Information Processing & Management Vol. 23, No. 4, pp. 341–366, 1987
Printed in Great Britain.

0306-4573/87 $3.00 + .00

DEVELOPMENT OF THE CODER SYSTEM: A TESTBED FOR ARTIFICIAL INTELLIGENCE METHODS IN INFORMATION RETRIEVAL

EDWARD A. FOX
Department of Computer Science, Virginia Tech, Blacksburg, VA 24061

(*Received 25 February* 1987)

Abstract—The CODER (COmposite Document Expert/Extended/Effective Retrieval) system is a testbed for investigating the application of artificial intelligence methods to increase the effectiveness of information retrieval systems. Particular attention is being given to analysis and representation of heterogeneous documents, such as electronic mail digests or messages, which vary widely in style, length, topic, and structure. Since handling passages of various types in these collections is difficult even for experimental systems like SMART, it is necessary to turn to other techniques being explored by information retrieval and artificial intelligence researchers. The CODER system architecture involves communities of experts around active blackboards, accessing knowledge bases that describe users, documents, and lexical items of various types. The initial lexical knowledge base construction work is now complete, and experts for search and time/date handling can perform a variety of processing tasks. User information and queries are being gathered, and a simple distributed skeletal system is operational. It appears that a number of artificial intelligence techniques are needed to best handle such common but complex document analysis and retrieval tasks.

1. INTRODUCTION

Online searching of bibliographic data bases, originally an aid to research in areas like medicine and chemistry, has become increasingly important in serving the diverse needs of both the public and private sectors. With the spread of word processing and electronic publishing, a greater proportion of written materials now being produced is available in machine-readable form. The advent of full-text data bases [1], originally important primarily for legal research, has helped push the total number of publicly accessible online data bases above the 3,000 mark. If one includes corporate and private text collections, such as can develop from office information systems with a text filing capability, tens of thousands of searchable collections already exist.

1.1. End user searching

Machine-aided searching of early data bases began during the 1950s but was a cumbersome and expensive process. Today, with far-reaching networks connected to mainframe computer systems that manage vast banks of online storage or with powerful microcomputers controlling high-capacity CD ROM (compact disc read only memory) optical drives, individuals have the hardware tools to perform their own searches, but still only have fairly primitive software support.

Even in the domain of bibliographic retrieval, many end users prefer to locate interesting items without having to involve a search intermediary [2]. Not every user feels this need, but many search intermediaries also desire a simpler means of access to the multiple systems and data bases involved. Furthermore, in the context of office automation, individual office workers usually *must* search on their own.

Project funded in part by grants from the National Science Foundation (IST-8418877) and the Virginia Center for Information Technology (INF-85-016) and aided by an AT&T Equipment Donation.

In 1981 it was argued that searching methods should be improved and that additional research was of value [3]. Although many desirable qualities of a good interface had been identified and some experimental systems had been developed, access to information was difficult for the average user [4]. Since that time, knowledge of underlying retrieval principles has progressed to the point where textbooks applying those concepts to conventional online searching are available [5]. A few systems like PaperChase, which serves medical practitioners [6], are simple, well engineered, and powerful. Yet although there are now commercially available systems aimed at facilitating searching, and published lists of suggested techniques [7], there is still no truly helpful search software that can effectively meet the needs of most end users [8].

1.2. Need for improvement in information retrieval systems

One recent study indicated that only about 20% of the relevant items were found by careful searchers who used a commercially available full-text retrieval system [9]. Earlier work had shown the overlap between search results of different people to be small [10]. Putting these results together, one is not surprised that the effectiveness and efficiency of searching by a single individual seems to be rather low [11]. Since it is typically not possible to follow the obvious suggestion derived from these studies, namely to have several different searchers work on the same interest statement and then pool their results, it seems appropriate to consider a different approach, such as making a computer play the role of several (intelligent) searchers. In view of the advances made in automatic text retrieval systems [12], this should be feasible in the near future.

The opportunity of discovering how to dramatically improve retrieval effectiveness has challenged the body of researchers working on automatic indexing and retrieval systems and has led to the development of a variety of methods based primarily on statistical and probabilistic processing of text collections and user queries. While the marriage of these methods to powerful microcomputers and optical stores is now possible and would benefit many users [13], some researchers feel that more "intelligent" approaches are needed to provide even greater effectiveness [14]. Since artificial intelligence (AI) methods have begun to provide assistance in a variety of other complex tasks, the information retrieval (IR) problem is being re-formulated as one involving "knowledge" bases [15].

1.3. CODER and Prolog

The CODER project was proposed as a means of systematically investigating the use of knowledge-based approaches, to be implemented in Prolog, for handling the complex task of analysis and retrieval of composite documents [16]. Earlier Pollitt had suggested that logic programming methods in general, and Prolog in particular, should be applied to information storage and retrieval problems [17]. Prolog can be easily learned [18] and, although classified as an AI language, can be applied to a variety of problems [19]. Advanced texts on Prolog programming are now available, so the elegance of logic programming and the efficiency and metamathematical expressiveness of Prolog can both be properly unleashed [20]. Data base researchers find it appealing to bring together the flexibility and expressive power of Prolog and the efficiency of data base management systems [21]. A version of Prolog with a built-in data base capability, MU-Prolog [22], was therefore selected for use in CODER. Pattern matching (by unification), recursion, automatic backtracking, searching through sets of facts or rules, and list manipulation are additional features of Prolog that allow it to be easily adapted to handle lists of keywords, perform natural language parsing of queries, manage a relational data base describing document features, and make inferences about complex knowledge structures associated with the content of a text.

With such a powerful tool, it was necessary to carefully define the system requirements and problems to be addressed by CODER and to study the various related efforts under way in the IR and AI areas. These subjects are discussed in Sections 2 and 3, respectively. Section 4 follows, providing an explanation of the particular approach taken. Implementation details are then given in Section 5, and conclusions appear in Section 6.

2. PROBLEM

During the last two decades, the communication and computing technologies supporting computer-based message systems have matured to the point that most large computer systems, and many small systems, are involved in some type of networking. Thousands of computers are on far-flung networks, such as ARPANET, BITNET, MILNET, NSFNET, or USENET [23]. Only a few support remote logging on of users, or conferencing, but nearly all handle electronic mail.

2.1. Network mail

Following the lead of standards groups, like the National Bureau of Standards and IFIP Working Group 6.5, CCITT developed and approved the X.400 Message Handling Standard for later submission to the International Standards Organization [24]. In terms of the established ISO/OSI framework for open systems interconnection, X.400 dealt with the topmost, or application, layer. Users invoke the message-handling system (MHS) by talking to a user agent (UA). The UA in turn communicates with the message transport system (MTS), which can link together computers across the globe. Ultimately, one user's message follows the chain of UA-MTA-. . . -MTA-UA so that another user can receive it.

The X.400 model allows user agents to carry out other related tasks. Since many UAs will be part of office information systems, it seems appropriate for them to perform additional functions, such as filing and retrieval of recent or archived messages or other office objects [25]. Because any text file (or other file, such as an image or voice segment) can be sent as mail, it is crucial to consider the logical and conceptual, as well as the layout, structure of multimedia objects that are to be indexed, filed, and retrieved [26]. Of particular interest are relationships of inclusion or reference among messages [27]. Clearly, there is a need for IR support of mail-handling systems, and that must include the ability to analyze the document structure and type [28].

2.2. AI message examples

Figure 1 illustrates some of the types of documents and document components present in messages sent out over the DARPA Internet as part of recent issues of AIList Digest. An archive of these messages is being maintained as a test collection for CODER. Some queries have been collected from interested students, and others are available in the text collection itself in the form of questions sent in for distribution. Finding passages relating to these queries may require reasoning about message types, about message organizations, and about the composite data elements (e.g., names, dates, addresses, bibliographic entries) that appear in many documents.

Four representative messages from the collection were selected to serve as examples. Only the important portions of each are shown in Fig. 1. In A through C, only the initial portion of the message body appears, while in D the header is also included. Example A gives a bibliographic citation and an abstract that can be identified by their format and indenting. Example B makes use of embedding, which is one means of referring to another message. A memo-style seminar announcement is shown in example C to illustrate the importance of special tokens and layout. Here, the speaker's address is split between two lines of the heading. On the other hand, example D shows a different address form and uses block centering to separate the title and abstract.

Recognition of the various layout features is obviously valuable for understanding how to analyze the different kinds of text and data elements present (in similar fashion to the recognition problem for document images [29]). Justified, indented, centered, and multicolumned texts exist as a result of horizontal structuring, while vertical organization is governed mainly by spacing and separator lines. Capitalization, use of punctuation marks like the colon, and the presence of a variety of special tokens or lexical items are additional clues suggesting what conceptual organization is appropriate. Yet deduction of the conceptual structure of a document from this data also requires understanding of the syntax and semantics of names, titles, dates, times, addresses, bibliographies, meeting announcements, and other message components and types.

A) Citation, abstract, question about that article

RIT Researchers Find Way to Reduce Transmission Errors,
Communications of the ACM, Vol. 29, No. 7, July 1986, p. 702:

Donald Kreher and Stanislaw Radziszowski at Rochester Institute of Technology
...
Integer (Diophantine) equations are notoriously difficult to solve. Is this ...

B) Embedded prior msg, followed by commentary

Date: Mon 29 Sep 86 09:55:11-PDT
From: Pat Hayes <PHayes@SRI-KL.ARPA>
Subject: Searle's logic

Look, I also don't think there's any real difference between a human's
...
At one end of the human knowledge spectrum we have that knowledge of a ...

C) Seminar, with memo style header, abstract

Title: Learning Apprentice Systems
Speaker: Prof. Tom Mitchell, Carnegie-Mellon University
Location: Rm. 2324 Dept of CS, U of MD, College Park
Time: 4:00pm
We consider a class of knowledge-based systems called Learning
Apprentices: systems that provide interactive aid in solving some problem, ...

D) Header, explanation, address, dissertation title and abstract

Date: Thu, 9 Oct 86 10:21:18 EDT
From: "Charles W. Anderson" <cwa0%gte-labs.csnet@CSNET-RELAY.ARPA>
Subject: Dissertation - Multilayer Connectionist Learning

Fig. 1. Examples of AIList message contents.

While context-free grammars are often used for parsing simple languages, it may be appropriate to use more powerful notations, like that described by Kimura [30], to characterize the various classes of documents. Owing to the complex interplay of layout, syntactic, semantic, pragmatic, and discourse rules, a blackboard (see the end of Section 3.2) rather than a direct parsing approach seems necessary to solve the analysis problem. Simpler specialized grammars can then be used by different experts to parse particular instances of the various types of document components (e.g., bibliographic entries). Finally, after this decomposition it is possible to index entire documents, parts of documents, and low-level structures like the bibliographic citation at the top of example A of Fig. 1. In the context of probabilistic retrieval, a scheme has already been proposed for adapting to different components [31]. Some of the passage retrieval techniques mentioned by O'Connor [32] may also be applicable to identify coherent blocks that should be indexed as a unit.

