The Design and Implementation of an Intelligent Interface for Information Retrieval

Roger Howard Thompson
Ph.D. Thesis
Computer and Information Science Department
University of Massachusetts

COINS Technical Report 88-88

The Design and Implementation of an Intelligent Interface for Information Retrieval

A Dissertation Presented

by

ROGER HOWARD THOMPSON

Submitted to the Graduate School of the

University of Massachusetts in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

February 1989

Department of Computer and Information Science

The Design and Implementation of an Intelligent Interface for Information Retrieval

A Dissertation Presented

by

Roger Howard Thompson

Approved as to style and content by:

W. Bruce Croft, Chairperson of Committee

David W. Stemple, Member

Nicholas J. Belkin, Member

Joseph Sardinas, Outside Member

W. Richards Adrion, Department Head
Department of Computer and Information Science

DEDICATION

This work is dedicated to the memory of

Dr. Victor Paul Wierwille.

ACKNOWLEDGMENTS

I would like to thank the following people, who helped me greatly to accomplish this work. First, I would like to thank my advisor Bruce Croft, whose constant encouragement and thoughtful constructive criticism was instrumental in helping see this project through. Professors Dave Stemple and Nick Belkin provided me with different perspectives that enabled me to think more clearly.about the subject.

I would like to thank some of the residents of the Wombat Research Lab, Larry Lefkowitz, Carol Broverman, Tom Parenty, Norm Carver, and Al Hough for making the time bearable.

I would like to thank my supervisors at Hughes Aircraft, Bill King and Jim Blackburn for their understanding while finishing my writing.

I would like to thank my friend, Andy Zitelli, for his wise counsel throughout my entire undergraduate and graduate academic career.

Finally, I would like to express my gratitude to wife, Darlene, for her unfailing encouragement and support, and my daughter Rebeca for the joy that only a young child can bring.

ABSTRACT

THE DESIGN AND IMPLEMENTATION OF AN INTELLIGENT INTERFACE FOR INFORMATION RETRIEVAL

February, 1989

ROGER HOWARD THOMPSON, B.A., UNIVERSITY OF CALIFORNIA AT BERKELEY

M.S., NEW MEXICO STATE UNIVERSITY

Ph.D., UNIVERSITY OF MASSACHUSETTS

Directed by: Professor W. Bruce Croft

Commercial information (text) retrieval systems have been available since the early 1960's. While they have provided a service allowing individuals to find useful documents out of the millions of documents contained in online databases, their are, a number of problems that prevent the user from being more effective. The primary problems are an inadequate means for specifying information needs, a single way of responding to all users and their information needs, and an inadequate user interface.

This thesis describes the design and implementation of I^3R, an intelligent interface for information retrieval the purpose of which is to overcome the limitations of current information retrieval systems by providing multiple ways of assisting the user to precisely specify his information need and to search for information. The system organization is based on a blackboard architecture and consists of a number of "experts" that work cooperatively to assist the user. The operation of the experts is coordinated by a control expert that makes its decisions based on a plan derived from the analysis of human search intermediaries, end user dialogues, and user model. The experts provide multiple formal search strategies, the use and collection of domain knowledge, and browsing assistance. The operation of the system is demonstrated by four scenarios.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

OVERVIEW

1.1 Introduction

In this chapter, an overview of this dissertation is presented. We begin by discussing the problems of traditional information retrieval systems and how they are usually overcome. These problems form the basis for the requirements of a more sophisticated system called I^3R, an Intelligent Interface for Information Retrieval. A design is then outlined that will meet the specified requirements. The design has two major aspects: the first is facilities that should be provided; the second is how these facilities are to be supported in ways that allow easy modification.

1.2 Retrieval Problems

Commercial retrieval systems have been available since the early 1960's. At that time, they were a significant breakthrough in the use of computers for non-numeric applications. They allowed scientists and engineers to sort through the many journals, technical reports, and other written works to find information that might be useful in helping them solve their problems. The utility of these systems has been recognized in other professions such as law and medicine, where major retrieval services are now available.

While developments in storage technology, such as ever increasing densities in disk storage, and developments in communications technology, such as relatively inexpensive 2400 baud modems, have made these systems more widely available, the interface technology has remained for the most part stagnant, reflecting the designs of the original systems. These interfaces were designed to operate with simple input/output devices such as 110 character/second printing terminals. This significantly limits the kind of information

that can be displayed. Furthermore, the operation of the system has a command-line orientation, which is reflected in the use of specialized languages for query specification.

These languages are based on Boolean logic and are usually augmented with proximity operators and "don't care" or wildcard characters. The former specify how close words must be in sentences or paragraphs. The latter handle alternative spellings and inflected forms of words. The use of these languages requires specialized training for the user to teach them the semantics for AND, OR, and NOT. While the basic concepts are relatively simple, use of these languages is mastered only after a significant amount of experience. Furthermore, different systems have different query languages and many users do not have the time or the inclination to learn Boolean logic.

Boolean logic cannot precisely specify many relationships between words. For example, AND can be used to describe phrases or words that are required; OR may specify alternative words, synonyms, or components of "higher" level concepts. In addition, AND and OR in some situations in everyday language can be used synonymously. This lack of precision or multiple meaning can be overcome by adding other operators to specify relationships more exactly or adding weights to the AND and OR to give "soft" Boolean operators [Salton 83].

Both solutions, while feasible, simply add to the amount of knowledge that the user must know in order to use a system effectively. This increases the potential for confusion and, hence, frustration on the part of the end user. The casual user or "permanent novice" will, in all likelihood, never bother to learn how to use the advanced features of the query language.

