval approach [FREI 84]. Weyer [82b] experimented further with browsding-based methods by having students compare an "electronic" to a regular book, and found the results somewhat promising [WEYE 82a]. That approach has been carried further to develop electronic encyclopedias [WEYE 84], which are now becoming affordable due to the availability of new display and storage devices [COOK 84].

The newest approach to retrieval systems has been to build an expert system for that purpose. One of the earliest was developed under the supervision of Marcus at MIT, and aimed at recording and applying the expertise of search intermediates [YIPM 79]. Belkin et al. [84] used human experts to simulate an information provision mechanism, and found that a bulletin board with modified distributed control was nearly optimal. Applying the KANDOR language for frame-based knowledge representation [PATE 84b], Patel-Schneider et al. constructed ARGON which used a strict frame taxonomy to aid in classification [PATE 84a].

The RUBRIC system was developed to employ both fuzzy logic and inference rules [TONG 83]. Users whose queries will be repeatedly applied to incoming batches of messages will essentially construct a knowledge base that is evaluated to determine the similarity of documents to the original question [TONG 85].

Another expert system, being developed by Thompson and Croft [85], is closer in scope to the CODER system. There is a short and a long term memory, and seven experts, to: browse, explain, manage a thesaurus, build request models, build user models, search, and analyze natural language. LISP is employed so that rules can be coded for each expert's task.

2.3 Document Models

The CODER project is distinctive in emphasizing the analysis and retrieval of composite documents [FOXE 85b]. Most retrieval systems have fixed models of documents, and do minimal analysis. While it has been argued that sophisticated analysis is not feasible for documents, and should only be done on queries [SPAR 84], with recent improvements in computer hardware it seems that document analysis should be investigated further.

Documents have been studied under the aegis of other disciplines. Thus, the Electronic Manuscript Project is releasing in 1986 its initial findings regarding document organization and tagging, to aid authors and publishers alike [JENN 84]. The OTTER system was tailored for office documents [SACC 84]. The Office Document Architecture advanced yet another model for documents [HORA 84]. CALIBAN also used special data structures to represent both the typical hierarchical relationship present and other connections [BART 83].

Kimura investigated the structure of a variety of types of documents and proposed a technique for describing documents as well as editing them [KIMU 84]. His structure editor supported manipulation of both the structure and content of a document.

The recent work of Peels et al. focused on document analysis and formating [PEEL 85]. They discuss both a physical (page, box/glue) and a logical model. The logical model is made up of three different but interwoven information streams: primary, secondary, and illustrative. Each of these has its own simple grammar. Their prototype COBATEF system could analyze free text, build suitable representations, and reformat the original text to produce a nicely organized publication.

A special class of document is that used in electronic mail messages. There are special relationships among documents, established by operations such as embedding, forwarding, and cross referencing [BABA 85]. These matters are of particular importance in the CODER analysis subsystem.

2.4 AI Approaches

For more than a dozen years, work has progressed on the application of linguistic insights to the development of information retrieval systems [SPAR 73]. Lehnert studied the variety of possible question types that people construct , and how they should be answered [LEHN 78]. Oddy [77] viewed the retrieval problem as a dialog, stressing the human-computer interaction. To better understand users and their behavior, a small amount of psychological research has been conducted; much further study is required [BORG 85]. Most of the linguistic effort, however, relates to document analysis. In the retrieval community, attention has been given to the use of discourse analysis methods to aid in identification of answer-passages (i.e. for passage retrieval) [OCON 80]. Discourse analysis has also been of value in the TOPIC system, which uses word expert parsing [RIEG 81] to summarize text into a hierarchical condensation. Research relating to the Linguistic String Project has focused on applying a powerful parser to a given sublanguage, such as that found in medical reports [SAGE 75].

Parsing unrestrained text, however, presents a number of problems to conventional parsers. Charniak points out the importance of understanding the context of a given sentence [CHAR 82], and advances the idea that context is crucial for integrating syntax and semantics [CHAR 83]. Indeed, the FRUMP system [DEJO 82] could not function without having stored "scripts" to match against in order to establish the proper context(s). Schank et al. have used scripts [SCHA 77] for a variety of natural language analysis, including a conceptual approach to retrieval [SCHA 81]. Simmons [1984] uses a similar construct, the **schema**, to aid in analysis, building of a representation, and possible later translation. Wilensky et al. [84] use a phrasal approach to parsing, and have developed support for dialog, translation, and accessing a knowledge base on UNIX™ use.

At the heart of natural language processing is the matter of understanding words. Evens and Smith described a comprehensive lexicon to support natural language processing and question answering [EVEN 79]. A relational lexicon was also found useful for expansion of natural language queries submitted to the SMART system [FOXE 80, 83b]. Having a large lexicon proved crucial in the LSP efforts [WHIT 83]. Consequently, there has been a great deal of interest in machine readable dictionaries and lexicon construction. Merriam-Webster, in previous years, allowed researchers to format the *Seventh Collegiate* [SHER 74]. Amsler studied their *Pocket Dictionary* and discovered a "tangled hierarchy" of word relationships [AMSL 80]. Since that time there have been other dictionary studies as well [PETE 82, AMSL 84]. Ultimately, with new storage and processing devices, electronic dictionaries will be ubiquitous tools [FOXM 80].

A great deal of analysis is needed to build a lexicon from a dictionary. Tagging the Brown corpus with grammatical information was very valuable [FRAN 82]; similar work must be done, hopefully mostly automatically, on dictionary definitions. Ahlswede has studied adjective definitions [AHLS 83] and has been developing a tool kit for manual and/or automatic construction of a relational lexicon [AHLS 85]. Chodorow, Byrd, and Heidorn have focused on automatic extraction of semantic hierarchies from large dictionaries, considering some 40,000 nouns and 8,000 verbs [CHOD 85]. Rules for identifying word government have been shown to frequently aid in resolving ambiguities [EARL 73] - and word sense disambiguation is a crucial problem in working with machine readable dictionaries.

The result of parsing is typically some type of knowledge representation. Manual analysis can also be used to construct a knowledge base from text, if strict rules are applied [SKUC 85]. Many representation schemes have been recommended for document retrieval [SMIT 84]. One very popular scheme is the frame, which may have a variety of slots that can be filled [MINS 75]. Frames can be used to describe objects as well as to record control information, and so can aid in machine reasoning efforts [FIKE 85]. Alternatively, collections of rules have been employed in building many of the expert systems now in preparation or use [HAYE 85].

Prolog, which was chosen as the primary development language for the CODER project, has been been criticized by some and hailed by others. Bobrow [85] compares Prolog with the Loops system, and feels that a good AI environment should integrate a variety of paradigms instead of having just one language. On the other hand, Lee [85] has built an expert system shell in Prolog that handles forward and backward chaining, agendas, a blackboard, and some user interfacing aids. Further discussion of the pros and cons of using Prolog for expert system construction appears in [SUBR 85]. Winett and Fox describe the application of information retrieval methods (e.g., p-norm queries) in a Prolog expert system [WINE 85]. Indeed, logic programming seems to be a very important paradigm for wide classes of AI development efforts [GEGE 85]. As Prolog interpreters and compilers mature and continue to improve in speed and clarity, and as environments are developed, the language should become even more widely utilized [COHE 85].

3. Design Principles

The purpose of the CODER project is to provide an experimental testbed for investigating the use of artificial intelligence techniques in the storage and retrieval of composite documents. The project is designed to allow a variety of techniques from different branches of AI to be applied to various aspects of the task of analysis, indexing, and retrieval of documents. It is hoped that the system will be of use for a wide range of experiments, as the SMART system has been over the last decades, and that it will have the flexibility and ruggedness to endure, like SMART, over a considerable lifespan [SALT 83a].

In this section, we discuss design issues raised by the project mission and some of the decisions that were made to resolve them. In making these decisions, our basic aim has been to keep the environment both powerful and flexible enough to satisfy the evolving demands of an

experimental system. The fields of artificial intelligence and information retrieval are both currently undergoing a rapid process of change. If CODER is to serve its purpose as a comparative system, it is important that it be able to adapt to such changes as they occur.

3.1 Knowledge Representation

The CODER system is designed to bring considerably more power to bear on the problems of information storage and retrieval than can more conventional systems. It does this both through more powerful methods and more powerful representations. The **methods** include a marriage of the more successful results of the latest generation of information retrieval systems with the recent developments in expert systems and computational linguistics. The **representations** are symbolic structures of the type evolved in the artificial intelligence research of the last two decades with the aim of representing knowledge.

Basically, CODER will need to use at least two sorts of knowledge: general knowledge about a subject area, and specific knowledge about entities in the problem universe. It will also need to model these entities, which include words, names, and other lexical items; documents and fields of documents, and users of the system. The system needs to represent and control facts about these entities, if possible associating the facts with some sort of confidence level. In addition, it must represent and manipulate rules for recognizing general classes of entities, facts about such classes, and metaknowledge about the interaction of classes of facts. The problem faced in designing a system of knowledge representation for the system is therefore to provide facilities for modeling these entities, modeling attributes of the entities and facts relating one to another, and modeling knowledge about classes of entities, including both general factual knowledge and procedural knowledge detailing, for instance, how to classify or manipulate objects of a given class.

The CODER system provides these facilities through three categories of data types (or, in anthropomorphic terms, three domains of discourse that it can understand). First, the system understands elementary data types. These are the sorts familiar from the theory of abstract data types: sets of objects associated with sets of valid operations ranging over them. The classical types CHAR, INT, REAL and ATOM are provided in this category. New elementary data types can be created through the specification of the set of operations defining them, or through quantification or restriction of existing elementary types.