2.3. Limitations of SMART handling

In the vector-space retrieval model, each document is represented by a list of concepts and weights. This perspective has provided insight for a variety of studies undertaken with the SMART system [33]. SMART was extended in 1982 so that several kinds of concepts could be used to represent the different aspects of composite documents; instead of a single vector, multiple subvectors were allowed [34]. Based on the notion of multiple concept types, an AIList digest message can be indexed or inquired about according to the scheme shown in Fig. 2. The digest header is represented by concept types 0–2; parts of the message header are represented using concept types 3–5; and the subject line and body of the message are represented with type 6. The result of a document preparsing phase applied to the message used in the last example of Fig. 1 is shown in Fig. 3. While the heading is analyzed in a reasonable fashion, all structure present in the body of the message is lost by the current analysis scheme used in SMART. Although parsing the message body prior

FIELD LETTER (& explanation)	EXAMPLES / Indexing Instructions
.WHEN DIGEST ISSUED	(any format: Jan. 1, 1986, or 1/01/86, etc.) Concept type: 0 Parsing: full Token-type: date
.VOLUME NUMBER	(a known volume number: 1, 2, 3, 4,...) Concept type: 1 Parsing: full Token-type: number
.U (ISSUE NUMBER)	(a known issue number: 34, 87, etc.) Concept type: 2 Parsing: full Token-type: number
.DATE MESSAGE SENT	(any format: Jan. 1, 1986, or 1/01/86, etc.) Concept type: 3 Parsing: full Token-type: date
.NAME OF SENDER	(first and/or last name(s): Roger Schank) Concept type: 4 Parsing: full Token-type: name
.A (network ADDRESS of sender)	(e-mail address: benda@usc-isi.arpa) Concept type: 5 Parsing: token Token-type: token
.SUBJECT	(term(s) in message heading: case grammar) Concept type: 6 Parsing: full Token-type: word, proper-noun
.BODY	(term(s) in message body: linguistic methods) Concept type: 6 Parsing: full Token-type: word, proper-noun

Fig. 2. Query input form, annotated with indexing instructions.

to SMART processing is possible, there is no easy way to represent the relationship between document components. If one large vector is built, the structural relationships are lost. On the other hand, if a different vector is built for each component, it is awkward in a vector scheme to relate the various vectors so that, for example, characteristics of an object are inherited by another object that is part of the first.

All in all, while SMART has served as a testbed for statistical retrieval methods, it cannot be adapted easily to the AIList Digest collection. It is hoped that CODER will aid in such processing and become a widely used testbed for AI applications to IR.

3. RELATED WORK

In designing CODER so as to solve the problem of handling composite documents, a variety of related efforts were surveyed. Since a fairly extensive bibliography has been provided elsewhere [35], comments in this section will be more focused. For simplicity, the discussion will be divided somewhat arbitrarily into sections on IR, AI, and related systems.

3.1. IR research

There are many IR investigations that have motivated the CODER effort. Of greatest importance are those relating to the problems of combining similarities, finding related terms, selecting and efficiently using a search method for interactive retrieval, presenting results, and truly serving end user needs.

First, it is clear that CODER should be able to combine different forms of evidence when comparing an interest statement representation with a document representation. Bichteler and Eaton [36] have shown the value of this approach for using bibliographic coupling and cocitations. That work was extended by Fox [37,38] to consider terms, biblio-

.I 5495
.W **<When was digest sent?>**
Friday, 10 Oct 1986
.V **<Volume of AIList>**
Volume 4
.U **<issUe of AIList in current volume>**
Issue 211
.D **<Date author sent in message to digest editor>**
Thu, 9 Oct 86 10:21:18 EDT
.N **<Name of message author>**
Charles W. Anderson
.A **<Address of message author>**
cwa0%gte-labs.csnet@CSNET-RELAY.ARPA
.S **<Subject field of message>**
Dissertation - Multilayer Connectionist Learning
.B **<Body of message>**
The following is the abstract from my Ph.D. dissertation completed in August, 1986, at the University of Massachusetts, Amherst. Members of my committee are Andrew Barto, Michael Arbib, Paul Utgoff, and William Kilmer. I welcome all comments and questions.

Chuck Anderson
GTE Laboratories Inc.
40 Sylvan Road
Waltham, MA 02254
617-466-4157
cwa0@gte-labs

Learning and Problem Solving
with Multilayer Connectionist Systems

The difficulties of learning in multilayered networks of computational units has limited the use of connectionist systems in

Fig. 3. Output of SMART preparser, ready to be indexed.

graphic coupling, cocitations, direct citations, author names, and so on, and again led to improvements beyond using terms alone. Additional experiments are in progress to further validate these findings and to discover rules for computing similarities for such composite objects.

Second, it is obvious that a thesaurus would be of value. It is not obvious, however, how to (semi)automatically build a good thesaurus for a particular document collection or how to later use it effectively [39]. In the interest of generality it was suggested that a machine-readable dictionary [40] be used as a starting point. An initial experiment showed a small improvement in performance when queries were expanded by having terms added that were lexically or semantically related to low-frequency terms in the original query [41]. These results were confirmed by Wang et al. [42] and Evens et al. [43] and replicated again in the context of expanding extended Boolean queries [44]. On the basis of lexical/semantic relations, a more comprehensive lexicon could be produced [45], and one is being developed by Evens and co-workers. Their current version is based on *Webster's Seventh Collegiate Dictionary* [46]. Williams et al. [47] noted that associative aids are easy and effective to use in retrieval; such links are explicitly represented in a relational lexicon. Indeed, there are many uses of comprehensive lexicons [48], including to aid in natural language processing. Walker explains the importance of these and other types of knowledge resources [49].

Third, it is known that improvements in effectiveness can result from combining the results of a number of good searches made to answer a given question [10]. Some of this benefit can be obtained through the use of simple feedback methods [50]. Yet, since users can only examine details of one document at a time, incremental searching is really all that is required. One way to achieve this might be to repeatedly select a good search strategy [51], find a few new items, test their appropriateness, and display what appears to be the

best entry in the pool of retrieved/not-presented items. Another possibility is to view probabilistic feedback [52] as a sequential learning process [53], which might be implemented using heuristics so that documents are actually retrieved one by one with a reduced computational load at each step. It is also conceivable that optimization methods extending those described in [54] could be developed to reduce the number of inverted file accesses required for such truly interactive search sessions.

Fourth, several systems give users other opportunities to interact besides supplying a Boolean query [55]. CALIBAN made use of high-quality raster displays with windows [56]. Korfhage [57] suggested that users could navigate through a document vector space. Crouch [58] surveys a variety of methods for portraying such complex structures. Miller [59] also suggested browsing, but between words and word senses instead of documents.

Finally, it has become clear that it is important to understand more about how end users interact with retrieval systems. It is necessary to go beyond the anecdotal stage if computers are to behave more intelligently [60]. Rouse and Rouse [61] review key concepts relating to the general problem of human information seeking. Markey [62] extends the theory of question negotiating proposed earlier by Taylor, recognizing the potential involvement of a human intermediary throughout the process of proceeding from a "visceral" need to a question. Morrissey et al. [63] call for a model of querying, suggesting that rules are needed to indicate what actions a system should take for any given state of affairs. Ingwersen [64] gives a detailed analysis of a set of library searches, from a cognitive perspective, focusing on the dialogue, knowledge structures, and sequencing of search routines. For certain types of searchers in particular, more sophisticated browsing and feedback methods are desirable [65].

Findings regarding human-computer interaction can be applied [66] to help with designing systems for easier use [67]. The results of this approach are promising [68]. However, most of the research reported presumes use of conventional systems that require submission of Boolean queries [69]. It appears that with those systems experienced users have an edge, that precision is easier to achieve than recall, and that most searches are very simple [70]. Significant improvements might then be possible if expertise is brought to bear on important matters, such as finding search keys [71]. On the other hand, Bates [72] suggests that the overall process must be considered from a user perspective. Important phases of the process, like "finding the barn door" and "docking into a close connection," would then be better supported, and users would have more effective systems.

Daniels considers user modeling in the light of developments in cognitive science [73]. Studies of human-human interaction suggest characteristics of the user modeling function [74] in the overall context of human-computer problem solving [75]. Ultimately, these findings may be used to develop an intelligent intermediary that emulates the behavior of a human aid [76]. A distributed expert model for this process [77] seems particularly appropriate in light of the different tasks simultaneously performed by a human intermediary.

3.2. AI research

Many of the applications of AI to IR that Smith discussed in 1980 [78] have been studied by AI and IR researchers during the intervening period. Of particular interest are natural language processing (NLP), knowledge representation, time reasoning, and expert systems.

Linguistics has long been thought to be of value for information retrieval [79]. A variety of retrieval devices can be used to resolve problems relating to our use of natural language texts [80]. It is not clear, however, exactly what role should be played by each of the broad types of NLP that are possible [81]. Although Prolog can be used to parse large subsets of natural language and to build parse trees and other structures [82], matching of such structures is problematic. On the other hand, conversion of text descriptions to more structured forms *does* seem particularly useful [83]. Techniques for analyzing medical reports that use a specific sublanguage are also well established [84]. The resolution of anaphoric references in general documents might be of value, but no significant effect has yet been demonstrated [85]. On the other hand, some gain is possible when queries are parsed to obtain phrases that appear in documents [86]. More recently, "shallow" parsing

of documents, along with more thorough query parsing, has been shown to yield significant improvements in retrieval effectiveness [87]. Further development of syntax-based parsing schemes, especially in the context of office filing and retrieval, is under way [88].

A fundamental aspect of AI is the representation of knowledge. Many schemes have been proposed and several have clear application to IR. Frames are useful for representing objects [89] and can be manipulated in any one of several frame languages that support a variety of applications [90]. Thus, at the National Library of Medicine, a prototype system for computer-aided indexing produces not only MeSH keywords, but also a set of instantiated frames [91].

Rules are probably the easiest knowledge representation form to work with, can be coded by experienced programmers using Prolog or other special languages, and are present in many expert systems [92]. Semantic networks are helpful in situations where multiple associations exist. Thus, Simmons [93] describes early work with hand coding into networks of propositional knowledge from the first fifty pages of the *Handbook of Artificial Intelligence*. Regardless of the implementation scheme used, however, tools for creating knowledge bases from natural language text are at best semiautomatic [94].

Since many retrieval questions ask for documents that appear during a known time interval or that relate to events occurring on a certain date, it is important to have some temporal reasoning capability. Allen [95] presents an interval-based temporal logic with efficiency that can be improved through the use of a hierarchy of reference intervals. Mays [96] deals with temporal reasoning in a natural language front end to a data base system. Zarri [97,98] indicates how to apply temporal representation and reasoning to the analysis and retrieval of biographical information about French historical figures.

Expert systems are being constructed for a wide variety of tasks. To simplify development, it has been suggested [99] that generic experts with special languages be developed for a small number of common tasks, such as classification [100] or abductive assembly. Prolog can be used to develop expert systems if one is experienced with the language [101]. Uncertainty can be represented by adding confidence values to all parts of all rules [102] and by having appropriate computation routines. Alternatively, a meta-interpreter can be developed that hides this processing and so can readily switch to a different method for reasoning under uncertainty [103]. This flexibility may be necessary for experimental work since there are many different schemes, each with its own advocates [104].

To simplify the construction of expert systems in very complex domains, it is convenient to have a central blackboard with a variety of areas that can be examined and written on by experts. Although blackboards were first used to help with speech recognition [105], they have since been adapted to many different applications [106]. The theory and practice of blackboard use are carefully discussed in [107].