Compounding the problem of using the query language, which is a matter of query form, is the problem of determining precisely what is the content of the query. This is a problem of selecting the proper words to express what the user wants. Two potential problems arise here. The first is that the user may not know exactly what he wants, and the

second is that he may not know the precise terminology required to express the need. In some systems, the user has recourse to an online thesaurus, which is a collection of words that is structured to show the relationships between them, to find the proper descriptive terms and to give the structure of the knowledge of a domain. In others, the best that he can do is get an alphabetical list of terms occurring in the database.

The problems of query form and content are manifestations of the inflexible nature of retrieval systems. They have only one way to respond to every type of user and every type of problem.

To overcome this inflexibility, end-users, the persons with the information need, often resort to using the services of a search intermediary. Intermediaries have received specialized training in the use of retrieval systems. They often have a degree in librarianship or have a degree in the field in which they search or by constant use have developed a knowledge of the terminology of a domain. For example, an intermediary that searches Chemical Abstracts might have a Ph. D. in chemistry. This background allows them to concentrate on getting the best possible results from the retrieval system by knowing the correct terminology.

One of the main advantages of using intermediary services is that the intermediary, being a person, can be much more flexible than the current commercial systems. The intermediary can adapt to the needs of different users. If the session is the end-user's first experience, the intermediary can help the user understand the search process by explaining what he is doing as he goes along. The intermediary can adjust his explanations to match the kind of user that he is dealing with. A college freshman with an orientation to the humanities would require a different kind of assistance than a medical doctor with some computing experience. Another advantage of an intermediary is that he can continue to learn about the domains that people consistently search in and he can learn about the needs of the people that consistently use their services.

While the use of intermediary services removes the burden from the end-user of having to deal with the query language, and often provides him with terminological assistance, it adds a new difficulty, since the user is now often removed from participating directly in the search process. The user must now, as before, try to express his information need to the intermediary, but, in general, cannot take advantage of the recognition ability that humans have in the search process. This is due to the fact that often intermediaries will search without the user present. The preferred situation is to have the end-user present with the search intermediary while the search is taking place. This, however, often slows the intermediary down, since he often has to explain his actions to the end user. This situation is not always possible due to considerations such as scheduling, among others. Other factors such as the availability of intermediary services also come into play. These services may not be free; adding further to the cost of using the system. Furthermore, with the advent of extremely high density storage such as CD-ROM (Compact Disk–Read Only Memory), end-users may be searching for information in their own home, where search intermediaries are not available.

1.3 The Intermediary Model

The search intermediary provides a model that can be useful in designing systems that can help overcome the problems of using IR systems. There are two ways that this concept can be used. One way is to simulate the activity of an intermediary, that is to attempt to provide the same services as the intermediary. This has been the basis of a number of expert systems that provide such services as a common command language to multiple retrieval systems [Marcus 81a, Marcus 81b, Marcus 83] and rudimentary query formulation assistance [Yip 79, Pollitt 84]. More sophisticated systems [Brajnik 85, Brajnik 87, Chiaramella 87, Defude 85] that take this approach attempt to implement the strategies and tactics used by intermediaries for searching [Bates 79a, 79b] and attempt to

incorporate a natural language dialogue with the user. All of the systems that attempt this kind of simulation have been designed to work with Boolean systems and therefore have the limitations on retrieval effectiveness [Salton 83] that plague Boolean systems.

The approach taken in this thesis is to look at the intermediary concept as an intelligent interface system which is composed of the intermediary and the retrieval system. Analysis can then be made of the kinds of facilities that this system provides or should provide to assist the user in expressing his need and finding information that will meet it. The system designer can then determine how best to implement those facilities, taking advantage of the current research in information retrieval, and not be limited to ineffective, immature, or inappropriately applied technologies in an effort to exactly simulate the human intermediary.

1.4 System Analysis and Requirements

In analyzing the combined intermediary/retrieval system, the four basic elements of a retrieval system are the basis of the analysis. These basic elements are:

1. a representation of the content or meaning of the documents and the queries,
2. a process, usually called indexing, that maps the content of the document and the queries into the content representation,
3. a decision method, usually called a search strategy, that the system uses to determine whether or not a document should be retrieved,
4. a user interface.

The user interface element in the combined intermediary/retrieval system is composed of the services that the search intermediary provides, and the actual method (i.e. how the query is typed in, how results are displayed, etc.) of interacting with the system. The essential services that the intermediary provides are:

1. explanation of system operation,
2. term selection assistance,

3. construction of a model of the information need, which consists of the query and the documents that have been retrieved,

4. execution of the searches,

5. overall control of the course of a session.

To adapt to the different kinds of end-users, the intermediary must make some assessment with regard to the end-user's familiarity with the domain, his familiarity with the search process, and the kind of results that he wants, such as whether he wants a few specific documents or a comprehensive collection. Essentially, the intermediary forms a model of the end-user and adapts the session to that model.

While the intermediary aspect of the system addresses most of the issues of inflexibility, some of them are rooted in the retrieval portion of the system. In the past, systems have been limited to a single decision method (retrieval method) for determining what documents ought to be retrieved. By having different methods for different kinds of queries the effectiveness of a system can be increased substantially [Croft 85]. A system's effectiveness can also be increased by providing direct access to the documents by browsing, a heuristically driven incremental search and evaluation technique [Oddy 77]. Browsing need not be limited to just the examination of documents; it can also be used to find the appropriate concepts to describe the information need.