Second, the CODER system provides a general frame-manipulation ability. Frames model objects with attributes, not all of which need be specified in every instance. Only a weak inheritance principle as in [PATE 84b] is provided automatically; arbitrary relationships between frames can be added at a meta-level or inferred on the fly. This allows the frame domain to be itself well-behaved and easily defined while allowing arbirtarily complex and idiosyncratic accretions to occur at the level of knowledge *about* the frame domain. In particular, this allows the system to maintain knowledge relating frame types (or objects represented as frame instances) on the basis of their **similarity** to each other, something that is not possible in most classical frame-based systems.

Finally, the CODER system understands the domain of relations. Relations model facts or conditions relating objects together: *synonymy* is a relation between words; *representing*, a relation between a sentence and the abstract representation of it produced by a parser. Relations may have any number of arguments, and may include weights, but always either obtain or do not obtain: unlike frames, they cannot partially apply to an object. The CODER system provides facilities for definition and use of relations, including limitation of the types that may occur as a given argument of a relation, and for maintenance of the attributes of relations, for instance whether or not a given relation is symmetric, transitive, and so forth.

Relations can be used to model a wealth of knowledge representation schemes, including semantic networks and J. F. Sowa's "conceptual graphs" [SOWA 84]. Frames, of course, are

widely used themselves in knowledge representation, and can also be used to model more specialized entities such as case structures or Conceptual Dependencies. And elementary data objects, as mentioned, provide both the power of abstract data typing and a primitive method of operation inheritance. These three categories thus encompass most existing AI formalisms. In addition, we have noted that they are sufficient to represent attributes, objects, and relations among objects, and thus provide a rich descriptive vocabulary for representing not only the initial objects in the problem domain (documents, names, words, and so forth), but abstract structures built up out of those objects, and arbitrary relationships between them. So there is good reason to believe that they will continue to be adequate for any representation formalism that the problem domain requires.

3.2 Document Architecture

One of the first benefits of the system's representational power is its ability to easily handle **composite documents**. CODER represents the information with which it is designed to work neither as relational tuples nor as flat strings of text, but as structured entities composed of fields, each of which can be filled by only certain types of data. These fields may themselves be composed of other fields with more specific restrictions on the types and semantic content of data that may fill them, and so forth. This approach allows different aspects of a document to be represented in content-appropriate ways, rather as is currently done by a human cataloger. Moreover, by being able to recognize the semantic restrictions on a given field of a document, the system is given the opportunity to use specialized parsing techniques or inference methods in analyzing the data in that field, and to use specialized disambiguating and clustering techniques during retrieval.

To accomplish all this, however, the system must provide facilities for defining and manipulating both structures of fields and the different data types that fill the fields. These structures may themselves contain structures (as when a *date* occurs as part of a *bibliographic reference* within the *bibliography* field of a *journal article)* or sets or lists of structures. It must be possible both to navigate within such structures and to specify methods for recognizing and making inferences from the data types that make up their fields. Finally, it must be possible to create and store abstract representations of structures, for instance in indexing an analyzed document, even when they are not fully instantiated. For instance, the system must be able to segment a document into a list of bibliographic references even if not all the references are complete, or to identify it as a report of an event even if the document does not exactly match the template for a *report of event* structure.

Of the various knowledge representation structures provided in the CODER system, composite documents are probably best described by frames. As mentioned above, frames model objects with typed attributes. In composite documents, the attributes in question are the contents of the document fields. For example, one attribute of an *electronic-mail message* (one slot of the *electronic-mail message* frame) is its date of origin, which is of type *date* (itself a frame with slots for *day, month* and *year).* Thus, a document type is defined by listing the possible attributes that a document of that type can have and the types of its fields.

Of course, knowledge about a type of document is not limited to its definition. Associated with a given document type may be semantic knowledge (such as the expected content of a field), inter-type knowledge (which document types are also permitted to be -- or are also likely to be -- which other types), and relations among fields (this field is *required* in a document of this type; these two are *mutually exclusive).* This knowledge is well modeled through the logical relation data type and can be managed in the system by a Document Type Expert.

Using frames to represent documents has certain obvious advantages. Since frames may have other frames as slot fillers, it is relatively simple to mirror schemes for hierarchical decomposition of documents, such as ISO-WG3 [HORA 85] or the COBATEF model [PEEL 85].

Text fields, for instance, may be defined as lists of paragraphs; paragraphs as lists of sentences; sentences as lists of text items. Markup structures may be included as separate fields of the document frame and/or the component frames (lists or tables, for example, often require specific layout information to clarify their semantic form). In addition, since a frame instance need not have all its slots filled, it is easy to create representations of a document based on imperfect matches. If a document is interpreted as being similar to an ideal type, a partial instantiation of that type can be formed to represent the document. Those aspects of the document that correspond to the ideal can be used to fill slots in the instantiation and those aspects of the ideal that have no match in the document can be ignored. Any aspects of the document that do not fit the ideal can be either ignored as well or fit to another ideal, thereby creating a separate interpretation of the document.

When a document is modeled by several interpretations, of course, there is a possiblility that the interpretations are inconsistent among themselves. This is not always the case. We can say that a document is a *journal article* and at the same time a *book review* without being inconsistent. Neither is it inconsistent to describe the same document as a *bibliographic reference*, although the bibliographic portion of a book review is usually only a small part of it. Inconsistencies may nevertheless arise, however, when a key text item is interpreted in different ways, or simply when different aspects of the document invoke different ideals. When this occurs, we say that the interpretations formed from the document are **not mutually satisfiable.**† We can, however, find maximal satisfiable subsets of the set of all interpretations. Inclusion in one or more such subsets can then serve as a criterion for a good indexing interpretation; i.e., for an interpretation that can be stored as knowledge describing the document.

3.3 Natural Language Analysis

Probably the most difficult parts of a composite document for the system to handle are the **text fields** that make up the bodies of most documents in an information retrieval environment. Actually, given the experimental nature of the project, it must be possible to insert any of a number of different natural language analyzers, with different theoretical bases and of different levels of sophistication, into the system in order to assess their impact on the information handling process. Of course, it is impossible to create a system that will support equally well any of the multitude of natural language parsers proposed by the computational linguistics community. The CODER system, however, has enough flexibility to work well with any of a wide range of parsers, including but not limited to those based on Augmented Transition Networks [BATE 78], Definite Clause Grammars [PERE 83], Linguistic String Grammars [SAGE 81], and Phrase Structure Grammars [POLL 85]. Each of these paradigms can lead to high quality parsing of natural language text, and in the current state of computational linguistics research, it would be foolhardy in the extreme to commit to only one.

Flexibility in choice of parsers is ensured in two ways: first, through providing the flexible knowledge representation structures detailed above, and second, through divorcing the process of parsing from the remainder of the system. In accordance with our general design principles, the expert or experts responsible for parsing incoming text are independent from the rest of the system, communicating only through hypotheses posted to the analysis blackboard. This modular separation keeps the parser from interacting directly with other system experts, for instance with

†It should now become apparent that the term *interpretation* was not lightly chosen. What we are doing in describing a document through frame interpretations is building a (set of) theory(s) about the document content. This is the analog in the frame language to the analysis of validity for the language of predicate calculus in classical model theory, and we will adapt the vocabulary of model theory here as an alternative to the more problematic *possible worlds* interpretation of non-monotonic knowledge. Thus we describe the process of document cataloging as one of finding maximal satisfiable sets of representations in the frame language that hold for the document, and we will say that the document is a model of such a set.

those responsible for determining the type of an incoming document or for choosing appropriate indexing relations. It does not, however, prevent a more subtle interaction at the level of the knowledge structures used to represent text. In an information storage and retrieval system where the information stored and retrieved involves natural language, it is always necessary to arrive at a canonical representation of the input documents and a (possibly different) representation of the users' queries. The users' information needs can then be compared to the documents' information content, and matches or close matches can be discovered. These canonical representations can be as simple as lists of words or word stems, or can involve complex logical or semantic relationships. Theyu must, however, be based on a single set of primitives and structuring operations, even when the language in which the users' queries are constructed is different from the language of the input documents.† It is likely, therefore, that any change in the meaning representation structures produced during text analysis will require changes in the retrieval subsystem as well. These changes may either be made in the query parsing module, so that it can produce the same structures produced by the document text parser, or they can take the form of transformations from the structures produced by the query parser to the structures produced by the text analyser, performed during the process of retrieval.

In addition, changes in the representations produced by the text parser may require changes in the experts responsible for abstracting and using indexing knowledge. To the extent that these experts use heuristics linked to the type of knowledge structure produced, they will have to be changed whenever the structures are changed. Thus, the choice of an abstract representation for natural language has far-reaching effects in the system, effects which cannot be easily controlled. The choice of a system for converting text into such an abstract representation, however, is not constrained by interaction effects: many such systems may be tried with only local changes. And the CODER environment provides sufficient flexibility that the more far-reaching decisions of text representation can at least be approached from an experimental point of view.

Despite this flexibility, choice of a natural language parser is limited in two important ways. First, of course, the parsers are limited in the type of output they can produce. It is assumed that whatever parser is chosen will reduce the language of the input text to some sort of abstract structure, *and that that structure will be representable in the domains described above.* Actually, this is a minor restriction, since to our knowledge any conceptual structure yet proposed can be represented as a subset of the domains of relations and frames (semantic nets, for example, are formed from relations, while case structures can be regarded as types of frames). More importantly, the candidate parsers are limited by the raw material with which they are constrained to work. Any natural language parser requires some information about the words in the language in order to do its work. In the CODER system, this information is contained in a **relational lexicon** abstracted from several sources, most notably from machine-readable tapes of at least two major English dictionaries. Providing the information from these dictionaries to the parser designer does not, of course, prevent use of different sources of knowledge (attached procedures, for instance), but it poses what may be an irresistable temptation to use the knowledge already in the system. What is more, it poses the temptation to use the knowledge in the form in which it already exists: relations among words, or relations between words and elementary domains such as *parts of speech* or *semantic categories.* Again, these relations can be used in a wide range of parsing strategies, including those listed above. And it must be noted that machine-aided translation projects, which have a comparable goal in requiring robust parsing of most text and

†This is not as simple to achieve as it sounds. Even if we take it that both the input documents and the queries make use of the same subset of the natural language in which the system is based (which is almost a reasonable assumption, although not quite true in practice), it is a truism of empirical information-retrieval research that neither the individual words nor the linguistic constructs used in forming questions and expressing needs are the same as those used in the expository diction characteristic of the target documents. Thus a query parser may share the same language recognizer as a document analyzer, but will generally require a different set of meaning representation productions.

graceful failure on the remainder, have had good results using relatively simple grammars coupled to large lexicons [GAZD 85]. Thus we expect this restriction also to be relatively minor in practice.