3.3. Related systems

Several different systems relate to particular aspects of CODER. First, there is the issue of analyzing documents in a more comprehensive fashion than is done by automatic indexing. The FRUMP system could skim newspaper stories by filling in sketchy scripts (which are similar to frames) [108]. Its speed, which would be useful in CODER when processing moderate-size text collections, is achieved because only a partial parse is done. As an extension of FRUMP, the FERRET system has been proposed, to be applied to electronic mail messages [109]. However, the primary aim is to demonstrate feasibility of the sketchy script approach to this type of information analysis and retrieval. TOPIC is another system where document analysis is involved, carried out by a word expert parser that helps determine the topical structure of a text [110]. In CALIN, which allows interactive entry and querying of advertisements, a partial parse of the input, along with data from user replies to questions, leads to direct recording of ad representations in the data base [111]. IOTA is a full-text indexing and retrieval system that exploits the logical structure of a text so that each structural entity (at any level in the overall hierarchy of entities) can have its noun phrases indexed. A conceptual pattern match can take place later when queries are parsed [112].

Second, there is the approach of using AI knowledge representation schemes to describe documents. ARGON uses frames in a computationally efficient manner to handle classification of objects and queries, and determines matches according to frame subsumption hierarchies [113]. COREL uses frames to conceptually index legal articles, which can later be retrieved with the aid of discrimination nets, through a menu interface [114]. Standard Prolog relations can also be of use, as in PROBIB-2, which records key attributes of documents so that matching can take place later, after a simpler parser analyzes user queries [115].

Third, there are systems with special query forms and means of ranking documents. RUBRIC provides tools for users to develop comprehensive queries that are in the form of tree-structured knowledge bases [116,117]. At the base of each tree are terms connected with Boolean and proximity operators that can be matched against documents. Given a particular selection of an allowable scheme for rule-based inference under uncertainty, the system can compute a certainty value indicating how well the query can be inferred from the document.

Fourth, there are systems that provide a natural language front end to a commercial retrieval system. CITE, which was developed at the National Library of Medicine, represents this class of systems [118].

Fifth, it is possible to combine natural language processing (as in group 4) with special query evaluation methods. CALIN, IOTA, and PROBIB-2, all mentioned above, have a natural language query-handling capability and distinctive document representation schemes. In addition, Biswas et al. [119,120], in their work on knowledge-assisted document retrieval, consider both the natural language interface and the retrieval components. They developed a modular design, and plan to carry out a variety of experiments with the system. Their natural language interface can handle a restricted query sublanguage through its augmented transition network and can determine the number of documents desired, the time range of interest, and the subject matter or content [119]. The retrieval component uses fuzzy set theory and one of several combination of evidence schemes [120].

Sixth, there are expert systems helping with the retrieval process. Perhaps the earliest was implemented in connection with the CONIT system, to help with the construction of Boolean queries [121]. Building on extensive experience in this area, Marcus [122] surveys work on expert retrieval assistance. In 1983, Guida and Tasso [123] reported on IR-NLI, an expert intermediary with a natural language interface with modules for reasoning and for understanding and dialogue. The former draws upon expert rules and domain-specific knowledge, while the latter uses linguistic rules and a dictionary. An internal representation of the processing state is the basis for determining what rules can be applied and what actions should be taken. Also described in 1983 was CANSEARCH, which used a blackboard architecture to help end users search the cancer therapy literature [17]. The Prolog implementation referred to frames when helping users identify useful MeSH terms and then prepared searches that were automatically run on MEDLINE [124].

Finally, there are intelligent retrieval systems with user modeling capabilities. IRES also includes natural language query processing, and it is adaptable to users and the state of processing [125]. There is emphasis on an independent thesaurus and on using an expert system. After morphological and syntactic query analysis, the indexing terms are accessed, results are combined and assigned values, and re-formulation takes place as needed. A rather different kind of knowledge-based system that could adapt to users is described in [126]. MINDS operates as a distributed query-handling system on a collection of office workstations and can adapt to user preferences as heuristics are developed.

I^3R [127,128] is the most comprehensive and extensible of all the systems considered. It is blackboard-based, with a scheduler to control the use of the computer's resources. An interface manager refers to the knowledge base and the state of the blackboard during its interaction with a user. The knowledge base contains documents and their representations, models of users, and session histories. A thesaurus contains generally accessible domain knowledge, in the form of concept frames that have the concept name, rules for recognizing it in text, and a description of its relationships to other concepts. There are experts for user

and request model building, indexing, search, browsing, and explanation, as well as for the management of the thesaurus. I^3R uses certainty values on rules in a fashion similar to RUBRIC and can retrieve on concepts using the probabilistic model.

4. APPROACH

It is clear from the preceding section that many research efforts are now under way to explore various applications of AI methods to IR problems. Of particular importance, then, is to carry that work beyond the stage of demonstrating feasibility to the point of careful scientific evaluation and integration of the most productive approaches. It is precisely this goal that is addressed by the CODER project—to develop a testbed for AI methods in IR. Provision has been made to allow gradual inclusion of the most important features of earlier efforts. A careful design and a comprehensive support environment were needed so that in the future, as the system is expanded beyond its current skeletal form, it would be possible to compare different approaches and rules in a controlled manner.

The plan for CODER was to ultimately use logic programming, blackboard-based expert systems methods, natural language processing, several knowledge representation schemes, heuristics for applying search methods, planning of resource utilization, temporal reasoning, and user modeling. Logic programming methods are involved through the use of MU-Prolog with its built-in data base support. Expert systems are present in both the document analysis and the retrieval communities of experts. Natural language processing is supported through the lexicon and, when further developed, should play a larger role in both query and document analysis. Knowledge representation is organized by a knowledge administration complex, which allows the creation and manipulation of types for elementary objects, frames, and relations. Although limited at present, searching capabilities will be expanded to provide rapid feedback in the event of any clues and will be tuned through the application of heuristics that should make the process much more efficient. Planning is embodied in the strategist that deals with handling resources on multiple computers. Temporal reasoning, although now at the level of handling dates and times and short phrases, will be extended to deal with events appearing in documents and queries. User modeling will eventually relate to the human-computer interaction aspects, to dealing with term expansion, and to helping control all aspects of the retrieval process.

CODER has a number of distinguishing characteristics. To serve as a testbed, it is modular so as to be adaptable to different theories and can be experimented with easily because modules can be quickly changed or replaced. CODER is a stand-alone system, used to manipulate raw documents and communicate directly with users, without an external data base or indexing facility. It is designed from the ground up to operate on multiple computers and to benefit from whatever level of concurrency is achievable. Further, it is unique in focusing on composite documents, including their structure, basic data types, and interrelationships. It is hoped that CODER will also evolve to become a comprehensive system: in dealing with document analysis that yields vector and AI representation schemes; in building and applying knowledge about users; in supporting natural language processing with a large lexicon; in handling documents or passages; and in applying expertise to search and other tasks.

4.1. Evolution of CODER project

Originally proposed in 1984, the CODER effort began with studies of different document components and with methods for combining them [16]. Some 1,000 documents were examined to develop heuristics for determining document types. The lexicon was begun, and MU-Prolog was chosen as the main programming language (to be supplemented with C whenever necessary). Issues relating to the design have been discussed elsewhere [129]. Encouraged by the findings of Belkin et al. [130], a blackboard orientation was planned. In addition to ongoing development by graduate assistants, students in IR courses in AY 85/86 were assigned small parts of the system, and that process will continue in 1987 as well. Thus, work on CODER has both educational and research aspects.

Specifications were completed and collected together by France [131]. A system overview and description of the overall architecture has been provided by Fox and France [35]. Key features of the design are explained in the next two subsections, as preparation for discussing the current implementation in Section 5.

4.2. *Blackboard-based development*

As mentioned, a blackboard-based design was chosen, to provide maximum flexibility and to ease integration of components. A blackboard for the retrieval function and another blackboard for the document analysis task were both needed, surrounded by different groups of experts to address the tasks required. Figure 4 provides a high-level overview of the system, but owing to space limitations does not show the analysis blackboard and related experts.

Each blackboard can be subdivided into two parts: the blackboard proper and the strategist. The blackboard portion has subject areas for each major class of information, and two priority areas. While experts may be restricted to examining and writing to certain subject areas, all experts can access the priority areas. Thus, questions and answer sets are present in the question/answer area when experts need or can offer information. In addition, pending hypotheses (e.g., that a particular document might be relevant or that a user has a certain level of knowledge in a topical area) are posted in the pending hypothesis area when the strategist notices a hypothesis (or collection of related hypotheses) in some subject area that has a high confidence value. Relating this to the retrieval problem, it should be noted that since high levels of recall are often difficult to achieve, the possi-

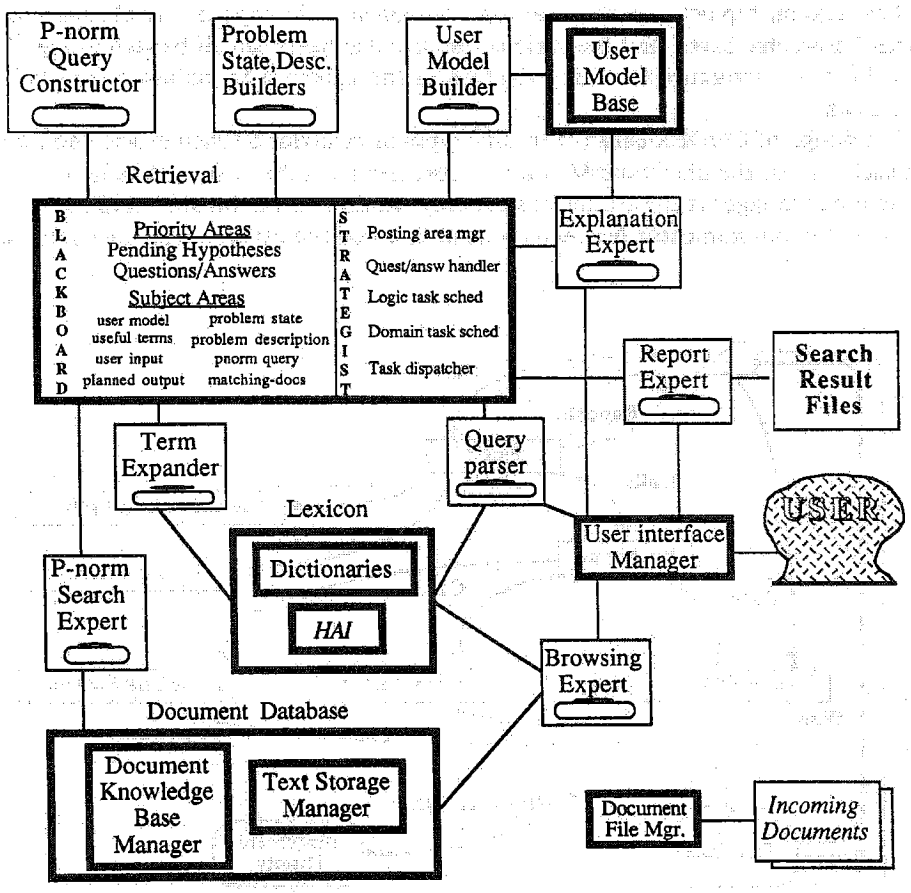

Fig. 4. Overview of CODER system. The diagram is an approximate representation, aimed at conveying essential features rather than exact details. Owing to space limitations, the analysis/blackboard strategist and related experts are not shown.

bility of having multiple interpretations and hypotheses should allow query splitting and other methods of expanding term sets and queries.