The preceding high level analysis of the elements of the combined intermediary/retrieval system has pointed out the need for the system to support a number of facilities or functions that either provide services similar to that of an intermediary or support functions that are part of the underlying retrieval system. These functions or services can be summarized in the following modules:

1. Explainer – explains system operation to the user,

2. Domain Knowledge Expert – suggests additional concepts to the user and acquires domain knowledge from the user,

3. Request Model Builder – maintains information about the current state of the session such as relevant concepts and relevant documents,

4. Search Controller – chooses search techniques that are appropriate to the current state of the session and information need,

5. User Model Builder – determines what kind of end-user is currently interacting with the system,

6. Browsing Expert – provides recommendations to the user about information to view that is likely to be relevant when the user is browsing, and remember the path that the user has taken during browsing,

7. Control Module – determines the direction of the "dialogue" that system has with the user.

The representation for the documents must contain all the information necessary to support multiple search strategies and browsing. Traditional systems have usually maintained simple inverted files that would be inadequate in this case. In addition, thesaurus information in most systems has not been integrated into the overall retrieval process.

A number of other factors come into play in determining what the requirements of the retrieval system should be. One important factor of traditional information retrieval systems that is desirable to maintain is their domain independence. This means that the system cannot depend on having a significant amount of domain specific knowledge. However, since domain knowledge is very useful in assisting the user to precisely express his information need, the system should have the ability to use whatever domain specific knowledge that is available, and should be able to acquire this knowledge from the user.

1.5 Architecture

In order to build a system that provides the kinds of facilities that the combined search intermediary/retrieval system does, it must have an architecture that allows it to be flexible. This flexibility is manifested in a number of different ways. First, the system must adapt itself to different kinds of users and different kinds of information needs; this is external flexibility. Second, it must be flexible enough so that it can incorporate new techniques as they are developed; this is internal flexibility.

The first kind of flexibility requires that the system changes the way it interacts with the user as does the intermediary. For a novice user, it should offer more explanation and assistance, and it should limit his choices so that he does not get in to a situation that he cannot handle; for an expert user it should not interfere with his use of the system, and should provide him access to all of the system's functionality. Another aspect of this flexibility is that different kinds of information needs require different kinds of searches. The system must be able to respond appropriately.

The second kind of flexibility requires an architecture that is modular in nature. This modularity should be at two levels. It should be able to support the addition of new large pieces of functionality. This would allow it to take advantage of new developments in information retrieval research. Each large scale function should also be modular, so that it can be adjusted to operate more effectively as the pattern of system usage is established. It also allows for the integration of new developments. For example, if a new search technique is developed that is particularly good at retrieving relevant information for one kind of information need, it can be incorporated into the search function of the system.

The architecture that best supports the requirements of an intelligent IR interface is a modified blackboard architecture [Erman 80, Nii 86a Nii, 86b]. A blackboard architecture, of which Hearsay II is a typical example, consists of a number of independently operating modules, called knowledge sources, that work together to solve a problem. Each works on a particular aspect of a problem. The results of their work is posted on a shared data structure called a blackboard. This blackboard is typically organized as a series of levels that represent abstraction levels of the problem. The operation of the knowledge sources is coordinated by a scheduler.

The basic operation of a blackboard system is as follows. First, each expert examines the state of the blackboard in its area of interest. It then decides if it has any action that it would like to perform based on the current conditions. If it does, it places an action

(called an instantiation) on the system agenda. The agenda is examined by the scheduler and is sorted in order of importance based on criteria that are problem dependent. The scheduler then takes the most important action and runs it. The cycle then begins again.

The blackboard architecture is appropriate since supports the easy addition of large scale functions by means of knowledge sources. In addition, the way that knowledge sources are to be implemented is not specified, so they can be implemented in the way that is most appropriate for their specific task. The knowledge sources in I^3R are called experts since they are implemented as individual rule based systems. This provides a means of incrementally developing the experts. These experts correspond to the functions that were derived in the system analysis.

The basic blackboard architecture must be adapted to fit the nature of the information retrieval problem. The first adaptation is to the structure of the blackboard; it is not structured into abstraction levels, since there is no single overall hierarchical representation that can be applied to IR. Instead, the blackboard, called the short term memory, consists of different models built by the experts in the course of the session.

The purpose of the control function in I^3R also differs from that of the scheduler in a typical blackboard system. In a typical system, the scheduler manages the system's resources to come to the solution of the problem in the shortest time possible. In I^3R the control expert manages the dialogue the system has with the user, so that it is consistent and coherent. This difference stems from the fact that information retrieval is better likened to a process than to a problem to be solved. The control expert makes sure that the process is conducted correctly.

The control function uses information provided by the user model builder and the request model to determine the course of a session. The information for the user model builder is based on the stereotypes that the UMB decides apply to the particular user for the particular session. Stereotypes are models of different kinds of typical users. In the

current system three general categories are used, with two values for each category. The categories are domain expertise, search system expertise, and search type. The values are novice and expert for the first two categories, and selective or exhaustive for the third category.

The documents, concepts, and user histories are kept in a long term memory. The user histories store information about the user obtained from previous sessions with the system. This includes the original query, concepts that were judged relevant, documents that were judged relevant, and the stereotypes that were in effect at the end of the session. Also included in the user histories is a model of the whatever domain knowledge that the user has contributed in the course of his interaction with the system.

The system also maintains a store of global domain knowledge that is derived from available sources such as thesauri, and domain experts that use the system. This store is organized as semantic net [Quillian 68] with the concepts being the nodes and the links being the relationships. Stored with the concepts nodes is their frequency of occurrence in the document collection. Included with the normal conceptual relationships is a statistical nearest neighbor relationship that reflects the occurrence of concepts together in documents of the collection.