3.4 Lexicon Construction

In order for any language analyzer to be other than a toy (or at best, an interesting research project with more implications than results), it must be able to draw on a large body of knowledge about the language it is analyzing. This knowledge can be thought of as belonging to two domains of specialization: knowledge about the words in the language, and knowledge about how those words can be combined to make larger meaning-carrying units such as phrases, sentences, and paragraphs. The latter is consigned in the CODER system to the local knowledge base of the natural language parsing expert (or experts); the former is contained in the CODER Lexicon. The Lexicon, which can be consulted by several experts in the system beside the text parser, serves as the repository for syntactic knowledge (parts of speech; whether a noun is countable or uncountable), semantic knowledge (relations of synonymy between words; hierarchical relations among definitions), and pragmatic knowledge (appropriate realms of diction; knowledge pertaining to specific domains of discourse).

Lexicons for computational linguistic purposes have been constructed in a number of ways. Most of the classic artificial intelligence text understanding programs have used lexicons constructed laboriously by hand. Generally, these have been small and restricted to a narrow realm of discourse. Exceptions to this rule, such as FRUMP, have still required hundreds of man-hours invested in lexicon construction. Accordingly, it seems appropriate to look elsewhere for word knowledge, and to enlist the help of the computer in obtaining it. Some researchers [WHIT 83, AHLS 84] have repaired to the document text itself to discover such knowledge as permissible subjects for verbs and (candidate) taxonomic relations. Using these tools and interactive techniques for obtaining knowledge from system users, they have successfully amassed lexicons in the thousands of words. There is, however, another common source of knowledge about words: that used by humans. Dictionaries provide not only the discrete information needed by a syntactically-driven parser, but also a wealth of semantic information that may be used to establish, for instance, that the parsing of a phrase uses word senses from consistent semantic categories. Ahlswede, Evens, and Smith have all advocated (semi-) automatic analysis of dictionary definitions to streamline the lexicon construction process, and such pioneering work as that of Robert Amsler supports the credibility of such an enterprise.

In a few years, the *Oxford English Dictionary* (OED) will become available in machine-readable form, courtesy of the efforts currently by Waterloo University and Oxford University Press [HULT 84]. The OED, itself a result of decades of effort, covers virtually the entire English language, with the exception only of terms and uses that have entered the language since its completion in 1928. (Even these are covered in supplemental volumes through 1984.) There are many issues of interest in that project, for computer scientists and linguists alike [LESK 1985]. Meanwhile, the *Oxford Advanced Learner's Dictionary of Current English* (OALDCE) [HORN 74] and the *Collins English Dictionary* (CED) [HANK 79] have recently been made available by the Oxford Text Archive for research purposes. Both of these are being used in the construction of the CODER lexicon. Several other texts of considerable computational linguistic interest, including the *Oxford Dictionary of Quotations* (ODQ) and the *Oxford Dictionary of Contemporary Idiomatic English* (ODCIE), are also available. These last in particular can be of value for matching phrases and idioms, and for obtaining archetypical samples of use.

OALDCE is a dictionary intended for use by people learning English as a second language. Though relatively small (c. 24,000 entries), it contains a great deal of highly specific information, including simple definitions, examples showing the use of different word senses (often in the form of sentences with explanations), idiomatic phrases, national differences in spelling and usages, and verb case structures. CED is a large up-to-date one-volume dictionary, with over 162,000

references (85,000 headwords) and 14,000 biographical and geographical articles and with excellent coverage of science and technology. It provides semantic category information for fully 15% of listed definitions, several different sorts of cross-references to related words, and notes explaining proper usage of the words. Together these two dictionaries have a wealth of information to use in automatic text analysis.

The CODER lexicon is being developed in stages. First, the tapes must be converted from typesetting format to a structural form more suited to high-level manipulation. This involves both a clean-up phase, to remove spurious data such as page breaks and space left for illustrations, and a parsing phase, where the semantic content of indentations and font changes are translated to explicit semantic relation markers. Roger Mitton has recently completed a cleanup effort for the OALDCE; the Collins dictionary has been cleaned up locally during the Fall of 1985. Also locally, the UNIX™ tools *lex* and *yacc* are being used to convert the grammer implicit in the typesetting conventions of the dictionary definitions to produce relations in a syntactic form acceptable to direct manipulation by a Prolog interpreter. The files of Prolog statements produced as output by these parsers constitute the end point of the first stage.

Next, the CED and OALDCE data must be merged and ambiguities resolved. It is likely that a few words that occur in the OALDCE will not occur in the CED, and that many will occur only in the larger CED, but the overlap of the two is nonetheless considerable. Within that overlap, however, there may be little commonality between the two dictionaries in the differentiation of senses within a given word entry, or even in how many entries a given lexeme is given.

Third, other sources like ODQ and ODCIE can be utilized to add more information. This will provide knowledge on language pragmatics; it is an interesting open question how much help the "quotable quotes" of Shakespeare and Browning will provide in document analysis and retrieval. Additional knowledge on pragmatics can be obtained from problem domain-specific texts (for the first text collection of AI messages, such knowledge may be able to be abstracted from the machine-readable form of the *Handbook of Artificial Intelligence,* recently released to researchers on a limited basis). Knowledge about words can also be input interactively during the process of document analysis, for instance when a new name is encountered in the input text.

Finally, parsing of dictionary definitions will be undertaken so that kernel words and lexical semantic relations can be identified and recorded. Smith has found in work on *Webster's Seventh Collegiate Dictionary* that a high proportion of definitions fall into a few syntactic forms, and Ahlswede has used these **defining forms** to analyze adjective definitions [AHLS 1983]. These defining forms, besides helping to identify the key terms in a definition text and their function in explicating the word sense being defined, can themslves provide semantic information about the word sense: for instance, Evens notes that the form "one who" identifies the sense as referring to a human subject. Obviously, these last three steps may proceed in parallel.

3.5 Test Collections

One of the major criticisms leveled against the experimental work done on information storage and retrieval in the past has been that the data sets used were small and controlled, and that the results obtained and techniques evolved did not scale up well to large files of real data. Specifically, models for retrieval based either on vectors or Boolean combinations of words worked well when tested on cases where the number of word types was high compared to the number of documents in the system. Recent results indicate, however, that they may not fare as well when applied in situations where large numbers of the available documents can be found containing any reasonably common word [BLAI 85]. While the system investigated in this study was a commercial system, optimized for performance and not reflective of current advances in conventional ISR, the large-colection effect can be significant for any system. Consequently, the CODER system has been designed to function on reasonably large collections of realistic data.

As an example, the collection that has been created for the initial testing runs is a set of electronic mail messages drawn from about three years of postings to the ARPANET "AIList" digest. More than 3000 individual messages occur in the collection, each of which is considered as a single composite document; between all the documents, the collection comprises around a million words. The documents vary considerably in length, content, style and diction, and include such disparate entities as calls for papers, announcements of seminars, requests for information, philosophical wanderings, and lists of bibliographic references (including references to previous postings in the AIList!). There is throughout the collection, however, a certain unity of content and a common vocabulary and body of understood knowledge. Future document collections will include documents drawn from different sources, and eventually even from several sources at once.

Use of such large collections, however, raises several issues. First, of course, there are issues of efficiency. Since the CODER project is only required to operate in a research environment, the efficiency of analysis and storage of documents is not crucial, and CPU hours can be spent assembling, analysing, and storing the collection that could not easily be spared in a commercial environment. Retrieval speed, however, is if anything more crucial in an experimental system (where exhaustive testing of different configurations involves multiple sets of retrieval runs, each of which may require many documents being retrieved and presented to the user) than in a production system. Thus the databases, not only of the documents themselves, but of the knowledge indexing the documents, must respond quickly even given the large number of documents and pieces of knowledge in the system.

Next, the use of large collections raises issues of hardware. In a research environment no less than in a commercial one, storage space on any given computer system is a scarce and valuable commodity. CODER requires large amounts of space, both for the document databases and for the lexicon. Lacking special-purpose hardware, the most reasonable solution seems to be to structure CODER as a distributed system, allowing separate knowledge bases to exist on separate machines.

Finally, the use of large collections raises issues of testing. The early work in Boolean and vector retrieval used small, well-controlled collections precisely in order that the techniques under investigation could be tested completely. With a small collection of documents, one can determine whether or not a document is relevant to any query by asking knowledgeable human beings. In collections the size of actual document data sets, this is no longer possible. While it is precisely because these data sets are too large to be cataloged by humans that the issue of automatic analysis is so critical at this time, these sets can only be used in an experimental context if we are to give up measuring system performance in terms of absolute values of recall and precision. We can, of course, compare configurations of the system among each other, and we can compare sets of documents retrieved by versions of the CODER system with those retrieved by versions, for instance, of the SMART system. Such comparative measures, however, will have to be our standard in working with large collections of realistic data.