The strategist has five components: (1) to manage the aforementioned posting areas, (2) to identify experts and priorities relating to handling a new question, (3) to maintain dependency chains between hypotheses (in case of later retraction), (4) to select experts and priorities that are needed based on the current phase of processing, and (5) to actually "wake" and otherwise control experts.

All user interaction is through the user interface manager. Special commands for analysis or retrieval can be given and are handled by the command parser. A report expert can cause display or filing of results. Explanations are based on the current user and on the blackboard state. Browsing is possible of both the document data base and the lexicon. The user model builder updates the user model base as a result of events on the blackboard.

Document analysis begins when a command is received by the analysis blackboard. The document file manager affords access to incoming documents, which in raw form are handled by the text storage manager. Analysis of temporal references and document type is handled by specialists, while the document analyzer performs the general tokenizing, parsing, and knowledge structure building (that leads to entry in the document knowledge base).

Retrieval is prompted by an explicit (or default, from the user model base) query. User model building, problem state transformation, and building of the problem description all proceed. When some terms are available, the lexicon can be accessed by a term expander to obtain other related terms that can be browsed or automatically used to help construct a query. Eventually a p-norm or other query is constructed, a search is made, and a report is prepared for the user.

The relationship between the strategist components, the experts, the blackboards, the external knowledge bases, and the various resource managers can all be seen in Fig. 5. The almost loop-free structure simplifies the task of the strategist by reducing the likelihood of deadlock.

The design of CODER calls for specific types of behavior by each expert and indicates what each part of the blackboard/strategist complex must do. It should be noted that the posting area manager is the strategist's primary window to the outside, and the task dispatcher is the communicator between the strategist and the experts. The question/answer

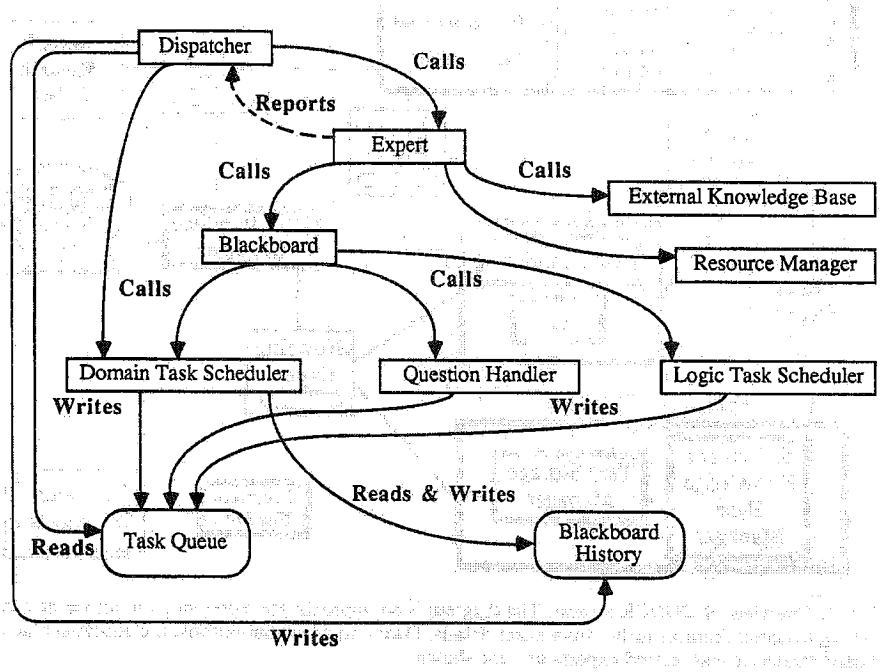

Fig. 5. Detailed calling hierarchy for CODER community of experts.

handler and the domain task scheduler together provide the real knowledge-based control of system operation, which is most restrictive if only one processor is involved and is minimal when many processors are available so that every expert can spot and perform work as soon as possible.

Since CODER is a message-passing system that can be distributed across machines, the TCP/IP protocol suite is used for communications. A client/server model allows each server to queue up pending requests that can be dealt with later. Information flows between sockets, but these details can be hidden from most developers. The communication facility is provided as a result of modifications made locally to the MU-Prolog interpreter.

A client routine must indicate what type of service is desired. A map on the client's computer is consulted to determine what machine and what process on that machine is available as a server. Then the client's request is converted to a character-string form, transmitted using TCP/IP, and converted back to its original form by the server. In the case of a blocking call, the client waits for processing by the server to complete before proceeding. More often, however, the client receives an immediate acknowledgment and can proceed in parallel with the server. This is especially useful when a client requests proof of a Prolog goal, which may involve a large number of inferences, before a reply with suitable answers (i.e., variable substitutions) can be made.

4.3. Knowledge engineering

A second key aspect of the approach taken in CODER is to use knowledge whenever possible. Behind the scenes, a knowledge engineer must develop a type system for each subject domain, so that objects/entities/events and relationships are properly described (by frames and relations, respectively) for matching and reasoning. For example, in the application domain of document retrieval, where citations to various objects are abundant, the frame hierarchy shown in Fig. 6 is valuable for classification.

It should be noted that the hierarchy shows "a kind of" (AKO) relationship, so that a "Proceedings" is AKO "Reports" which is AKO "Publications...." What this means is that all slots of a parent frame are inherited by the child type. By using strict typing, the computational complexity of matching is drastically reduced [132]. Since these operations may be performed on the large fact stores in the external knowledge bases, efficiency is an important consideration.

For document analysis it is vital to rapidly determine the type of given digest message, according to the breakdown in Fig. 7. Each type can be identified by applying a set of heuristics (see Fox [28] for an example), and frames can then be filled in with distinguishing characteristics.

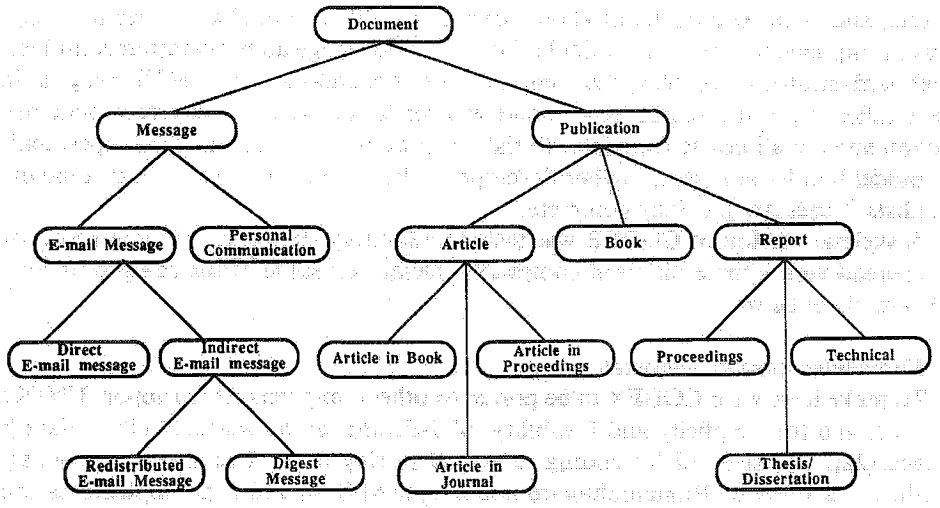

Fig. 6. Type hierarchy for common classes of documents.

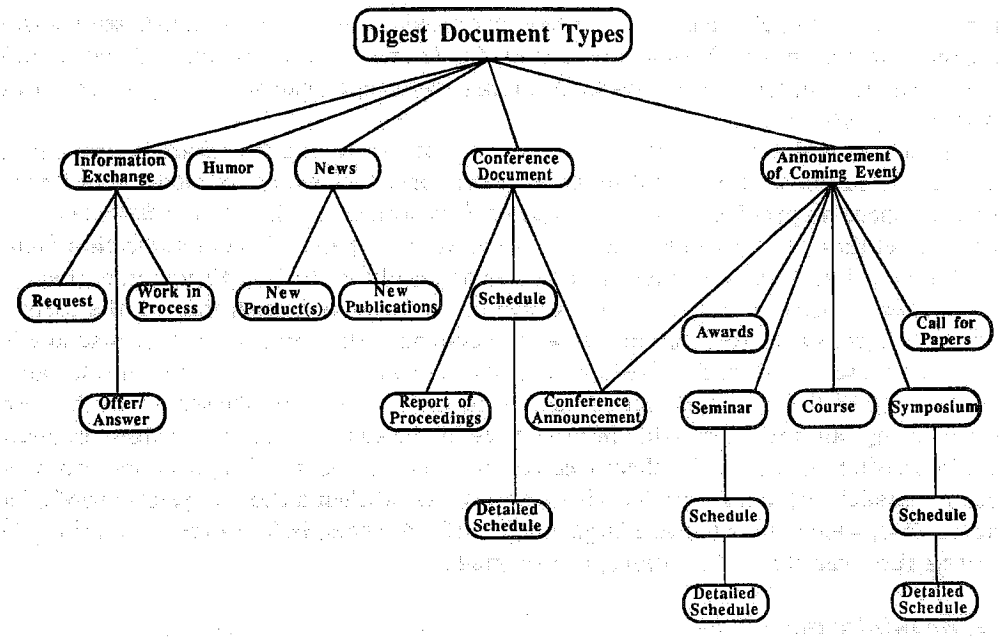

Fig. 7. Topical hierarchy for digest documents.

For query processing, it is crucial that the concepts involved are identified in such a way that matching with document representations is possible. CODER's lexicon and domain-specific knowledge base, discussed in the next section, have been developed to support this goal. Initially, documents will be indexed in SMART-like fashion, and queries will be expanded using available knowledge. Later, conceptual clusters in documents will be recognized and matched against corresponding query constructs. It has been shown that retrieval of certain AIList messages may indeed require this approach [133].

While the external knowledge bases store the factual information resident in the system, the various experts each have local rule bases that can be used for forward or backward-chaining purposes involved in their particular tasks. Some commonality among experts is possible, especially when similar tasks (e.g., classification) are involved.

5. IMPLEMENTATION

Since development of CODER involves research assistants, students working on MS projects, and students completing class projects, it is difficult to characterize precisely the status of implementation. The initial lexicon, the knowledge administration complex, the blackboard/strategist complex, the communications enhancements to MUProlog, a time/date handler, a p-norm search expert, and two versions of the user interface manager do function and are all nearly complete. Initial versions of the document-type expert and the user model builder are being further developed. The document analyzer and some of the specialists it uses are partially complete.

A skeletal version of CODER was tested in January 1987 and operated successfully when spread across three different computers. Details on some of the completed components are given below.

5.1. Knowledge-base development

To make it easy for CODER to be ported to other computers (that support UNIX and TCP/IP), and for simplicity and flexibility, MU-Prolog, along with its built-in data base package [22], was selected for coding rules and storing facts. The data base package is described in a study by Ramamohanaro and Shepherd [134] and is an implementation of one scheme for superimposed coding [135].

A variety of types of data have been stored in the form of Prolog facts, as can be seen in Table 1. Part A shows the current document collection, which is already about as big as the largest document collection used in earlier studies (i.e., the INSPEC collection). Statistics are based on the SMART form, since searching and data-collection routines are in use on that system.

Parts B and C relate to the two parts of the lexicon. The text of the *Handbook of Artificial Intelligence* (*HAI*) is available online, and various kinds of information have been extracted from it. First, to help provide a hierarchical organization to the AI discipline, the table of contents of the three-volume work is shown in Table 2.