The documents are represented by lists of concepts that occur in them (authors are also considered concepts) and their frequency in the document. The lists are determined using a standard automatic indexing technique [Porter 80]. Additionally, citation information is retained along with the document nearest neighbors, which is a link based on the similarity of the representations of two documents. Other information such as the date and journal is included. The combination of the user domain knowledge model, global domain knowledge model, and document database forms the concept/document knowledge base.

The concept/document knowledge base supports all of the traditional search techniques, as well as providing a structure that the user can browse. Browsing is considered

an important alternative method for finding information and incrementally specifying an information need. It allows users to use their recognition abilities to confirm or deny the relevance of items presented for display. By doing this, a model of their information need is built up.

The original Hearsay II system had little in the way of requirements for sophisticated input and output, whereas information retrieval needs a sophisticated user interface. Consequently, a separate interface manager is added to provide a window-based environment. Where the experts are responsible for what is displayed and when, the interface manager is responsible for how information is displayed and collected. The interface manager communicates with the experts by placing messages on and receiving messages from the short term memory.

The overall organization of the system is shown in figure 1.1

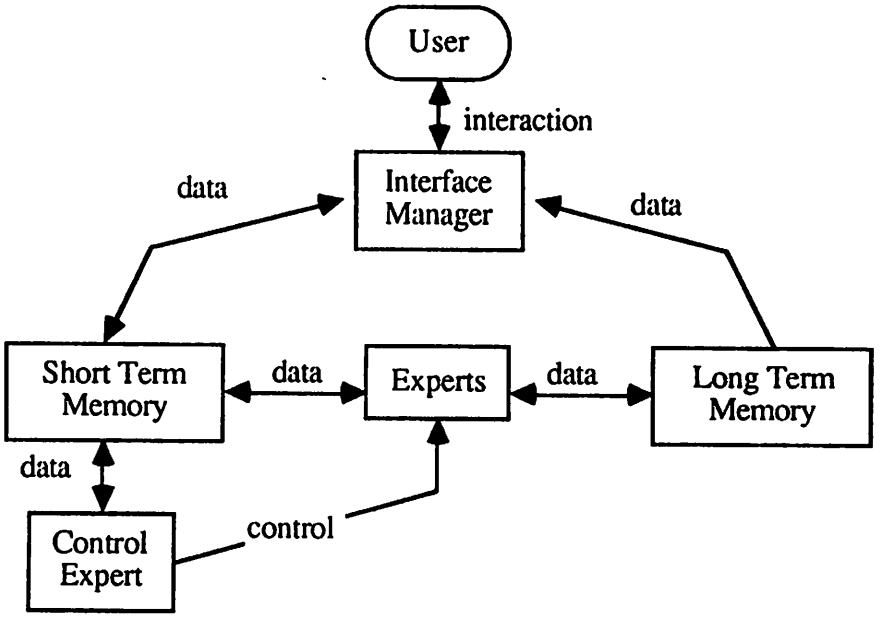

Figure 1.1: System organization of I^3R.

1.6 Organization

This chapter has provided a high level overview of the motivating factors behind and architecture of I^3R, Intelligent Interface for Information Retrieval. The remainder of this thesis is organized as follows. In chapter two an extensive review of the information retrieval field is presented. This covers the basic principles of IR research, the problems of traditional IR systems, and a number of systems that have attempted to attack different problems relating to IR interfaces. Based on this survey and analysis, chapter three sets forth the requirements for an intelligent IR interface. Chapter four presents the design and the implementation of a system, I^3R, that meets these requirements. Chapter five provides more detail on the browsing expert. Chapter six presents a discussion of the difficulties in evaluating a system such as I^3R, and three example scenarios that give the flavor of how I^3R operates. And, finally, chapter seven presents the conclusion and directions for future research.

1.7 Contributions

This thesis, presenting an architecture for an intelligent information retrieval interface, makes the following contributions to the art and science of information retrieval:

- The incorporation of multiple automatic search methods into one system.
- The integration of browsing, a user-directed search method, into a system with automatic search methods for the purpose of both query refinement and information search.
- A concept/document knowledge organization that supports the use of multiple search strategies, browsing and the use of user supplied domain knowledge.
- The first use of user stereotypes in an information retrieval system to control the adaptation of the system's operation to individual users.
- The use and integration of user supplied domain knowledge, which allows the system to retain its domain independent design, but allows it to gain domain knowledge with use.

- A flexible control structure that allows the system to alter its operation based on the user and the progress that the user and the system make in satisfying the user's information need.

- Implementation of the major functions of the system as separate rule based systems that allows each function to be incrementally developed, and allows new major functions to be smoothly integrated.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Introduction

In this chapter the background and problems of traditional information retrieval are first presented. Next, the concept and limitations of "intelligent" retrieval are discussed. This forms the basis for examining research directed at increasing the performance and usability of information retrieval (IR) systems in the fourth section.

2.2 Traditional Information Retrieval

2.2.1 Definition

An information retrieval (IR) system is concerned with providing an individual with access or references to documents or other text that contain information that is likely to be relevant to the user's expressed information need. This differentiates IR systems from fact retrieval systems that attempt to provide discrete pieces of knowledge, for example, the melting point of a substance or the salary of an individual. Although database systems are considered information retrieval systems, for the purposes of this work they are considered as part of an implementation base that would support both text and fact retrieval.

2.2.2 Components

An IR system has four major elements:

1. A representation of the content or meaning of the documents and the queries,

2. A process, usually called indexing, that maps the content of the documents and queries into the content representation,

3. A decision method, usually called a search strategy, that the system uses to determine whether or not a document should be retrieved,

4. A user interface.

Each of these elements will be discussed in turn in the following sections. Additionally, consideration must be made of how the representations are organized and what other information is required for search and indexing.