3.6 AI Support Environment

The CODER project has been conceived as an investigation into the applicability of the techniques of artificial intelligence. Other than the question of natural language parsing, there are several ways that AI techniques can aid the information storage and retrieval process. Abstraction from the results of document parsing to the key concepts under which the document can be indexed is a process beyond the reach of conventional programming techniques, but not intuitively beyond those of rule-driven systems. The knowledge maintenance techniques required to ensure consistency of a document interpretation or a set of hypotheses about a user's information need have only been explored in the context of AI, as have the techniques required to relate facts in a knowledge base to the entities that they describe (and entities to the facts that describe them) and to trace which types of knowledge are most helpful or what sources of knowledge least suspect. Planning a search, expanding a search through the discovery of semantically related concepts, and

understanding a user's response to a search all come under the general heading of areas where artificial intelligence techniques hold great promise. Use of these techniques, however, requires a commitment to an environment for their support and to the paradigms of their use.

The development of an artificial intelligence system typically follows a different paradigm than the conventional design/build/test approach [BOBR 85]. Typically, AI developers prefer an incremental, exploratory approach where system design and implementation evolve together with the developers' understanding of the problem. Maintaining a coherent system under exploratory development by several different people, however, requires more than an exceptionally steady hand. Thus it has been necessary to limit the exploration possible by any single developer working on the CODER system. Rather than limit the directions in which such exploration may proceed, though, coherence is maintained by limiting the interactions between system modules and by limiting the size of the domain within which any given developer may work. Developers working on the CODER system are required to work within the constructs of a limited set of module types obeying strict communication standards. The internal structure of each module (for instance, whether it is inferential, pattern-directed, or even procedural) is left to the designer's judgement, but the external interface it presents to the remainder of the system and the knowledge structures it represents, are rigidly specified. This provides a maximum of freedom for exploratory development within the domain of a given module, while still ensuring that the modules will fit together.

Supporting AI techniques also requires a very high-level language in which the knowledge and inference techniques can be coded. This requirement conflicts directly with the requirement for efficiency in retrieval, as VHLLs are notorious for their slow execution. For the CODER project, however, efficiency is most crucial in the database aspects of the project, and there exists a language dialect, MU-Prolog [NAIS 85], that provides a very high-level paradigm oriented to artificial intelligence work and also provides strong support for large built-in knowledge bases. Prolog has been used widely in AI programming, notably for expert systems [LEE 85] and natural language parsing [PERE 83], and MU-Prolog itself has proven effective in a variety of knowledge representation tasks [HELM 85]. However, Prolog is often cited as a better language for prototyping than for system construction. This is in part due to the power of the Prolog interpreter combined with the simple and untyped Prolog environment. While a Prolog program is easily decomposed into apparently independent modules, the rule database constructed by the interpreter from a program is monolithic, and large Prolog programs often collapse into a sea of unwanted interaction effects. This effect is difficult enough in a program crafted by a single person who can, at least, ensure the purity of the local name-space. In a multi-programmer environment, it is magnified beyond endurance. This effect is avoided in the CODER project by the simple expedient of invoking a separate copy of the interpreter for each Prolog-based module. Thus, an individual experimenter can build, for example, a natural language parser or a search planner without either fearing unwanted interaction with other modules of the system or needing to worry about how those modules are built.

Thus, for several reasons, we have found it necessary to break the CODER system into modules. Modular decomposition is, of course, the accepted methodology in conventional software design, but its application in artificial intelligence systems is problematic. It is relatively easy to specify the decomposition and external characteristics of a user interface; less easy, but still relatively straightforward to specify those of a frame manipulation module; very difficult to specify the decomposition of the task of retrieval, or the external characteristics of the (sub-)task managers involved. We have noted above the necessity of restricting the possibilities for exploratory development, but there is every reason to believe that our decomposition of the major tasks of the system will change as our understanding of the problem evolves through modeling and experimentation. Software engineering and AI make very uneasy partners.

Our solution to this apparent dilemna has been provided through the concept of **expert systems**. This term has been used widely in the last few years to mean many different things: here we use it to mean that the CODER system functions by applying general domain knowledge, or **expertise**, to solve the problems of document indexing and retrieval. This decision serves two

goals. First, it keeps expertise explicit, implying that it is coded separately from any inference engines used by the experts. This accords well with two of the primary lessons of the last decade of AI research: that intelligent behavior depends heavily on the knowledge of the behaving system, and that in artificial systems this knowledge is best engineered separately from the mechanisms for manipulating it. Second, it suggests that problem decomposition proceed along the lines of the domains of knowledge used in discovering solutions.

Each of the two primary tasks that CODER addresses is thus assigned to a group of cooperating experts. Each expert is closely bound to a single sub-task, either specific to one of the larger tasks or adaptable to either. The community of experts involved in a task can change over time, as the decomposition of the task changes, without affecting either system resources such as the lexicon or the document knowledge base, or, more importantly, the underlying structure of the system. Individual experts can also change, or even be replaced, in order to better adapt them to their missions, but such changes will not effect the other experts in the community. In fact, most changes in the composition of the community will not affect most experts: changes in the performance of the task of query parsing, for instance, need have no effect on an expert whose charge involves discovering synonyms for words.

An expert is assigned a small area of specialization. This both isolates the development of the expert from that of the surrounding system and mitigates the problems of rule interaction within the expert. Tasks which are found to be too complex can be further subdivided along the lines of the areas of expertise required to solve them. As a further benefit, experts can be specialized to deal with different types of knowledge, so that an expert that manipulates knowledge of *how* to do things can use different inference mechanisms than an expert in *what* to do. This will enable the reasoning portions of these experts to run with more efficiency than, for instance, general rule-based inference engines (see [LEVE 84] for a formal analysis of this effect). In practice, we expect that a few generic knowledge-handling engines can be specialized into a plethora of different experts. The work of Chandrasekanan (e.g., [CHAN 85]) holds great promise that a few such powerful generics can be isolated and put to good use. And as such methods are better understood, they can be used in CODER to build new experts, again without affecting the existing modules of the system.

This method of problem decomposition is particularly well suited to the tasks of document analysis and retrieval, where the relevance of as given document to a given information need is typically overdetermined by many weak factors: occurrence of certain terms or meaning structures in the document text, authorship by an authority in the field of the user's need, or currency of the information in the document, to name but a few. It is expected that it will also be relevant to a much larger class of problems where solutions are also weakly overdetermined by the solution to many different subtasks, and where the factors can be isolated by the domains of knowledge required to solve each subtask. The final design criterion of the CODER system is thus that it be built with as much generality as possible, both so that structures created at one point in the system can be used in other areas, and so that the general structure of the system, *qua* expert system, can be reused to solve other problems.

4. Architecture

The CODER system has been designed to run under UNIX™ as a collection of C and MU-Prolog processes which communicate through pipes or are interconnected by sockets using TCP/IP [LEFF 84]. It is expected that two 3B2/300's, one SUN-2/170, and a VAX-11/785, each running slightly different versions of UNIX, will ultimately be interconnected so that varying storage and user interface cabilities can be properly utilized. Later, one or more AI machines with Prolog compilers and/or microcoded language support may be added. Special bitmap interfaces for the SUN and 3B systems will be used to provide enriched user interfaces. The C portion will be coded in C++ [STRO 85] so that abstract data type specification and object oriented message

passing can be adopted as the preferred style.

Conceptually, the CODER system can be thought of as two separate subsystems sharing a central spine of common resources and knowledge. On one hand, the **analysis subsystem** is responsible for cataloging new documents; on the other, the **retrieval subsystem** is responsible for retrieving documents or portions of documents that satisfy a given user's information need. Actually, any number of analysis and retrieval sessions may be running at a given time (Figure 1).

The CODER **spine** is made up of the central knowledge bases of the system and a set of type managers that support the knowledge representation structures used by the system. The knowledge bases are comprised of the **lexicon**, which includes the system's knowledge about individual words, and the **document database**, which handles knowledge about individual documents. Type managers for frames, relations, and elementary data types support the uniform use of these representations throughout the system. Associated with the spine is that portion of the expert community that specializes in manipulating the knowledge available from these external knowledge bases and type managers. These satellite experts, like the modules of the spine proper, can be thought of as resources available to any session, either analysis or retrieval, ongoing at any time.

Any ongoing session is moderated by an active blackboard. A **blackboard** [ERMA 80] is a repository for communication between experts, usually divided into a number of subject posting areas. In addition to subject areas, each CODER blackboard includes a **question-and-answer area** and a single **pending hypothesis area**, which are accessible to all experts in the community (see Figure 2). The question-and-answer area provides a structure through which an expert can request information from the other experts in the community before continuing processing; the pending hypothesis area contains a consistent set of high-confidence hypotheses, accessible not only to the experts in the community but to the outside world. CODER blackboards are considered to be active since each is managed by a **strategist**, which carries out the main planning and control operations for the session. The strategist initiates the participation of each expert in the community through a set of heuristic rules based on a model of the expert's area of competence. It is also responsible for selecting the contents of the pending hypothesis area from the hypotheses proposed by the community of experts, combining evidence supported by suitable levels of confidence through a related set of rules.

The analysis subsystem includes a specialized user interface that allows for easy document entry and on-the-fly correction. With sufficient permission, the user of the analysis subsystem may also add knowledge to the portion of the lexicon dedicated to specialized knowledge in the problem domain, for instance in defining technical terms new to the system. Coupled to this interface is a blackboard that coordinates the experts involved in natural language processing and document cataloging. During an analysis session, the system accepts input documents of various types, arranges for as-is storage, and constructs a set of interpretations describing document structure and content. Each interpretation is itself a set of facts (ground instances of logical propositions) that can be stored in the document knowledge base (see Figure 3).

The retrieval subsystem uses these facts along with the knowledge about words stored in the lexicon and a specialized fact base of knowledge about users, to match documents or portions of documents to a user's information need. The user interface for the retrieval subsystem is designed to be adaptable to different styles of query presentation, including Boolean or extended Boolean logic queries and natural language descriptions of information needs. User behavior is monitored by a specialized expert, and the resulting feedback can be applied to sharpen the retrieval. The entire session is coordinated by a strategist whose heuristic rule base may contain expertise on search approaches and search strategies.