Second, the initial portion of a more detailed topical hierarchy is shown in Table 3. Internally, a list structure with embedded sublists is used to represent the hierarchical organization, and the correspondence of subject names and numbers is stored in a Prolog relation. However, for external presentations like that of Table 3, an outline structure is more appropriate. Thus, the organization of the first part of Volume I is represented by indenting once for chapter, twice for section, three times for subsection, and so on. A searcher can browse through the subject heading structure of the *HAI*, to become more familiar with the organization of its knowledge about AI.

Third, the entries derived from the back of the book index entries are described in Table 4. There are three relations involved. Some index items refer to a single text line in a particular file and are handled by the "index_ref" relation. Other index items refer to a range of lines and are handled by the "index_rng" relation. For ease of use, the "aih_person" relation was also built, as shown in part C. Its entries were taken from the other two index relations whenever a person name was present.

Finally, there is a very long file of italicized words/phrases, along with related text pointers. A user can then go back and forth between a phrase and the section it appears

Table 1. Approximate statistics on external knowledge bases

A. *AIList Document Collection*	
Number of messages	5,750
Number of authors	300
Number of digest issues	750
Dates covered	4/83–11/86
Characters of text	10 Mbytes
Concepts (in SMART)	
Word stems, proper names	25,000
Total	32,000
SMART collection sizes	
Document vectors	4 Mbytes
Inverted file	5 Mbytes
B. *Handbook of Artificial Intelligence*	
Number of files	106
Characters of text	4 Mbytes
Number of table of contents subjects	218
Number of index entries	853
Number of index range-entries	158
Number of index person names	138
Number of italicized words/phrases	5,009
C. *Collins Dictionary of the English Language*	
Number of relations produced	21
Number of headwords	85,000
Number of different parts of speech	46
Number of categories (used≥30 times)	120
Number of definitions	165,000
Number of morph. variants	28,000
Number of usage samples	17,000
Number of comparisons	8,000
Numbers for parts of speech	
Nouns	63,000
Verbs	15,000
Adjectives	13,000
Adverbs	1,300

Table 2. Volume/chapter structure of Handbook of Artificial Intelligence

Volume I by Avron Barr and Edward A. Feigenbaum
I. Introduction
II. Search
III. Knowledge Representation
IV. Understanding Spoken Language
Volume II by Avron Barr and Edward A. Feigenbaum
VI. Programming Languages for AI Research
VII. Applications-oriented AI Research: Science
VIII. Applications-oriented AI Research: Medicine
IX. Applications-oriented AI Research: Education
X. Automatic Programming
Volume III by Paul R. Cohen and Edward A. Feigenbaum
XI. Models of Cognition
XII. Automatic Deduction
XIII. Vision
XIV. Learning and Inductive Inference
XV. Planning and Problem Solving

Table 3. *Handbook of Artificial Intelligence* subject hierarchy

```
% Subject facts provide a number for each 'subject' in HAI.  A subject is an entry in
%   the table of contents outline; it includes chapter titles, subheadings, etc.  It is of
%   form:
%           subject(subject-number,subject-title).

% There are about 220 subjects, e.g.,
%           subject(0,'handbook of artificial intelligence').

% The hierarchical structure of the first set of subject-titles is shown below.

handbook of artificial intelligence
    introduction
        artificial intelligence
        the ai handbook
        the ai literature
    search
        overview 1
        problem representation
            state-space representation
            problem-reduction representation
            game trees
        search methods
            blind state-space search
            blind and/or graph search
            heuristic state-space search
                basic concepts in heuristic search
                a*--optimal search for an optimal solution
                relaxing the optimality requirement
                bidirectional search
            heuristic search of an and/or graph
            game tree search
                minimax procedure
                alpha-beta pruning
                heuristics in game tree search
        sample search programs
            logic theorist
            general problem solver 1
            gelernters geometry theorem-proving machine
            symbolic integration programs
            strips
            abstrips
```

in, and on up the hierarchy; alternatively, the user can begin with a section and find phrases that are important. All in all having the machine-readable form of the *HAI* allows users to browse in one text and in a substantial portion of the terminology used by the AI community.

Table 4. Relations derived from the *Handbook of Artificial Intelligence* back-of-the-book indexes

A. References to line of text in *HAI* for word/phrase in indexes

```
% index_ref([file_number,line_number],index_entry).
index_ref([4,33],'natural language').
index_ref([4,92],'logic').
index_ref([4,93],'computation').
index_ref([4,108],'turing, a.').
index_ref([4,125],'cybernetics').
index_ref([4,146],'computers').
index_ref([4,147],'computational complexity').
index_ref([4,211],'chess').
index_ref([4,212],'search').
index_ref([4,286],'problem solving').
index_ref([4,298],'problem representation').
```

B. References to range of lines of text in *HAI* for word/phrase in index

```
% index_rng([file_number_1,line_number_1],
%        [file_number_2,line_number_2],index_entry).
index_rng([4,4],[4,485],'intelligence').
index_rng([4,91],[4,233],'early ai').
index_rng([8,75],[8,444],'problem representation').
index_rng([8,155],[8,276],'forward reasoning').
index_rng([8,156],[8,277],'backward reasoning').
index_rng([8,318],[8,443],'search space').
index_rng([8,451],[8,582],'heuristics').
index_rng([8,521],[8,581],'heuristic search').
index_rng([8,522],[8,580],'blind search').
index_rng([8,537],[8,562],'generate-and-test').
```

C. Person names extracted from above index relations

```
% aih_person('person_name')
aih_person('abelson, robert').
aih_person('adelson-velskiy, g. m.').
aih_person('amarel, s.').
aih_person('anderson, j.').
aih_person('artsouni, g. b.').
aih_person('atkin, l. r.').
```

In addition to the *HAI* knowledge and text base, the CODER lexicon includes a set of twenty-one relations extracted [136] from the *Collins Dictionary of the English Language (CDEL)* [137]. This very large (over 80,000 headwords) dictionary (see statistics in part C of Table 1) provides a wealth of information to help with term expansion and natural language processing. Initial efforts have been made to supplement the *CDEL* portion of the lexicon with data from three other machine-readable dictionaries [138–140]. Additional information can be obtained from these dictionaries through more detailed analysis. A scheme for this, along with a more thorough description of the current lexicon, is given in a study by Fox et al. [141].

5.2. p-norm search expert

The p-norm query notation, which extends Boolean expressions to allow relative weights to be attached to terms and clauses and which allows "p-values" on the AND and OR operators to indicate the strictness of interpretation of the operation, is discussed in a work by Salton et al. [142]. While other schemes for "soft Boolean evaluation" have been proposed [143], none has been shown to perform as effectively as the p-norm method [144]. p-norm query processing has been incorporated in both the SMART and SIRE systems [145].

Because of its expressive power, the p-norm query form has been adopted in CODER as one of the canonical query forms. As can be seen in Fig. 8, a p-norm search expert has been developed that supports calls through the blackboard to attend to the "pnorm_query" area. The result of normal processing is to generate hypotheses for documents best satisfying the query expression, estimating the degree of relevance they have to the query.

A novice may encounter difficulty with p-norm query construction, so methods have been developed for automatic query formulation with or without feedback data [37]. Since

A) System Predicates Involved

ask:	used by any system module that wants to and is allowed to communicate with another, e.g., expert and external knowledge base, expert and blackboard/strategist.
attend_to_area:	sent by the strategist to an expert when it should examine the contents of a specific area of the blackboard.
facts_with_rel:	sent by an expert to an external knowledge base to obtain all facts in the designated relation that match the given pattern.
view_area:	sent by an expert to the (posting area manager of the) blackboard to obtain a list of the contents of a particular area.

B) Processing Sequence

1) Strategist wakes up the search-expert by using "ask" to send it
   attend_to_area(pnorm_query).
2) The search-expert begins its processing, and asks the blackboard for pending queries with the command
   view_area(pnorm_query, Hypothesis_set).
3) The search-expert receives the Hypothesis_set and extracts a query of form
   [relative_clause_wt, AND-or-OR, p-value, list_of_clauses].
4) The search-expert obtains document-no/wt pairs for each term. Presently this is done by directly accessing a MU Prolog inverted file. Later, for improved performance, an external knowledge base will have a C program as resource manager that can be asked for appropriate facts, using
   facts_with_rel(rel_id(Term, Doc_no_wt_list)).
5) The search-expert computes similarities for the desired number of documents and then posts the results to the doc-similarities area of the blackboard by asking, for each document, to
   post_hypothesis(doc-similarities, Result-hypothesis).

Fig. 8. Operation of the p-norm search expert.

these techniques require the identification of likely term dependencies, it would be of interest to relate the effectiveness of the resulting queries to that of probabilistic forms that also make use of those dependencies. Since Croft has demonstrated how dependency information in Boolean queries can be used to enhance the performance of probabilistic systems [146], a comparison with his findings might be of value.

5.3. Time/date handling

In order to support time references appearing in connection with a problem description, it is necessary to be able to parse queries and documents and to carry out temporal reasoning as required. Since a separate time reasoning project (by J. Roach et al.) is already under way at Virginia Tech, and may be adaptable for our purposes, our focus has been on identification, parsing, and representation of time-indicative expressions. The easiest to work with are the entries in digest and message headers.

In the "Date" field of electronic mail messages, a variety of notations for date and time can be found. Figure 9 shows forms for dates and times that our parser can process, and some samples of the fields found in AIList messages. As a result of using the date/time parser, each digest and each message can be related to a (small) absolute time interval attached to a time line.

Many different words and phrases, found in the body of documents and in queries, indicate that a time interval is being described [147,148]. A list of some common time words/phrases has been identified from the *CDEL* lexicon, and has been categorized by

A) Partial list of date patterns recognizable by current parser

1. Month Day, Year
2. Month Day, Year
3. Month / Day Year
4. Day - Month Year
5. Day - Month - Year
6. Day, Day_of_month
7. Year
8. Month Day Year
9. Month / Day / Year
10. Day_of_week Month / Day /Year

B) Partial list of time patterns recognizable by current parser

1. hour o'clock
2. hour o'clock a-pm
3. hour a-pm
4. hour : minute : second
5. hour : minute : second a-pm
6. hour a-pm zone

C) Sample of date-line facts from the AIList messages

Monday, April 25, 1983 5:27PM
Mon 25 Apr 83 14:51:42-PDT
Mon 25 Apr 83 09:34:04-PDT
22 Apr 1983 0227-EST
Thursday, 21-Apr-83 15:23:45-BST
Sun 24 Apr 83 20:41:46-PDT
Sunday, May 1, 1983 11:00AM
Thu 28 Apr 83 14:40:26-PDT

Fig. 9. Time and data forms in message headers.

part of speech. Since the duration of an interval is important for temporal reasoning, each time word/phrase has also been labeled as indicating: point of time (e.g., "sudden"), short duration (e.g., "momentary"), long duration (e.g., "durability"), endless duration (e.g., "eternal"), indefinite duration (e.g., "lapse"), or a specific duration (e.g., "semester"). Further, a past/present/future aspect classification also applies, where simultaneity is marked for words like "contemporary."

Given that a time word has been identified, it is common to find it as an element of a prepositional phrase. Parsing of such a phrase in connection with domain knowledge may allow text passages to be related to real world events and to support various kinds of reasoning with time intervals. As has been shown by Zarri [97], this type of inference may be of particular value in handling certain types of queries. A preliminary version of a time/date phrase expert is now operational.