2.2.2.1 Representation

The representation most often used is a keyword approach, where the text contained in the system is represented by a list of terms or concepts that are indicators of the information contained in the text. In order to conserve storage the text of these terms is replaced with a number, and a dictionary is maintained to map the number to the original text. Along with this, information such as the authors, date of publication, language, and journal issue identification may also be kept.

An alternative to keywords is a full text representation, where the entire text of the document is stored and available for search. Full text systems are typically used in domains such as law where the specific wording of text is of particular importance.

Additional information is often kept to support more sophisticated search techniques. In the document representations, the frequency of the concepts is retained. This information allows search methods to know what terms are important in a document, based on the assumption that the more frequent a term is in a document, the more important it is. Furthermore, some systems store the reference or citation information. This information may be composed of pointers to the documents in the reference list (cited documents) and, additionally, pointers to documents that reference a document (citing documents).

2.2.2.2 Indexing

The text must be transformed from its raw state into the representation. This is accomplished by either manual indexing, or automatic indexing.

2.2.2.2.1 Manual Indexing

Manual indexing systems rely on human judgement to decide what are appropriate keywords for describing a document's content. It is accomplished by a person reading the text and assigning keywords. These may be from a controlled vocabulary, like MeSH (Medical Subject Headings) that specifies all the keywords that can be used and their hierarchical relationships, from the text itself, or supplied by the user himself.

2.2.2.2.2 Automatic Indexing

Automatic indexing takes the text and removes stopwords, which are high frequency words like "and," "was," "the," and "it." For a typical list see [Van Rijsbergen 79, pp. 18-19]. The remaining words are stemmed (i.e., have their suffixes removed using a standard algorithm [Porter 80]). The stems are then used in a table lookup in a dictionary that maps stems to term numbers to find the corresponding term number. The primary reason for this lookup dictionary is to reduce the amount of space required to store all of the document representations. This is significant, since some document databases have millions of documents. If there is no corresponding number, a new is one generated and the stem/term number pair is added to the dictionary. This dictionary also contains a count of the occurrences of a term in the entire collection. This information is useful in various retrieval techniques to be discussed in the section on automatic retrieval. The term numbers for all the terms in the document along with their frequency in the document form the document representation.

Queries are processed in a similar manner. The only difference is that if there is a term in the query that is not found in the collection, it is deleted from the query, and not inserted into the dictionary.

To illustrate the automatic indexing process, consider the following document abstract :

> The problem of the mutual exclusion of several independent processes from simultaneous access to a critical section is discussed for the case where there are two distinct classes of processes known as "readers" and "writers." The "readers" may share the section with each other, but the "writers" must have exclusive access. Two solutions are presented: one of the cases where we wish minimum delay for the "readers"; the other for the case where we wish writing to take place as early as possible.

The first step is to remove the stop words and sort the remaining words, which results in the following list.

> access, access, case, case, case, classes, critical, delay, discussed, distinct, early, exclusion, exclusive, independent, known, minimum, mutual, place, possible, presented, problem, processes, processes, readers, readers, readers, section, section, share, simultaneous, solutions, take, two, two, wish, wish, writers, writers, writing

Next the list of words is stemmed, and the list is compressed to add up the number of times that a stem is found in the document. The original words in this example are saved since this process is also used on the query, and the original words can be used to find related terms. The result is the following list of forms (in this example, the indexing algorithm is implemented in Lisp, and so it produces Lisp forms):

```
(access 2 (access)) (case 3 (case)) (class 1 (classes))
(critic 1 (critical)) (delay 1 (delay))
```

```
(discuss 1 (discussed)) (distinct 1 (distinct))
(ear 1 (early)) (exclus 2 (exclusion exclusive))
(independ 1 (independent)) (known 1 (known))
(minimum 1 (minimum)) (mutual 1 (mutual))
(place 1 (place)) (possibl 1 (possible))
(present 1 (presented)) (problem 1 (problem))
(process 1 (processes)) (reader 3 (reader))
(section 2 (section)) (share 1 (share))
(simultan 1 (simultaneous)) (solut 1 (solutions))
(take 1 (take)) (two 2 (two)) (wish 2 (wish))
(writer 2 (writers))
```

Next, the term numbers that correspond to the stems are looked up in the stem dictionary resulting in the following.

```
(10191 88 access 2 (access)) (11384 128 case 3 (case))
(36481 146 class 1 (classes))
(38351 26 critic 1 (critical)) (11673 5 delay 1 (delay))
(33740 356 discuss 1 (discussed))
(20610 21 distinct 1 (distinct))
(21787 21 ear 1 (early))
(25518 18 exclus 2 (exclusion exclusive))
(10445 71 independ 1 (independent))
(18391 79 known 1 (known)) (637 1 minimum 1 (minimum))
(3540 18 mutual 1 (mutual)) (25845 23 place 1 (place))
(29239 81 possibl 1 (possible))
(27663 253 present 1 (presented))
(31459 452 problem 1 (problem))
(2012 427 process 1 (processes))
(646 70 reader 3 (reader)) (9112 25 section 2 (section))
(9943 99 share 1 (share))
(12768 50 simultan 1 (simultaneous))
(6723 279 solut 1 (solutions)) (19792 50 take 1 (take))
(20170 8 two 2 (two)) (4183 10 wish 2 (wish))
(6626 126 write 3 (writers writing))
```

If this is a new document, the second number represents the frequency of the term in the collection, including the occurrences of the term in the new document. If this is a query, the number would not include the frequency count in the query. This list is then reduced to the term numbers and frequencies in the document resulting in:

```
(10191 2) (11384 3) (36481 1) (38351 1) (11673 1)
(33740 1) (20610 1) (21787 1) (25518 2 ) (10445 1 )
(18391 1) (637 1) (3540 1) (25845 1) (29239 1) (27663 1)
(31459 1) (2012 1) (646 3) (9112 2) (9943 1) (12768 1)
(6723 1) (19792 1) (20170 2) (4183 2) (6626 3)
```

Depending on what other kinds of information are maintained with the document, this is stored as the document's representation in the document database. If this process is applied to a query the complete information would be saved and forms the request .