Communication among modules is restricted to input/output lines modeled on the UNIX™ pipe construct [RITC 74]; no other sharing of data is allowed. The socket construct provides a means by which even modules that service many clients (on one or many machines) can appear to be serving a single input stream. Vagaries of individual machines, such as the means for implementing this communication policy and the exact interfaces presented to users, are shielded within **resource managers** that map invariant abstract operations into code that can be

CODER

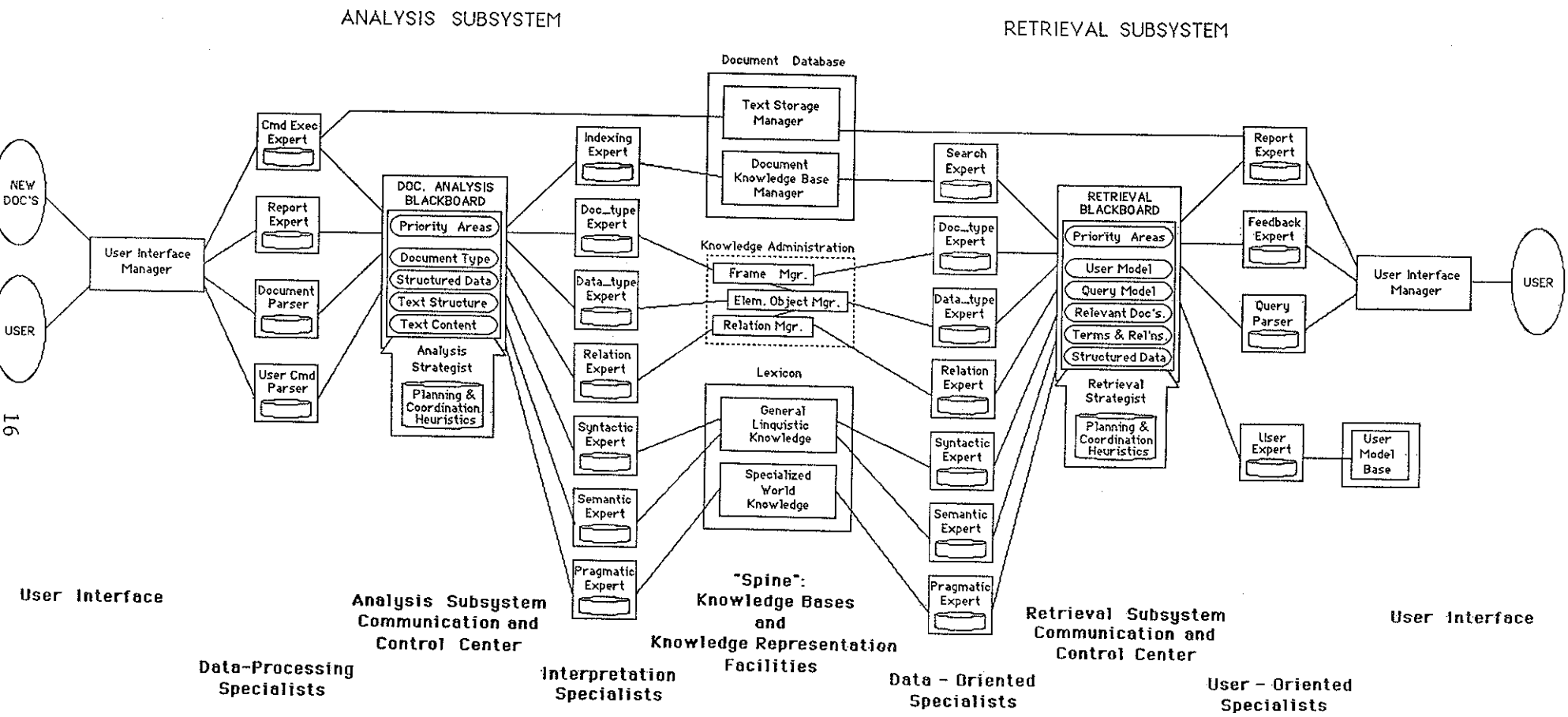
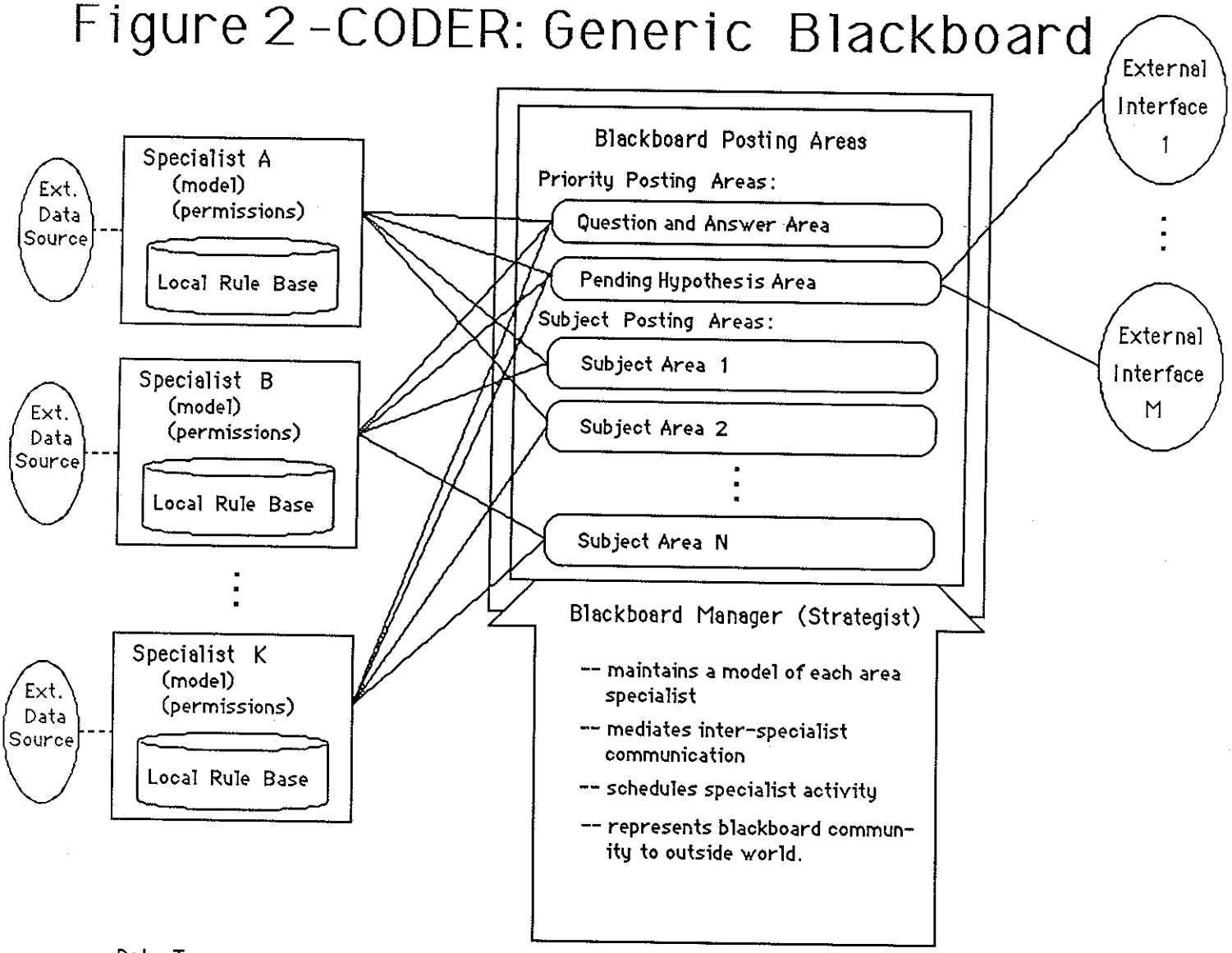


Figure 1

Figure 2 -CODER: Generic Blackboard



Data Types:

Hypothesis -- A 4-tuple
<fact, confidence-level, expert, reasoning>
where 'fact' is a Prolog fact, 'confidence-level' a real weight, 'expert' the ID of the specialist making the hypothesis, and 'reasoning' the chain of deduction that justifies the hypothesis.

Question -- An incomplete hypothesis, one or more arguments of which are replaced by variables. Each question is posted to the reserved Question area of the blackboard until it accumulates a full set of answers (one for each specialist in the system), upon which it is returned to the specialist that originated it.

Answer -- An hypothesis that partially instantiates the question with which it is associated; ie, a copy of the question with one or more variables replaced by constants, signed and justified by the specialist supplying it.