5.4. User interaction and information gathering

At the end of Section 3.1, work on user modeling was briefly surveyed. The findings of Daniels, Brooks, and Belkin, [75] fit in nicely with the design of CODER and have informed our approach to user interaction. The long-term plan is to use the knowledge structures and rules they uncover in their study of user-intermediary dialogues, perhaps slightly adapted to our particular environment and collection, as a foundation for the user model data base and builder; the problem state and problem description builders; the report, browsing, and explanation experts; and the interface manager.

Fig. 10 provides an overview of the planned functionality of these parts of CODER. Part A, given in the form of a two-level menu, shows the main capabilities and their organization. Part B is a complementary view, illustrating the kinds of data that are recorded

A) Summary of Menu Choices Possible During a Session
1. ASSISTANCE: Would you like
   a- An explanation of the current situation
   b- Help regarding what you might do next
   c- A tutorial about some phase of the system's operations
2. COLLECTING: Can you provide more information about
   a- Your background
   b- The context or problem that prompted you to begin this session
   c- Your evaluation of the system's performance
3. QUERY: Can you
   a- Enter a description of your information need
   b- Revise the already existing description
4. BROWSING: Would you like to examine
   a- Facts from the *Handbook of Artificial Intelligence*
   b- Entries in the *Collins Dictionary of the English Language*
   c- Retrieved or other documents
   d- Information recorded about you in the User Model database
5. RESULTS: Do you need to
   a- Print some items
   b- Save some items in a file
6. EXIT

B) Examples of Data Recorded During a Session
1. USER BACKGROUND
   a- Reason for search
   b- Academic level
   c- Linguistic ability
   d- Experience or courses on computers, information retrieval systems
2. PROBLEM STATE
   a- Topic: general or specific
   b- Phase: continuing search, locating known object, searching, browsing
3. PROBLEM DESCRIPTION
   a- Topic: according to *HAI* contents (Table 3)
   b- Type: according to digest document type hierarchy (Fig. 7)
   c- Quantity: of items desired, indicating recall/precision needs
   d- Text: English prose describing document/passage desired
   e- Relevance: full documents or highlighted sections selected as useful

Fig. 10. User interaction and user information.

and used during a session. Data items listed in B.1 relate to fairly static and permanent entries, while those specified in parts B.2 and B.3 together specify the current state of the dialogue. The various system experts, some of which were shown in Fig. 4, are actually in charge of these activities—for example, the explanation expert handles tutorials, help, and other forms of explanation.

In the current implementation, background information as listed in part B.1 of Fig. 10 is gathered from users. Some initial work on the user model builder has taken place, and more is scheduled through the middle of 1987. At present, all data collected are logged. Problem state and description indicators are also requested, as shown in parts B.2 and B.3 of Fig. 10, and will later be handled by the appropriate builder experts (shown at the top of Fig. 4).

Finally, to gauge the user's feeling toward the system and its operation, evaluation questions are asked, as indicated in menu item A.2.c of Figure 10. With this feedback, the system could be tuned as a whole and to the needs of individual users, and should hopefully be shown to more effectively aid end user searching than would conventional approaches.

6. CONCLUSIONS

In light of new developments in computer and communications technology, there is a growing need for end users to search for bibliographic references, document passages, or other items with a textual component. In the case of composite documents, where structure and type interact and where particular components or passages are to be retrieved, it is necessary to carry out a more complete analysis of input text and to use additional information besides counts of word (stem) matches.

The CODER system has been under development since 1985 to serve as a testbed for the application of AI methods to IR problems. Although organized in a flexible fashion to handle a variety of retrieval tasks, its initial testing will be with messages in an archive of AIList Digest issues. It is hoped that CODER will perform well in this difficult domain, demonstrating the feasibility of analyzing and searching for passages in a composite document collection and of applying AI techniques.

While it may be true that certain types of knowledge-based processing do not contribute to retrieval performance [149], there is evidence that the approach taken in the CODER project will be of use. The blackboard-based construction allows modular development and enforces clean separation between fact bases, resource managers, blackboard communication, strategist control, and expert processing. An expert for p-norm searching functions well in conjunction with the MU-Prolog data base package, and some work has begun on time expression parsing and reasoning. Data is being gathered that will aid with user model building. The emphasis on using knowledge has led to a large lexicon constructed from machine-readable texts; the dictionary portion should be of general use and the domain-specific portion could easily be replaced by similar information taken from reference books in another domain. A skeletal system was operational, with different modules running on each of three computers, in January 1987. It is expected that by the fall of 1987 a fairly complete prototype will demonstrate the utility of the CODER design.

Acknowledgment—Robert K. France completed his MS thesis on the design of the CODER system, Robert C. Wohlwend completed his MS project working on building the *CDEL* Prolog fact base, and Sachit C. Apte completed an MS project relating to the communications and configuration issues. Qi Fan Chen and Marybeth T. Weaver have worked on numerous parts of CODER in connection with class projects and as graduate research assistants. Joy Weiss has provided secretarial assistance. Numerous other students have helped develop other parts of the system in connection with course and MS project efforts.

REFERENCES

1. Tenopir, C. Full-text databases. Annual Review of Information Science and Technology, 19: 215–246; 1984.
2. Ojala, M. Views on end-user searching. Journal of the American Society for Information Science, 37(4): 197–203; 1986.
3. Bates, M. J. Search techniques. Annual Review of Information Science and Technology. 16: 139–170; 1981.
4. Brenner, L. P.; Huston-Miyamoto, M.; Self, D. A.; Self, P. C.; Smith, L. C. User-computer interface designs for information systems: a review. Library Research, 2(1): 63–73; 1980–81.
5. Harter, S. P. Online information retrieval: concepts, principles and techniques. Orlando, FL: Academic Press; 1986.
6. Horowitz, G. L.; Jackson, J. D.; Bleich, H. L. PaperChase: self-service bibliographic retrieval. Journal of the American Medical Association, 250(18): 2494–2499; 1983.
7. Bates, M.J. Information search techniques. Journal of the American Society for Information Science, 30(4): 205–214; 1979.
8. Williams, P. W. How do we help the end user? Proceedings of the 6th national online meeting. Medford, NJ: Learned Information; 1985: 495–505.
9. Blair, D. C.; Maron, M. E. An evaluation of retrieval effectiveness for a full-text document-retrieval system. Communications of the ACM, 28(3): 289–299; 1985.
10. Katzer, J.; McGill, M. J.; Tessier, J. A.; Frakes, W.; Das Gupta, P. A study of the overlap among document representations. Information Technology: Research and Development, 1(4): 261–274; 1982.
11. Trivison, D.; Chamis, A. Y.; Saracevic, T.; Kantor, P. Effectiveness and efficiency of searchers in online searching: preliminary results from a study of information seeking and retrieving. Proceedings of the forty-ninth annual meeting of the American Society for Information Science; 1986 September 28–October 2; Chicago, IL. Medford, NJ: Learned Information; 1986: 341–349.
12. Salton, G. Another look at automatic text-retrieval systems. Communications of the ACM, 297: 648–656; 1986.
13. Fox, E.A. Information retrieval: research into new capabilities. In: Lambert, S.; Ropiequet, S., eds. CD-ROM: the new papyrus. Redmond, WA: Microsoft Press; 1986: 143–174.
14. Van Rijsbergen, C. J. A new theoretical framework for information retrieval. Proceedings of the 1986 annual

international ACM SIGIR conference on research and development in information retrieval; 1986 September; Pisa, Italy. New York: ACM; 1986: 194–200.
15. Biswas, G.; Subramanian, V.; Bezdek, J. C. A knowledge based system approach to document retrieval. Proceedings of the second conference on AI applications. Washington, DC: IEEE Press; 1985: 455–460.
16. Fox, E. A. Composite document extended retrieval: an overview. Proceedings of the eighth annual international ACM SIGIR conference on research and development in information retrieval; 1985 June; Montreal, Canada. New York: ACM; 1985: 42–53.
17. Pollitt, S.E. End user searching for cancer therapy literature—a rule based approach. ACM SIGIR Forum, 17(4): 136–145; 1983.
18. Clocksin, W. F.; Mellish, C. S. Programming in Prolog, 2nd ed. New York: Springer-Verlag; 1984.
19. Kowalski, R. A. Logic for problem solving. New York: Elsevier North-Holland; 1979.
20. Sterling, L.; Shapiro, E. The art of Prolog. Cambridge, MA: MIT Press; 1986.
21. Sciore, E.; Warren, D. S. Towards an integrated database-Prolog system. Kerschberg, L. ed. Expert Database Systems: proceeding from the first international workshop. Menlo Park, CA: Benjamin/Cummings 1986: 293–305.
22. Naish, L. MU-Prolog 3.2db reference manual. Department of Computer Science, University of Melbourne; July 1985.
23. Quarterman, J. S.; Hoskins, J. C. Notable computer networks. Communications of the ACM, 29(10): 932–971; 1986.
24. Myer, T. A. Standards for global messaging: a progress report. Journal of Telecommunication Networks, 2(4); 1983: 413–433.
25. Croft, W. B.; Pezarro, M. T. Text retrieval techniques for the automated office. In: N. Naffah, ed. Office information systems. Amsterdam: North-Holland; 1982: 565–576.
26. Gallelli, S.; Oacobelli, C.; Marchisio, P. An approach to multimedia information management. Proceedings of the 1986 ACM conference on research and development in information retrieval; 1986 September 8–10; Pisa, Italy. New York: ACM, 1986: 31–38.
27. Babatz, R.; Bogen, M. Semantic relations in message handling systems: referable documents. Proceedings of the IFIP Working Group 6.5 symposium; 1985 September; Washington, DC.
28. Fox, E. A. Expert retrieval for users of computer based message systems. Proceedings of the forty-ninth annual meeting of the American Society for Information Science, 1986 September 28–October 2; Chicago, IL. Medford, NJ: Learned Information; 1986: 88–95.
29. Niyogi, D.; Srihari, S. N. Document understanding using a knowledge-based methodology. Proceedings of the expert systems in government symposium; 1986 October 22–24; McLean, VA. Washington, DC: IEEE Computer Society Press; 1986: 128–134.
30. Kimura, G. D. A structure editor and model for abstract document objects. Ph.D. dissertation. Technical Report No. 84-07-04, Department of Computer Science, University of Washington; July 1984.
31. Kwok, K. L. The concept of document components for probabilistic indexing. Proceedings of the forty-ninth annual meeting of the American Society for Information Science; 1986 September 28–October 2; Chicago, IL. Medford, NJ: Learned Information; 1986: 158–162.
32. O'Connor, J. Answer-passage retrieval by text searching. Journal of the American Society for Information Science, 31(4): 227–239; 1980.
33. Salton, G. The SMART system 1961–1976: experiments in dynamic document processing. Encyclopedia of Library and Information Science, 28:1–36; 1980.
34. Fox, E. A. Some considerations for implementing the SMART information retrieval system under UNIX. Technical Report 83-560, Department of Computer Science, Cornell University, September 1983.
35. Fox, E. A.; France, R. K. Architecture of an expert system for composite document analysis, representation and retrieval. International Journal of Approximate Reasoning, 1(2); 1987, in press.
36. Bichteler, J.; Eaton, E. A. The combined use of bibliographic coupling and cocitation for document retrieval. Journal of the American Society for Information Science, 31(4): 278–282; 1980.
37. Fox, E. A. Extending the Boolean and vector space models of information retrieval with p-norm queries and multiple concept types. Ph.D. dissertation, Cornell University. Ann Arbor, MI: University Microfilms; August 1983.
38. Fox, E. A. Combining information in an extended automatic information retrieval system for agriculture. In: El-Hadidy: B.; Horne, E. E., eds. Infrastructure of an information society. Proceedings of the first international information conference; 1982 December; Cairo, Egypt. Amsterdam and New York: Elsevier North-Holland; 1984: 449–466.
39. Svenonius, E. Unanswered questions in the design of controlled vocabularies. Journal of the American Society for Information Science, 37(5): 331–340; 1986.
40. Amsler, R. A. Machine-readable dictionaries. Annual Review of Information Science and Technology, 19: 161–209; 1984.
41. Fox, E. A. Lexical relations: enhancing effectiveness of information retrieval systems. ACM SIGIR Forum, 15(3): 5–36; 1980.
42. Wang, Y. C.; Vandendorpe, J.; Evens, M. Relational thesauri in information retrieval. Journal of the American Society for Information Science, 36(1): 15–27; 1985.
43. Evens, M.; Vandendorpe, J.; Wang, Y. C. Lexical semantic relations in information retrieval. In: Williams, S., ed. Humans and machines: the interface through language. Norwood, NJ: Ablex; 1985: 73–100.
44. Fox, E. A. Improved retrieval using a relational thesaurus expansion of Boolean logic queries. Evens, M. W., ed. Proceedings of the workshop on relational models of the lexicon; 1984 July; Stanford, CA, in press.
45. Evens, M. W.; Smith, R. N. A lexicon for a computer question-answering system. American Journal of Computational Linguistics, Microfiche 83: 1–93; 1979.
46. Ahlswede, T.; Evans, M.; Rossi, K.; Markowitz, J. Building a lexical data base by parsing Webster's Seventh New Collegiate Dictionary. Proceedings of the second annual conference of the UW Centre for the New Oxford English Dictionary: Advances in Lexicology; 1986 November 9–11; Waterloo, Ontario, Canada. Waterloo, Ontario: University of Waterloo Centre for the New OED; 1986: 65–78.