The document database consists of all of the representations of the documents and the stem dictionary that maps the stems to the term numbers. Most systems also have an inverted file that has for each term, the documents in which it occurs. The use of an inverted file, while causing as much as a 100% increase in the space needed to store the document database, significantly increases the efficiency of searching.

2.2.2.3 Search Methods

There are two basic methods for deciding what documents to retrieve, user-directed and automatic.

2.2.2.3.1 User-Directed Methods

User-directed methods are characterized by being highly interactive and generally very slow. The first kind of user-directed method, which is found in nearly all commercial systems, is based on Boolean logic. In this case the form of the query and the decision method are the same. The query is essentially a decision rule in which the Boolean ex-

pression is the condition part and retrieval is the action part. A document is retrieved if it fulfills the conditions exactly. Most commercial systems also provide proximity operators so that the user can also specify phrases in the query formulation. In addition, these systems use an inverted file of the documents, containing for every term in the collection, the documents it is in.

Search in these systems is generally not performed by submitting a query and waiting for the results. Typically, the searcher will use the query as a guide to plan a series of actions that will retrieve documents. Often it is the case that the searcher will replan as he gets feedback during the search. The process begins by retrieving an initial set or sets of documents by using the terms provided by the user. Then, new terms or constructions made with adjacency operators are included using the AND, OR, and NOT operators as specified by the query expression. The query can be broadened by adding terms using the OR operator causing more documents to be retrieved, or it can be narrowed by using the AND and NOT operators causing fewer documents to be retrieved.

Another kind of user-directed method is *browsing*, which is characterized as an informal search that uses the structural links or connections between items in an organized body of information to look for relevant information. Browsing is often pursued when the user does not have a firm idea of what information exactly he desires, but has a general idea. He may use a classification system to help locate a large group of documents or books that are in the general area of his topic. From there he will pick some initial entry point and start to explore.. He will view and evaluate information in relation to his need, which may cause him to adjust his topic of interest.

2.2.2.3.2 Automatic Methods

Automatic methods are characterized as more batch oriented than the user-directed search methods. The user generally develops some specification of his information need (a

query), submits it to the system, and waits for the results. The differences in the methods lie primarily in how they interpret the information provided by the document collection and the query, in the form of term frequencies.

2.2.2.3.2.1 Coordination measures

There are a variety of automatic search methods. The simplest kind of method counts the number of terms that the document and query representations have in common; this is called the coordination level. Retrieval is based on the ranking of the coordination level, but no ordering is done within a level. To take into account the differing sizes of documents and queries, the co-occurrence measure can be normalized. A typical normalized measure is Dice's coefficient [Van Rijsbergen 79] which is:

$$2 \cdot \frac{|D \cap Q|}{|D| + |Q|}, \tag{2.1}$$

where D is the set of terms representing a document, Q is the set of terms representing the query, and |.| is the number of terms in the representation. For example, consider the following document, D, and query, Q, where

D = ((1, 1) (2, 2) (3, 1) (4, 3) (5, 2) (6, 1)
(7, 1) (8, 1) (9, 2) (10, 2) (11, 2) (12, 6))

Q = ((6, 1) (7, 4) (10, 1) (12, 2) (15, 1) (16, 2)
(20, 1)),

collection = ((1, 21) (2, 23) (3, 14) (4, 16) (5, 81)
(6, 13) (7, 62) (8, 23) (9, 17) (10, 5)
(11, 14) (12, 18) (13, 23) (14, 31) (15, 6)
(16, 14) (20, 5)).

The first value of the pair is the term number and the second is the frequency. The value for Dice's coefficient then is $2 \cdot 4 / 10 + 7 = 0.47$.

2.2.2.3.2.2 Vector space model

A more sophisticated approach is to consider the document and query representations as vectors in an n-dimensional space, where n is the number of unique terms in the collection. The similarity between a document and a query can then be formalized as the cosine of the angle between their respective vector representations, and is expressed as follows.

$$\frac{\sum_i (d_i \cdot q_i)}{\sqrt{\sum_i d_i^2 \cdot \sum_i q_i^2}} \qquad (2.2)$$

Where d_i is a term in D and q_i is a term in Q. The value for the cosine between the query and the document , from the previous example, if frequencies are not taken into consideration is $4 / 9.16 = 0.43$.

Search is performed in these systems in the following way.

1. The query, Q, is indexed, converting it to a list of terms and their frequency in the query.
2. If no more terms, done, else get the next term.
3. Get the documents that this term occurs in.
4. If no more documents, then go to 2, else get the next document, D.
5. If this document has not been seen before, compute cos(D, Q) and place it in the ordered list of documents that have been seen.
6. Goto 2.