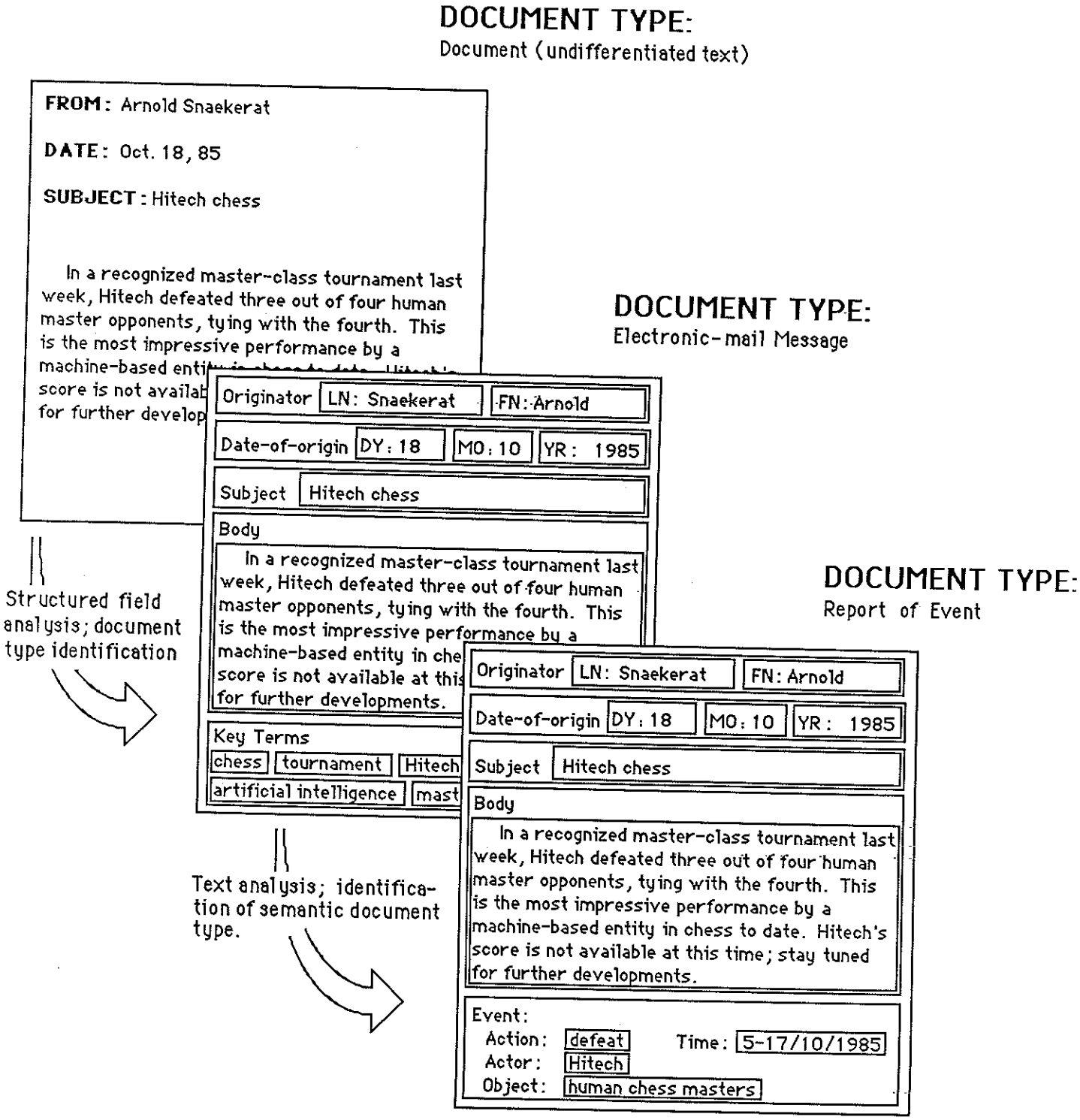
Objects

Specialist -- An expert in a particular field of inference. Each specialist has permissions to read only certain posting areas and to write to only certain areas. All specialists have permission to read and write to the question/answer area of the Priority Posting Area and to read the Pending Hypothesis area.

Subject Posting Area -- A set of (not necessarily consistent) hypotheses on a subject; ie, on a semantically limited set of rules and/or arguments. Subject Posting Areas and Specialists are in a many-to-many relationship.

Strategist -- An expert in moderating among the specialists.

Figure 3
AUTOMATIC IDENTIFICATION AND ANALYSIS OF COMPOSITE DOCUMENTS



reimplemented differently as the system is transported to different environments. These resource managers can be written in a lower-level language, such as C++, provided only that it has the requisite networking and pipe-manipulation primitives. The entire CODER system, thus, is a set of concurrent modules, executing on one or several machines, and connected together by pipes within machines and the TCP/IP protocols between machines. Some of these modules are procedural, some rule-based, and some coupled tightly to large databases, but all use a common interface paradigm, and all use a unified representation of the knowledge that the system applies to the task of information storage and retrieval.

5. Future Plans

The CODER system is being developed by a small core of graduate assistants, along with undergraduate and graduate students involved in information storage and retrieval courses. Initial processing of the CED and OALDCE should be completed by the spring of 1986. Coding of strategists and experts is beginning during the early part of 1986. A limited prototype should be operational by mid-1986, to be subsequently refined. By the end of 1986, a comparative study of information retrieval system effectiveness is planned on the AIList collection, rating CODER versus SMART.

Further development of CODER is planned throughout 1987. Also during that year it is hoped that special hardware will become available to support a high speed Prolog compiler and development environment.

Acknowledgements

The Oxford Text Archive and Melbourne University Dept. of Computer Science have supplied tapes with dictionaries and Prolog interpreters, respectively.

Numerous students in the last two years have played a role in the development of CODER. Muriel Kranowski studied hundreds of messages to identify message types and rules for recognition. Lee Hite and Mark Tischler have developed user interfaces using SUN Windows and CURSES software packages, respectively. Sachit Apte has explored problems with distributed processing. Joshua Mindel prepared a message pre-parser to support the indexing and search possible with SMART, which Sharat Sharan has kept up-to-date. Robert Wohlwend has effected the constrcution of the Prolog lexicon from the typesetting tapes of the major dictionaries. Pat Cooper and Joy Weiss have provided secretarial support.

REFERENCES

[AHLS 83] Ahlswede, Thomas E. A Linguistic String Grammar of Adjective Definitions from *Webster's Seventh Collegiate Dictionary*. In Williams, S. (Ed.). *Humans and Machines*. 1983, pp. 101-127.

[AHLS 84] Ahlswede, Thomas E. A Lexicon for a Medical Expert System. Presented at the Workshop on Relational Models (Coling84): a revised version is forthcoming.

[AHLS 85] Ahlswede, Thomas E. A Tool Kit for Lexicon Building. *Proceedings of the 23rd Annual Meeting of the ACL (July 8-12 ,1985)*. ACL, 1985, pp. 268-276.

[AMSL 80] Amsler, R.A. The Structure of the Merriam-Webster Pocket Dictionary. Dissertation. Univ. of Texas at Austin, Dec. 1980.

[AMSL 84] Amsler, R.A. Machine-Readable Dictionaries. *ARIST*, 19:161-209, 1984.

[ATTA 77] Attar, R. and Aviezri S. Fraenkel. Local Feedback in Full-Text Retrieval Systems. *J. ACM*, 24(3): 397-417, July 1977.

[BABA 85] Babatz, R. and M. Bogen. Semantic Relations in Message Handling Systems: Referable Documents. Presented at *IFIP WG 6.5 Symposium, (Sept. 1985)*.

[BART 83] Bartschi, M. and H.P. Frei. Adapting a Data Organization to the Structure of Stored Information. In Salton, Gerald and Hans-Jochen Schneider (Eds.). *Research and Development in Information Retrieval, Proc., Berlin, May 18-20, 1982*. Berlin: Springer-Verlag, 1983, pp. 62-79.

[BATE 78] Bates, Madeleine. The Theory and Practice of Augmented Transition Networks. In Bolc, L. *Natural Language Communication via Computers*. Berlin: Springer-Verlag, 1978.

[BELK 84] Belkin, N.J., Hennings, R.D., and T. Seeger. Simulation of a Distributed Expert-Based Information Provision Mechanism. In *Inf. Tech.: Res. Dev. Applications*. 3(3): 122-141, 1984.

[BICH 80] Bichteler, J. and Eaton III, E.A. The Combined Use of Bibliographic Coupling and Cocitation for Document Retrieval. *J. Am. Soc. Inf. Sci.*, 31(4), July 1980.

[BLAI 85] Blair, D.C. and M.E. Maron. An Evaluation of Retrieval Effectiveness for a Full-Text Document-Retrieval System. *Commununications of the ACM* **28**:3 (Mar. 1985), pp. 289-299.

[BOBR 85] Bobrow, D.G. If Prolog is the Answer, What is the Question? or What it Takes to Support AI Programming Paradigms. In *IEEE Trans. on Software Engineering*, 11(11): 1401-1408, November 1985.

[BOOK 80] Bookstein, A. Fuzzy Requests: An Approach to Weighted Boolean Searches. *J. Am. Soc. Inf. Sci.*, 31(4):240-247, July 1980.

[BORG 84] Borgman, Christine L. Psychological Research in Human-Computer Interaction. *ARIST*, 19:33-64, 1984.

[BUCK 85] Buckley, C. Implementation of the SMART Information Retrieval System. TR 85-686, Cornell Univ., Dept. of Comp. Sci., May 1985.

[BUSH 45] Bush, V. As We May Think. *Atlantic Monthly*, 176:101-108, July 1945.

[CHAN 85] Chandrasekaran, B. Generic Tasks in Knowledge-Based Reasoning: Characterizing and Designing Expert Systems at the "Right" Level of Abstraction. *The Second Conference on Artificial Intelligence Applications: the Engineering of Knowledge-Based Systems (Miami Beach, FL, Dec. 11-13, 1985)*: IEEE, 1985, pp. 294-300.

[CHAR 82] Charniak, E. Context Recognition in Language Comprehension. In *Strategies for Natural Language Processing*, ed. by Wendy G. Lehnert and Martin H. Ringle, Lawrence Erlbaum Assoc., Hillsdale NJ, 1982, 435-454.

[CHAR 83] Charniak, E. Passing Markers: A Theory of Contextual Influence in Language Comprehension. *Cognitive Science*, 7:171-190, 1983.

[CHOD 85] Chodorow, Martin S., Roy J. Byrd, and George E. Heidorn. Extracting Semantic Hierarchies from a Large On–Line Dictionary. In *Proc. of the 23rd Annual Meeting of the ACL*, 299-304, July 1985.

[CHOE 80] Choeka, Y. Computerized Full-Text Retrieval Systems and Research in the Humanities: The Responsa Project. *Computers and the Humanities*, 14(3), Nov. 1980, 153-169.