47. Williams, M. E.; Kinnucan, M.; Smith, L. C.; Lannom, L.; Cho, D. Comparative analysis of online retrieval interfaces. Proceedings of the forty-ninth annual meeting of the American Society for Information Science; 1986 September 28–October 2; Chicago, IL. Medford, NJ: Learned Information; 1986: 365–370.
48. Kucera, H. Uses of on-line lexicons. Proceedings of the first conference of the UW Centre for the New Oxford English Dictionary: information in data; 1985 November 6–7; Waterloo, Ontario, Canada. Waterloo, Ontario: University of Waterloo Centre for the New OED; 1985: 7–10.
49. Walker, D. E. Knowledge resource tools for accessing large text files. Proceedings of the first conference of the UW Centre for the New Oxford English Dictionary: information in data; 1985 November 6–7; Waterloo, Ontario, Canada. Waterloo, Ontario: University of Waterloo Centre for the New OED; 1985: 11–24.
50. Salton, G. Recent trends in automatic information retrieval. Proceedings of the 1986 ACM SIGIR conference on research and development in information retrieval; 1986 September 8–10; Pisa, Italy. New York: ACM; 1986: 1–10.
51. Croft, W. B.; Thompson, R. H. The use of adaptive mechanisms for selection of search strategies in document retrieval systems. Proceedings of the third joint conference of the British Computer Society and the ACM symposium on research and development in information retrieval. Cambridge: Cambridge University Press; 1984: 95–110.
52. Robertson, S. E.; Sparck Jones, K. Relevance weighting of search terms. Journal of the American Society for Information Science, 27(3): 129–146; 1976.
53. Bookstein, A. Information retrieval: a sequential learning process. Journal of the American Society for Information Science, 34(5): 331–342; 1983.
54. Buckley, C.; Lewit, A. F. Optimization of inverted vector searches. Proceedings of the eighth annual ACM SIGIR international conference on research and development in information retrieval; 1985 June; Montreal, Canada. New York: ACM; 1985: 97–110.
55. Oddy, R. N. Information retrieval through man-machine dialogue. Journal of Documentation, 33(1): 1–14; 1977.
56. Frei, H. P.; Jauslin, J. F. Two-dimensional representation of information retrieval services. In: Dietschmann, H. J., ed. Representation and exchange of knowledge as a basis of information processes. New York: North Holland; 1984: 383–396.
57. Korfhage, R. R. Browser: a concept for visual navigation of a database. Technical Report No. 86-CSE-4, Department of Computer Science, Southern Methodist University, Dallas, TX; February 1986.
58. Crouch, D. The visual display of information in an information retrieval environment. Proceedings of he 1986 ACM conference on research and development in information retrieval; 1986 September 8–10; Pisa, Italy. New York: ACM; 58–67: 1986.
59. Miller, G. A. Wordnet: a dictionary browser. Proceedings of the first conference of the UW Centre for the New Oxford English Dictionary: information in data; 1985 November 6–7; Waterloo, Ontario, Canada. Waterloo, Canada: University of Waterloo Centre for the New OED; 1985: 25–28.
60. Sewell, W.; Teitelbaum, S. Observations of end-user online searching behavior over eleven years. Journal of the American Society for Information Science, 37(4): 234–245; 1986.
61. Rouse, W. B.; Rouse, S. H. Human information seeking and design of information systems. Information Processing & Management. 20(1-2): 129–138; 1984.
62. Markey, K. Levels of question formulation in negotiation of information need during the online presearch interview: a proposed model. Information Processing & Management, 17(5): 215–225; 1981.
63. Morrissey, J. M.; Harper, D. J.; van Rijsbergen, C. J. Interactive querying techniques for an office filing facility. Information Processing & Management, 22: 121–134; 1986.
64. Ingwersen, P. Search procedures in the library—analysed from the cognitive point of view. Journal of Documentation, 38(3): 165–191; 1982.
65. Ingwersen, P.; Wormell, I. Improved subject access, browsing and scanning mechanisms in modern on-line IR. Proceedings of the 1986 ACM SIGIR conference on research and development in information retrieval; 1986 September 8–10; Pisa, Italy. New York: ACM; 1986: 68–76.
66. Borgman, C. L. Psychological research in human-computer interaction. Annual Review of Information Science and Technology, 19: 33–64; 1984.
67. Borgman, C. L. Designing an information retrieval interface based on user characteristics. Proceedings of the eighth annual ACM SIGIR international conference on research and development in information retrieval; 1985 June; Montreal, Canada. New York: ACM, 1985: 139–146.
68. Borgman, C. L.; Case, D.; Meadow, C. T. Evaluation of a system to provide online instruction and assistance in the use of energy databases: the DOE/OAK project. Proceedings of the forty-ninth annual meeting of the American Society for Information Science; 1986 September 28–October 2; Chicago, IL. Medford, NJ: Learned Information; 1986: 32–38.
69. Fenichel, C. H. The process of searching online bibliographic databases: a review of research. Library Research, 2(2): 107–127; 1980–81.
70. Fenichel, C. H. Online searching: measures that discriminate among users with different types of experience. Journal of the American Society for Information Science, 32(1): 23–32; 1981.
71. Fidel, R. Toward expert systems for the selection of search keys. Journal of the American Society for Information Science, 37(1): 37–44; 1986.
72. Bates, M. J. Subject access in online catalogs: a design model. Journal of the American Society for Information Science, 37(6): 357–376; 1986.
73. Daniels, P. J. Cognitive models in information retrieval—an evaluative review. Journal of Documentation, 42(4): 272–304; 1986.
74. Daniels, P. J. The user modelling function of an intelligent interface for document retrieval systems. Proceedings of sixth international research forum in information science (IRFIS 6): intelligent information systems for the information society; 1985 September 16–18; Frascati, Italy. Brookes, B. C., ed. Amsterdam: North Holland; 1986: 162–176.
75. Daniels, P. J.; Brooks, H. M.; Belkin, N. J. Using problem structures for driving human-computer dialogues. Proceedings Recherche d'informations assistee par ordinatur (RIAO '85). Grenoble: IMAG; 1985: 645–660.

76. Brooks, H. M.; Daniels, P. J.; Belkin, N. J. Problem descriptions and user models: developing an intelligent interface for document retrieval systems. Advances in intelligent retrieval. Proceedings of Informatics 8. London: Aslib; 1985: 191–214.
77. Belkin, N.; Seeger, T.; Wersig, G. Distributed expert problem treatment as a model for information system analysis and design. Journal of Information Science, 5: 153–167; 1983.
78. Smith, L. C. Artificial intelligence applications in information systems. Annual Review of Information Science and Technology, 15: 67–106; 1980.
79. Sparck Jones, K.; Kay, M. Linguistics and information science. New York: Academic Press; 1973.
80. Doszkocs, T. E. Natural language processing in information retrieval. Journal of American Society of Information Science, 37(4); 191–196; 1986.
81. Korfhage, R. R.; Hemphill, C. Retrieval linguistics. Technical Report No. 84-CSE-12, Department of Computer Science, Southern Methodist University, Dallas, TX; 1984.
82. Pereira, F. Logic for natural language analysis. Technical Note 275, SRI International, Menlo Park, CA; Jan. 1983.
83. Sommerville, I.; Blair, A. DSA—a tool for descriptive text analysis. Research Report CS/ST/2/85, Software Technology Research Group, University of Strathclyde; Glasgow, Scotland; 1985.
84. Sager, N. Natural language information processing. New York: Addison-Wesley; 1981.
85. Katzer, J.; Bonzi, S.; Liddy, E. The effects of anaphoric resolution on retrieval performance: preliminary findings. Proceedings of the forty-ninth annual meeting of the American Society for Information Science; 1986 September 28–October 2; Chicago, IL. Medford, NJ: Learned Information; 1986: 118–122.
86. Sparck Jones, K.; Tait, J. I. Automatic search term variant generation. Journal of Documentation, 40(1): 50–66; 1984.
87. Smeaton, A. F. Incorporating syntactic information into a document retrieval strategy: an investigation. Proceedings of the 1986 ACM SIGIR conference on research and development in information retrieval; 1986 September 8–10; Pisa, Italy. New York: ACM; 1986: 103–113.
88. Smeaton, A. F.; van Rijsbergen, C. J. Information retrieval in an office filing facility and future work in Project Minstrel. Information Processing & Management, 22: 135–149; 1986.
89. Fikes, R.; Kehler, T. The role of frame-based representation in reasoning. Communications of the ACM, 28(9): 904–920; 1985.
90. Fox, M. S.; Wright, J. M.; Adam, D. Experiences with SRL: an analysis of a frame-based knowledge representation. In: Kershberg, L., ed. Expert database systems: proceedings from the first international workshop. Menlo Park, CA: Benjamin/Cummings; 1986: 161–172.
91. Humphrey, S. M.; Miller, N. E. The NLM indexing aid project. Proceedings of the forty-ninth annual meeting of the American Society for Information Science; 1986 September 28–October 2; Chicago, IL. Medford, NJ: Learned Information; 1984: 106–112.
92. Hayes-Roth, F. Rule-based systems. Communications of the ACM, 28(9): 921–932; 1985.
93. Simmons, R. F. A text knowledge base for the AI handbook. Technical Report TR-83-24, University of Texas at Austin, Austin, TX; December 1983.
94. Skuce, D.; Matwin, S.; Tauzovich, B.; Szpakowicz, S.; Oppacher, F. A rule-oriented methodology for constructing a knowledge base from natural language documents. Proceedings of the expert systems in government symposium. Washington, DC: IEEE Computer Society Press; 1985: 378–385.
95. Allen, J. F. Maintaining knowledge about temporal intervals. Communications of the ACM, 26(11): 832–843; 1983.
96. Mays, E. A temporal logic for reasoning about changing data bases in the context of natural language question-answering. In: Kershberg, L., ed. Expert database systems: proceeding from the first international workshop. Menlo Park, CA: Benjamin/Cummings; 1986: 559–578.
97. Zarri, G. P. An outline of the representation and use of temporal data in the RESEDA system. Information Technology: Research Development Applications, 2(2/3): 89–108; 1983.
98. Zarri, G. P. Expert systems and information retrieval: an experiment in the domain of biographical data management. International Journal of Man-Machine Studies, 20(1): 87–100; 1984.
99. Chandrasekaran, B. Generic tasks in knowledge-based reasoning: high-level building blocks for expert system design. IEEE Expert, 1(3): 23–30; 1986.
100. Bylander, T.; Mittal, S. CRSL: a language for classificatory problem solving and uncertainty handling. AI Magazine, 7(3): 66–77; 1986.
101. Helm, A. R.; Marriott, K.; Lassez, C. Prolog for expert systems: an evaluation. Proceedings of the expert systems in government symposium. Washington, DC: IEEE Computer Society Press; 1985: 284–293.
102. Lee, N. S. Fuzzy inference engines in Prolog/P-shell. Proceedings of the IEEE expert systems in government conference; 1986 October 20–24; McLean, VA. Washington, DC: IEEE Computer Society Press; 1986: 243–247.
103. Lecot, K. Logic programs with uncertainties: dealing with multiple evidence. Proceedings IEEE expert systems in government conference; 1986 October 20–24; McLean, VA. Washington, DC: IEEE Computer Society Press; 1986: 234–242.
104. Cheeseman, P. In defense of probability. Proceedings of the fourth national conference on artificial intelligence (AAAI-85). Menlo Park, CA: Morgan Kaufman; 1985: 1002–1009.
105. Erman, L. D.; Hayes-Roth, F.; Lesser, V. R.; Reddy, D. R. The Hearsay-II speech-understanding system: integrating knowledge to resolve uncertainty. ACM Computing Surveys, 12: 213–253; 1980.
106. Nii, H. P. Blackboard systems: blackboard application systems, blackboard systems from a knowledge engineering perspective. AI Magazine, 7(3): 82–106; 1986.
107. Nii, H. P. Blackboard systems: the blackboard model of problem solving and the evolution of blackboard architectures. AI Magazine, 7(2): 38–53; 1986.
108. DeJong, G. An overview of the FRUMP system. In: Lehnert, W. G.; Ringle, M. H., eds. Strategies for natural language processing. Hillsdale, NJ: Lawrence Erlbaum; 1982: 149–176.
109. Mauldin, M. L. Thesis proposal: information retrieval by text skimming. Unpublished manuscript, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA; May 22, 1986.
110. Hahn, U.; Reimer, U. Heuristic text parsing in 'Topic': methodological issues in a knowledge-based text