Associated with the terms can be weights which reflect the importance of the term in a document and in the entire collection. Based on empirical studies [Sparck Jones 77, Sparck Jones 80, Salton 83], the inverse document frequency (idf) weight has been shown to increase retrieval effectiveness. This weight is based on the observation that the less

frequently a term is found in the collection the better discriminator it is. It is expressed as either log(N) - log (n_i) + 1, or N/n_i where N is the number of occurrences the most frequent term in a collection and n_i is number of occurrences of term *i*. Additionally, this can be multiplied by the term frequency (tf) within a document or a term significance weight (tsw) which is m_j/M, where M is the frequency of the most frequent term in the document and m_j is the frequency of a term in the document, so that the weight for a term in a document (d_i) is tf × idf or tsw × idf. This weight is based on the observation that the importance of a term in describing a document is directly related to the number of times it occurs. In essence, if a term occurs many times in a document,then that is a good indication of what the document is "about."

2.2.2.3.2.3 Probabilistic model

The next group of methods are based on probability theory [Van Rijsbergen 79]. This interpretation means that the system selects documents that have a high probability of being relevant to the query. The documents and queries are represented by the same sets of keywords or terms and use the same frequency information as the previous searches. The decision to retrieve is also based on a ranking with each document being scored by the following function:

$$\sum_i T(x_i)\ W(x_i)\ x_i\ , \tag{2.3}$$

where x_i is the *i* th term in the vector of terms describing a document, $T(x_i)$ is the term significance weight (tsw), $W(x_i)$ is a weight related to the frequency of term *i*, in the collection of documents and to its frequency in relevant and non-relevant documents [Robertson 76, Sparck Jones 80]. The weight for each term is computed as:

$$\log \frac{r\ /\ (R\text{-}r)}{(n\text{-}r)\ /\ (N\text{-}n\text{-}R\text{+}r)}\ , \tag{2.4}$$

where r is the frequency of a term in relevant documents, (n-r) is the frequency in non-relevant documents, (R-r) is the absence of a term in relevant documents, and (N-n-R+r) is the absence of a term in non-relevant documents. In practice 0.5 is added to each numerator and denominator to avoid division by zero [Sparck Jones 76], so that the weight is

$$\log \frac{(r + 0.5) / (R - r + 0.5)}{(n - r + 0.5) / (N - n - R + r + 0.5)}. \qquad (2.5)$$

Since, in an initial search there is no information with regard to the frequency of a term in relevant or non relevant documents, this weight is estimated using the idf weight [Croft 80].

This computation is motivated by Bayesian decision theory [Van Rijsbergen 79]. The weight is also based on the assumption that terms are independent. Considering the case where terms are dependent requires more information than can be reasonably determined from the collection statistics.

However, the idea of term dependencies is useful and leads to some information that can be reasonably be used. An assumption can be made, called the *Association Hypothesis* [Van Rijsbergen 79], that if a term is a good discriminator of relevance and non-relevance then a closely associated term should also be a good discriminator. This leads to the supposition that the terms in the database can be organized into clusters of terms that are related because they are used together frequently in the collection of documents. This information can then be used to expand queries by adding terms that are closely related to the terms that the user provides. The clusters can be organized prior to search for the entire collection using an efficient nearest neighbor algorithm [Croft 84, Croft 86]. The measure of two terms "closeness" is based on the frequency of their co-occurrence in the collection using Dice's coefficient.

A similar hypothesis, called the *Cluster Hypothesis* [Van Rijsbergen 79] can be made about the documents. It states that closely related documents tend to be relevant to

the same queries. Therefore, it would seem worthwhile to determine what documents are closely related in content. This can be accomplished with the same algorithms used to cluster terms. There are a variety of different ways to organize clusters of documents [Salton 83, VanRijsbergen 79]. One way is to compute a *centroid* of a group of documents, which is an "average" of the documents that compose it. Then a centroid is computed for a cluster of centroids, and so forth leading to a hierarchic clustering of the entire collection. Search in this organization starts by examining the top level clusters, and choosing the one that is most similar to the query. Then this cluster is examined in the same way, and so forth until there are a reasonable number of documents, usually around 20, left to retrieve.

Another method is called the *single-link* method. In this method only the the lowest level clusters are formed. The collection is searched by computing similarities with documents as is done in a non-cluster search, but instead of retrieving a single document, the document found and all other documents connected to it in its cluster are retrieved also. The links that are used to form these clusters are nearest neighbor links.

Clustering also can lead to more efficient searching, since only the cluster representatives need to be examined to decide what documents to retrieve rather that all the documents. There are many other considerations with regard to clustering, but they are beyond the scope of this work. The important thing about cluster searches is that they tend to retrieve different documents than non-cluster searches. The reason for this is shown in figure 2.1, where the query may share a great number of terms with the document, but share none of the terms that define the nearest relationship with another document. The nearest neighbor would be retrieved along with the document.

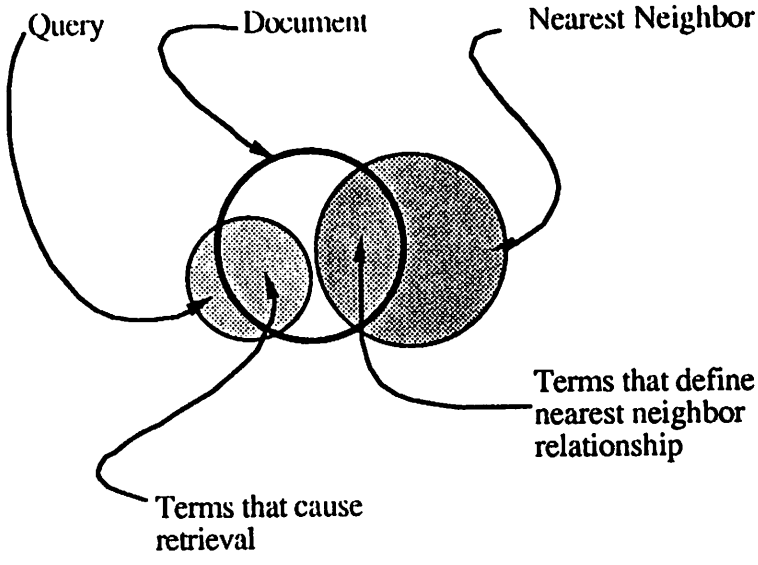

Figure 2.1: Showing how a cluster search can retrieve different documents.