[COHE 85] Cohen, Jacques. Describing PROLOG by its Interpretation and Compilation. *Commun. ACM*, 28(12):1311-1324, Dec. 1985.

[COOK 84] Cook, P.R. Electronic Encyclopedias. *Byte*, 9(7):151-170, July 1984.

[DEJO 82] DeJong, G. An Overview of the FRUMP System. In *Strategies for Natural Language Processing*, ed. by Wendy G. Lehnert and Martin H. Ringle, Lawrence Erlbaum Assoc., Hillsdale NJ, 149-176, 1982.

[EARL 73] Earl, Lois. Use of Word Government in Resolving Syntactic and Semantic Ambiguities. *Inform. Stor. Retr.*, 9:639-664, 1973.

[ERMA 80] Erman, L.D., Hayes-Roth, F., Lesser, V.R., and D.R. Reddy. The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. *ACM Comp. Surveys*, 12:213-253, 1980.

[EVEN 79] Evens, M.W. and R.N. Smith. A Lexicon for a Computer Question-Answering System. *Am. J. Comp. Ling.*, Microfiche 83, 1979.

[FALO 85] Faloutsos, C. Access Methods for Text. *ACM Comp. Surveys*, 17(1):49-74, March 1985.

[FIKE 85] Fikes, Richard and Tom Kehler. The Role of Frame-Based Representation in Reasoning. *Communications of the ACM*, **28**:9 (Sept. 1985), pp. 904-920.

[FOXE 80] Fox, E.A. Lexical Relations: Enhancing Effectiveness of Information Retrieval Systems. *ACM SIGIR Forum*, 15(3):5-36, Winter 1980.

[FOXE 81] Fox, E.A. Implementing SMART for Minicomputers Via Relational Processing with Abstract Data Types. In *Joint Proc. of SIGSMALL Symp. on Small Systems and SIGMOD Workshop on Small Data Base Systems, ACM SIGSMALL Newsletter*, 7(2), Oct. 1981.

[FOXE 83a] Fox, E.A. Characterization of Two New Experimental Collections in Computer and Information Science Containing Textual and Bibliographic Concepts. TR 83-561, Cornell University, Department of Computer Science, Sept. 1983.

[FOXE 83b] Fox, E.A. Extending the Boolean and Vector Space Models of Information Retrieval with P-Norm Queries and Multiple Concept Types. Dissertation, Cornell University, University Microfilms Int., Ann Arbor MI, Aug. 1983.

[FOXE 83c] Fox, E.A. Some Considerations for Implementing the SMART Information Retrieval System under UNIX. TR 83-560, Cornell University, Department of Computer Science, Sept. 1983.

[FOXE 85a] Fox, E.A. Combining Information in an Extended Automatic Information Retrieval System for Agriculture. In *The Infrastructure of an Information Society* , ed. B. El-Hadidy and E.E. Horne (eds.), North Holland, 1984.

[FOXE 85b] Fox, E.A. Composite Document Extended Retrieval: An Overview. In *Res. & Dev. in Inf. Ret., Eighth Annual Int. ACM SIGIR Conf.*, Montreal, 42-53, June 1985.

[FOXE 86a] Fox, E.A. and S. Sharat. A Comparison of Two Methods for Soft Boolean Operator Interpretation in Information Retrieval, TR-86-1, Virginia Tech Dept. of Comp. Sci., Jan. 1986.

[FOXE 86b] Fox, E.A. Information Retrieval: Research into New Capabilities. In *The New Papyrus: CD-ROM*, Suzanne Ropiequet and Steve Lambert (eds.), Microsoft Press, 1986 (to appear).

[FOXE 86c] Fox, E.A. Improved Retrieval Using a Relational Thesaurus Expansion of Boolean Logic Queries. In *Relational Models of the Lexicon*, Martha Evens (ed.), Cambridge Univ. Press, 1986 (to appear).

[FOXM 80] Fox, M.S., D.J. Bebel, and A.C. Parker. The Automated Dictionary. *IEEE Computer*, 35-48, July 1980.

[FRAN 82] Francis, W. Nelson and Henry Kucera. *Frequency Analysis of English Usage: Lexicon and Grammar*. Boston: Houghton Mifflin Company, 1982.

[FREI 83] Frei, H.P. and Jauslin, J.F. Graphical Presentation of Information and Services: A User Oriented Interface. *Inf. Tech.: Res. Dev.*, 2(1):23-42, Jan. 1983.

[FREI 84] Frei, H.P. and J.F. Jauslin. Two-Dimensional Representation of Information Retrieval Services. In Dietschmann, Hans J. (Ed.). *Representation and Exchange of Knowledge as a Basis of Information Processes*. New York: North-Holland, 1984, pp. 383-396.

[FUJI 84] Fujitani, L. Laser Optical Disk: The Coming Revolution in On-Line Storage. *Commun. ACM*, 27(6):546-554, June 1984.

[GARF 78] Garfield, E. *Citation Indexing: Its Theory and Application in Science, Technology, and Humanities*. John Wiley & Sons, New York, 1978.

[GAZD 85] Gazdar, Gerald and Geoffrey K. Pullum. Computationally Relevant Properties of Natural Languages and their Grammars. Report No. CSLI-85-24. Stanford CA: Center for the Study of Language and Information, 1985.

[GENE 85] Genesereth, Michael R. and Matthew L. Ginsberg. Logic Programming. *Commun. ACM*, 28(9):933-941, Sept. 1985.

[GOLD 84] Goldstein, C.M. Computer-Based Information Storage Technologies. *ARIST*, 19: 65-96, 1984.

[HALL 80] Hall, P.A.V. and Dowling, G.R. Approximate String Matching. *ACM Comp. Surveys*, 12(4):381-402, 1980.

[HASK 80] Haskin, R. Hardware for Searching Very Large Text Databases. In *Proc. 5th Workshop on Comp. Arch. for Non-Numeric Proc.*, ACM, New York, 49-56, March 1980.

[HAYE 84] Hayes-Roth, Barbara. BB1: An Architecture for Blackboard Systems that Control, Explain, and Learn About Their Own Behavior. Technical Report No. STAN-CS-84-1034, Stanford University Dept. of Comp. Science, December 1984.

[HAYE 85a] Hayes-Roth, Barbara. A Blackboard Architecture for Control. *Artificial Intelligence* **26** (1985), pp. 251-321.

[HELM 85] Helm, A.R., Marriott, Kimbal, and Catherine Lassez. Prolog for Expert Systems: An Evaluation. *Expert Systems in Government Symposium (Maclean, VA: Oct. 24-25, 1985)*. IEEE, 1985, pp. 284-293.

[HOLL 85] Hollaar, L.A. A Testbed for Information Retrieval Research: The Utah Retrieval System Architecture. *Proc. 8th Annual. Int. ACM SIGIR Conf. on R&D in Inf. Ret.*, Montreal, 227-232, June 1985.

[HORA 84] Horak, W. and G. Kronert. An Object-Oriented Office Document Architecture Model for Processing and Interchange of Documents. In *Proceedings of the Second ACM-SIGOA Conference on Office Information Systems (June 25-27, 1984)*, pp. 152-160.

[HORA 85] Horak, W. Office Document Architecture and Office Document Interchange Formats: Current Status of International Standardization. *Computer* **18**:10 (Oct. 1985), pp. 50-60.

[HORN 74] Hornby, A.S. ed. *Oxford Advanced Dictionary of Current English*. Oxford: Oxford University Press, 1974.

[HULT 84] Hultin, N.C. and H.M. Logan. The *New Oxford English Dictionary* Project at Waterloo. *Dictionaries: Journal of the Dictionary Society of North America* **6** (1984), pp. 128;183-198.

[JENN 84] Jennings, M. The Electronic Manuscript Project. *Bulletin of the American Society for Information Science* **10**:3 (Feb. 1984), pp. 11-13.

[KATZ 82] Katzer, J., et. al. A Study of the Overlap Among Document Representations. Unpublished paper, Syracuse University School of Information Studies, 1982.

[KESS 63] Kessler, M.M. Bibliographic Coupling Between Scientific Papers. *Amer. Doc.*, 14(1), 1963.

[KIMU 84] Kimura, Gary D. A Structure Editor and Model for Abstract Document Objects. PhD Dissertation. Tech. Report No. 84-07-02, Dept. of Comp. Sci., Univ. Washington, July 1984.

[LEEN 85] Lee, N.S. P-Shell: A Prolog-Based Knowledge Programming Environment. AT&T Technical Report No. TM 59567-851201-01, December 1985.

[LEHN 78a] Lehnert, Wendy G. *The Process of Question Answering: A Computer Simulation of Cognition.* Hillsdale, NJ: Lawrence Erlbaum Assoc., 1978.

[LEHN 78b] Lehnert, Wendy G. Representing Physical Objects in Memory. Yale Artificial Intelligence Project Technical Report #131 (May, 1978).

[LESK 85] Lesk, M. Report on *Information in Data: Using the Oxford English Dictionary on a Computer. IRList Digest* **1**:28 (31 Dec. 1985).

[LEVE 84] Levesque, Hector J. A Fundamental Tradeoff in Knowledge Representation and Reasoning. *Proceedings CSCSI-84 (London, ON, May 1984):* pp. 141-152.

[MEAD 81] Meadow, C.T. and P.A. Cochrane. *Basics of Online Searching.* John Wiley & Sons, New York, 1981.

[MICH 71] Michelson et al. An Experiment in the Use of Bibliographic Data As a Source of Relevance Feedback in Information Retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing,* ed. G. Salton, Prentice Hall, Englewood Cliffs, NJ, 1971.

[MINS 75] Minsky, M. A Framework for Representing Knowledge. In Winston, P. (Ed.). *The Psychology of Computer Vision.* New York: McGraw-Hill, 1975.