condensation system. In: Dietschmann, H. J., ed. Representation and exchange of knowledge as a basis of information processes. New York: North Holland; 1984: 143–163.
111. Bosc, P.; Courant, M.; Robin, S. CALIN: a user interface based on a simple natural language. Proceeding of the 1986 ACM SIGIR conference on research and development in information retrieval; 1986 September 8–10; Pisa, Italy. New York: ACM; 1986: 114–122.
112. Chiaramella, Y.; Defude, B.; Bruandet, M. F.; Kerkouba, D. IOTA: a full text information retrieval system. Proceeding of the 1986 ACM SIGIR conference on research and development in information retrieval; 1986 September 8–10; Pisa, Italy. New York: ACM; 1986: 207–213.
113. Patel-Schneider, P. F.; Brachman, R. J.; Levesque, H. J. ARGON: knowledge representation meets information retrieval. Proceedings of the first conference on artificial intelligence applications; 1984 December; Denver, CO. Washington, DC: IEEE Computer Society Press; 1984: 280–286.
114. Di Benigno, G.; Cross, R.; deBessonet, C. G. COREL: a conceptual retrieval system. Proceedings of the 1986 ACM SIGIR conference on research and development in information retrieval; 1986 September 8–10; Pisa, Italy. New York: ACM; 1986: 144–148.
115. Watters, C. R.; Shepherd, M. A. A logic basis for information retrieval. Technical Report; Department of Mathematics, Statistics, and Computing Science, Dalhousie University, Halifax, Nova Scotia, Canada; 1986.
116. McCune, B. P.; Tong, R. M.; Dean, J. S.; Shapiro, D. G. RUBRIC: a system for rule-based information retrieval. IEEE Transactions on Software Engineering, SE-11(9): 939–945; 1985.
117. Tong, R. M.; Appelbaum, L. A.; Askman, V. N.; Cunningham, J. F. RUBRIC III: an object-oriented expert system for information retrieval. Proceedings of the IEEE expert systems in government conference; 1986 October 20–24; McLean, VA. Washington, DC: IEEE Computer Society Press; 1986: 106–115.
118. Doszkocs, T. E. From research to application: the cite natural language information retrieval system. In: Salton, G.; Schneider, H.-J., eds. Research and development in information retrieval: Proceedings, Berlin, May 18–20, 1982. Berlin; New York: Springer-Verlag; 1983: 38–50.
119. Biswas, G.; Bezdek, J. C.; Marques, M.; Subramanian, V. Knowledge-assisted document retrieval: I. the natural-language interface. Journal of the American Society for Information Science, 38(2): 83–96; 1987.
120. Biswas, G.; Bezdek, J. C.; Viswanath, S.; Marques, M. Knowledge-assisted document retrieval: II. the retrieval process. Journal of the American Society for Information Science, 38(2): 97–110; 1987.
121. Yip, M. An expert system for document retrieval. M.S. thesis, MIT, Cambridge, MA; 1979.
122. Marcus, R. S. Design questions in the development of expert systems for retrieval assistance. Proceedings of the forty-ninth annual meeting of the American Society for Information Science; 1986 September 28–October 2; Chicago, IL. Medford, NJ: Learned Information; 1986: 185–189.
123. Guida, G.; Tasso, C. An expert intermediary system for interactive document retrieval. Automatica, 19: 759–766; 1983.
124. Pollitt, S. E. A 'front-end' system: an expert system as an online search intermediary. Aslib Proceedings, 36(5): 229–234; 1984.
125. Defude, B. Different levels of expertise for an expert system in information retrieval. Eighth annual international ACM SIGIR conference on research and development in information retrieval; 1986 June; Montreal, Canada. New York: ACM; 1985: 147–153.
126. Mukhopadhyay, U.; Stephens, L. M.; Hughs, M. N.; Bonnell, R. D. An intelligent system for document retrieval in distributed office environments. Journal of the American Society for Information Science, 37(3): 123–135; 1986.
127. Thompson, R. H.; Croft, W. B. An expert system for document retrieval. Proceedings of the expert systems in government symposium. Washington, DC: IEEE Computer Society Press; 1985: 448–456.
128. Croft, W. B. User-specified domain knowledge for document retrieval. Proceedings of the 1986 ACM conference on research and development in information retrieval; 1986 September 8–10; Pisa, Italy. New York: ACM; 1986: 201–206.
129. Fox, E. A. A design for intelligent retrieval: the CODER system. Second conference on computer interfaces and intermediaries for information retrieval; 1986 May 28–31; Boston, MA. Alexandria, VA: Defense Technical Information Center; 1986: 135–153.
130. Belkin, N. J.; Hennings, R. D.; Seeger, T. Simulation of a distributed expert-based information provision mechanism. Information Technology: Research Development Applications, 3(3): 122–141; 1984.
131. France, R. K. An artificial intelligence environment for information retrieval research. MS thesis, Computer Science Department, Virginia Polytechnic Institute & State University, Blacksburg, VA; July 1986.
132. Levesque, H. J. A fundamental tradeoff in knowledge representation and reasoning (revised version). In: Brachman, R. J.; Levesque, H. J., eds. Readings in knowledge representation; Los Altos, CA: Morgan Kaufmann; 1985; 41–70.
133. France, R. K.; Fox, E. A. Knowledge structures for information retrieval: representation in the CODER project. Proceedings of the IEEE expert systems in government conference; 1986 October 20–24; McLean, VA. Washington, DC: IEEE Computer Society Press; 1986: 135–141.
134. Ramamohanaro, K.; Shepherd, J. A superimposed codeword indexing scheme for very large Prolog databases. Technical Report 85/17, Computer Science Department, University of Melbourne, Melbourne, Australia, 1985.
135. Sacks-Davis, R. Performance of a multi-key access method based on descriptors and superimposed coding techniques. Information Systems, 10: 391–403; 1985.
136. Wohlwend, R. C. Creation of a Prolog fact base from the Collins English dictionary. MS report, Computer Science Department, Virginia Polytechnic Institute of State University, Blacksburg, VA; March 1986.
137. Hanks, P., ed. Collins dictionary of the English language. London: William Collins Sons; 1979.
138. Cowie, A. P.; Mackin, R. Oxford dictionary of current idiomatic English. Volume 1: verbs with prepositions & particles. Oxford: Oxford University Press; 1975.
139. Cowie, A. P.; Mackin, R.; McCaig, I. R. Oxford dictionary of current idiomatic English. Volume 2: phrase, clause & sentence idioms. Oxford: Oxford University Press; 1983.
140. Hornby, A. S., ed. Oxford advanced dictionary of current English. Oxford: Oxford University Press; 1974.
141. Fox, E. A.; Wohlwend, R. C.; Sheldon, P. R.; Chen, Q.; France, R. K. Building the CODER lexicon: the

Collins English dictionary and its adverb definitions. TR-86-23, Computer Science Department, Virginia Polytechnic Institute & State University, Blacksburg, VA; October 1986.
142. Salton, G.; Fox, E. A.; Wu, H. Extended Boolean information retrieval. Communications of the ACM, 26(11): 1022–1036; 1983.
143. Paice, C. D. Soft evaluation of Boolean search queries in information retrieval. Information Technology: Research Development Applications, 3(1): 33–42; 1984.
144. Fox, E. A.; Sharat, S. A comparison of two methods for soft Boolean operator interpretation in information retrieval. TR-86-1, Computer Science Department, Virginia Polytechnic Institute & State University, Blacksburg, VA; Jan. 1986.
145. Fox, E. A.; Koll, M. B. Practical enhanced Boolean retrieval: experiences with the SMART and SIRE systems. Information Processing & Management; accepted for publication.
146. Croft, W. B. Boolean queries and term dependencies in probabilistic retrieval models. Journal of the American Society for Information Science, 37(2): 71–77; 1986.
147. Bennett, D. C. Spatial and temporal uses of English prepositions: an essay in stratificational semantics. London: Longman; 1975.
148. Funk & Wagnalls Editorial Staff. Standard handbook of prepositions, conjunctions, relative pronouns and adverbs. New York: Funk & Wagnalls; 1953.
149. Salton, G. On the use of knowledge-based processing in automatic text retrieval. Proceedings of the forty-ninth annual meeting of the American Society for Information Science; 1986 September 28–October 2; Chicago, IL. Medford, NJ: Learned Information; 1986: 277–287.