[MORR 83] Morrissey, J. An Intelligent Terminal for Implementing Relevance Feedback on Large Operational Retrieval Systems. In *Res. & Dev. in Inf. Ret.*, Proc., Berlin, May 18-20, 1982, ed. by G. Salton and Hans-Jochen Schneider, Springer-Verlag, Berlin, 38-50, 1983.

[NAIS 85] Naish, Lee. *MU-Prolog 3.2db Reference Manual.* Melbourne University, 1985.

[OCON 80] O'Connor, J. Answer-Passage Retrieval by Text Searching. *J. Am. Soc. Inf. Sci.*, 31(4):227-239, 1980.

[ODDY 77] Oddy, R.N. Information Retrieval Through Man-Machine Dialogue. *J. Doc.*, 33(1):1-14, March 1977.

[PAIC 84] Paice, C.D. Soft Evaluation of Boolean Search Queries in Information Retrieval. *Inf. Tech.: Res. Dev. Applications,* 3(1):33-42, 1984.

[PATE 84a] Patel-Schneider, P.F., R.J. Brachman, and H.J. Levesque. ARGON: Knowledge Representation meets Information Retrieval. *The First Conference on Artificial Intelligence Applications (Denver, CO: Dec. 5-7, 1984):* IEEE, 1984, pp. 280-286. (Also: Fairchild Technical Report No. 654; FLAIR Technical Report No. 29, Sept. 1984).

[PATE 84b] Patel–Schneider, P.F. Small can be Beautiful in Knowledge Representation. *Workshop on Principles of Knowledge-Based Systems (Denver, CO: Dec. 3-4, 1984):* IEEE, 1984, pp. 11-16. (Also: FLAIR Technical Report No. 37, October 1984).

[PEEL 85] Peels, Arno J.H.M., Norbert J.M. Janssen and Wop Nawijn. Document Architecture and Text Formatting. *ACM Transactions on Office Information Systems* **3**:4 (Oct. 1985), pp. 347-369.

[PERE 83] Pereira, F. Logic for Natural Language Analysis. Tech. Note 275, SRI Int., Jan. 1983.

[PETE 82] Peterson, James L. *Webster's Seventh New Collegiate Dictionary:* A Computer-Readable File Format. Technical Report TR-196. Austin, TX: University of Texas at Austin, Department of Computer Science. May, 1982.

[PIGM 84] Pigman, Victoria. The Interaction Between Assertional and Terminological Knowledge in Krypton. *Workshop on Principles of Knowledge-Based Systems (Denver, CO: Dec 3-4, 1984).* Ieee, 1984, pp. 3-10.

[POLL 85] Pollard, Carl. J. and Lewis G. Creary. A Computational Semantics for Natural Language. *Proceedings of the 23rd Annual Meeting of the ACL (July 8-12 ,1985).* ACL, 1985, pp. 172-179.

[RIEG 81] Rieger, C. and S. Small. Toward a Theory of Distributed Word Expert Natural Language Parsing. *IEEE Trans. on Systems, Man, and Cybernetics,* SMC-11(1):43-51, Jan. 1981.

[RITC 74] Ritchie, D.M. and K. Thompson. The UNIX Time-Sharing System. *Commununications of the ACM* **17**:7 (July 1974), pp. 365-375.

[ROBE 76] Robertson, S.E. and K. Sparck Jones. Relevance Weighting of Search Terms. *J. Am. Soc. Inf. Sci.*, 27(3):129-146, 1976.

[ROCC 71] Rocchio, Jr., J.J. Relevance Feedback in Information Retrieval. In *The SMART Retrieval System, Experiments in Automatic Document Processing*, ed. by G. Salton, Prentice Hall, Englewood Cliffs, NJ, 1971.

[SACC 84] Sacco, G.M. OTTER - An Information Retrieval System for Office Automation. In *Proc. Second ACM-SIGOA Conf. on Office Information Systems (June 25-27, 1984)* pp. 104-112.

[SAGE 75] Sager, N. Sublanguage Grammars in Science Information Processing. *J. Am. Soc. Inf. Sci.*, 26(1):10-16, Jan.-Feb. 1975.

[SAGE 81] Sager, Naomi. *Natural Language Information Processing*. New York: Addison-Wesley, 1981.

[SALT 63] Salton, G. Associative Document Retrieval Techniques using Bibliographic Information. *J. Am. Soc. Inf. Sci.*, 10(4), Oct. 1963.

[SALT 83a] Salton, G. and M.J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, 1983.

[SALT 83b] Salton, G., Buckley, C., and E.A. Fox. Automatic Query Formulations in Information Retrieval. *J. Am. Soc. Inf. Sci.*, 34(4):262-280, July 1983.

[SALT 83c] Salton, G., Fox, E.A., and Wu. H. Extended Boolean Information Retrieval. *Commun. ACM,* 26(11):1022-1036, Nov. 1983.

[SALT 85] Salton, G., Fox, E.A. and E. Voorhees. Advanced Feedback Methods in Information Retrieval, *J. Am. Soc. Inf. Sci.*, 36(3):200-210, May 1985.

[SCHA 73] Schank, Roger C. and Kenneth M. Colby (Eds.). *Computer Models of Thought and Language.* San Francisco, CA: Freeman, 1973.

[SCHA 77] Schank, R.C. and R.P. Abelson. *Scripts, Plans, Goals and Understanding.* Hillsdale, NJ: Lawrence Erlbaum Assoc., 1977.

[SCHA 81] Schank, R.C., Kolodner, J.L., and G. DeJong. Conceptual Information Retrieval. In *Information Retrieval Research*, ed. R.N. Oddy et al., Butterworths, London, 1981.

[SHER 74] Sherman, D. A New Computer Format for *Webster's Seventh Collegiate Dictionary.* In *Computers and the Humanities,* 8:21-26, 1974.

[SIMM 84] Simmons, Robert F. *Computations from the English.* Prentice-Hall, Englewood Cliffs, NJ, 1984.

[SKUC 85] Skuce, D., S. Matwin, B. Tauzovich, S. Szpakowicz and F. Oppacher. A Rule-Oriented Methodology for Constructing a Knowledge Base from Natural Language Documents. *Expert Systems in Government Symposium (Maclean, VA: Oct. 24-25, 1985).* IEEE, 1985, pp. 378-385.

[SMAL 73] Small, H.G. Co-Citation in the Scientific Literature: A New Measure of the Relationship Between Two Documents. *J. Amer. Soc. Inf. Sci.*, 24(4), July-Aug. 1973.

[SMIT 84] Smith, Linda C. and Amy J. Warner. A Taxonomy of Representations in Information Retrieval System Design. In Dietschmann, Hans J. (Ed.). *Representation and Exchange of Knowledge as a Basis of Information Processes.* New York: North-Holland, 1984, pp. 31-49.

[SOWA 84] Sowa, John F. *Conceptual Structures: Information Processing in Mind and Machine.* Reading, MA: Addison-Wesley, 1984.

[SPAR 73] Sparck Jones, K. and Martin Kay. *Linguistics and Information Science.* New York: Academic Press, 1973.

[SPAR 84] Sparck Jones, K. and J.I. Tait. Automatic Search Term Variant Generation. *J. Doc.*, 40(1):50-66, March 1984.

[SUBR 85] Subrahmanyam, P.A. The "Software Engineering" of Expert Systems: Is Prolog Appropriate? *IEEE Trans. on Software Engineering* **11**:11 (Nov. 1985), pp. 1391-1400,.

[TENO 84] Tenopir, Carol. Full-Text Databases.*ARIST*, 19:215-246, 1984.

[THOM 85] Thompson, R.H. and W.B. Croft. An Expert System for Document Retrieval. *Expert Systems in Government Symposium (Maclean, VA: Oct. 24-25, 1985).* IEEE, 1985, pp. 448-456.

[TONG 83] Tong, R.M. et al. A Rule-Based Approach to Information Retrieval: Some Results and Comments. *Proc. AAAI-83,* 1983.

[TONG 85] Tong, R.M., V.N. Askman, J.F. Cunningham, and C.J. Tollander. RUBRIC An Environment for Full Text Information Retrieval. *Proc. 8th Annual Int. ACM Conference on R&D in Inf. Ret.*, Montreal, June 1985.

[VANR 79] Van Rijsbergen, C.J. *Information Retrieval: Second Edition.* Butterworths, London, 1979.

[VOOR 85] Voorhees, E.M. The Effectiveness and Efficiency of Agglomerative Hierarchic Clustering in Document Retrieval. Dissertation. TR 85-705, Cornell Univ., Dept. of Comp. Sci., Oct. 1985.

[WEYE 82A] Weyer, S.A. The Design of a Dynamic Book for Information Search. In *International Journal of Man Machine Studies*, 17(1): 87-107, July 1982.

[WEYE 82B] Weyer, S.A. Searching for Information in a Dynamic Book. SCG-82-1. Xerox PARC, Palo Alto, CA, Feb. 1982.

[WEYE 84] Weyer, S.A., and A.H. Borning. A Prototype Electronic Encyclopedia. In *ACM Trans. on Office Information Systems*, 3(1): 63-68, January 1984.

[WHIT 83] White, C. The Linguistic String Project Dictionary for Automatic Text Analysis. In Proc. *Workshop on Machine Readable Dictionaries,* SRI, Menlo Park, CA, May 1983.

[WILE 84] Wilensky, R., Arens, Y., and D. Chin. Talking to UNIX in English: An Overview of UC. *Commun. ACM*, 27(6):574-593, 1984.

[WINE 85] Winett, S. and E.A. Fox. Using Information Retrieval Techniques in an Expert System. *The Second Conference on Artificial Intelligence Applications (Miami Beach, FL: Dec 11-13, 1985).* IEEE, 1985, pp. 230-235.

[YIPM 79] Yip, Man-Kam. An Expert System for Document Retrieval. M.S. Thesis. M.I.T., 1979.