2.2.2.3.2.4 Determining nearest neighbors

Determining nearest neighbors for both documents and terms is done using the same algorithm. In this discussion, documents will be used to explain the method. The most direct way to find nearest neighbors is to compute the similarity that a document D has with all other documents; this requires N (N-1) / 2 calculations. There are a number of optimizations, however, that can be made to significantly speed up the process . First, the assumption is made that only the closest nearest neighbor is needed (more if there is a tie). The maximum number saved is 5. Second, the documents must share at least one term. This means that a large number of documents will not be checked with a particular document since they have no terms in common. Third, a minimum similarity is placed as a threshold; this implies that documents must be more than weakly related to qualify; this value is 0.30 (This similarity is computed using Dice's coefficient).

Fourth, an upper bound (U_1) [Smeaton 81] can be calculated that will terminate the examination of any more of a document's term list when the remaining documents cannot have a larger similarity than the current largest similarity calculated so far. For example, a

document D has 20 terms and 7 terms have been processed so far. For any document not already seen, the largest similarity possible is if it contains the remaining 13 terms and only those terms. If the similarity is smaller than the smallest on the list of similarities calculated so far, processing for the current document is stopped. This bound avoids the calculation of similarities involving very common terms. It requires that the term lists of the documents are sorted in order of increasing frequency.

Another upper bound (U_2) [Murtagh 82] can be calculated while looking at documents not already seen. It is a variation of U_1 applied to the calculation of a single document. It is interpreted to say that a document,D′, not already seen, can have either the remaining terms on the current documents term list or fewer terms, whichever is smaller. If this similarity is smaller than the smallest on the list of similarities calculated so far, no similarity is calculated. This bound avoids the retrieval of many term lists, which is the most expensive part of determining nearest neighbors.

Fifth, since the documents are processed in increasing order of their unique identifier, save the highest value found so far for a document. For example, if in processing document 50 the largest similarity was with document 234, save that value with document 234 also. When processing comes to document 234, any similarity has to be greater than or equal to this value. And no document with a number less than 234 need be examined.

The algorithm for determining nearest neighbors is the following

1. Get the next document, D.
2. Get the term list for that document.
3. Sort the list in order of increasing frequency, if not already done.
4. Calculate the bound U_1.
5. If U_1 < the current neighbor similarity on the list or
   U_1 < 0.30 (the minimum similarity)
   Goto 1.
6. Get the next term from D's term list.

7. Get the list of documents for that term.

8. Get the next document, D′, from the list where the D′ > D.

9. Calculate the bound U_2

10. If U_2 < the current neighbor similarity on the list OR
    U_2 < 0.30 (the minimum similarity)
    Goto 8.

11. Calculate similarity = Dice(D, D′)

12. If similarity > previous largest similarity, for the pair (D, D′) replace it.

13. If similarity >= current neighbor similarity the replace it if > or add it if equal

14. Goto 8.

2.2.2.3.3 Network Representation

By combining the association and the cluster hypothesis and performing the nearest neighbor calculations on both the terms and the documents, a network representation of the document collection can be formed. This representation supports both cluster searches and normal search techniques efficiently [Croft 83, Croft 85]. The bounds U_1 and U_2 used in the calculation of nearest neighbors can also be used in the searches to make them more efficient. The only alteration is in the size of the number of documents saved, which determines the value of these bounds. In the nearest neighbor calculations, only the most similar document or term was sought, so the bounds were relatively high. In the normal search environment, this restriction can be adjusted to find a reasonable number of documents, usually about 20, for retrieval.

2.2.2.3.4 Relevance Feedback

After a search has been evaluated, the information provided by the user can be used to adjust the query so that it more accurately reflects his interest. This process can be applied to both the vector and probabilistic searches. In vector space model searches, terms

that are in relevant documents are given additional positive weights and terms that are in non-relevant documents are given negative weights. Of course, some terms may be in both categories in which case the weights will tend to cancel each other out. In probabilistic model searches, the occurrence of a term in relevant or non-relevant documents is already factored into the weight, so all that needs to be done is to fill in the values.

Relevance feedback can also provide additional terms for the query. A simple approach is to add all of the terms to the query from the documents judged relevant. This approach in systems that use automatic indexing tends to expand the query too much, adding many terms that are of little use. A more sophisticated approach is to have the user select terms from the relevant documents that are particularly interesting and add only those terms to the query. In this way, terms such as `interesting` from phrases such as "`An interesting approach is...`" are not added to the query.

2.2.2.4 User Interface

The user interface aspect has largely been ignored in the design of traditional retrieval systems. Most commercial interfaces are designed to work with hardcopy terminals, forcing the style of interaction, generally, to a command line orientation. Online help facilities are limited to command explanation and, if available, simple alphabetical listings of terms. To get help in selecting the proper terms to use, a user must generally refer to a printed thesaurus. More sophisticated interfaces have been developed, some based on the concept of menu selection, and others based on the use of a high resolution bit-mapped screen. A number of these kinds of interfaces will be discussed in relation to specific systems in section 2.5.

2.2.3 Evaluation

There are a variety of factors that can be evaluated with regard to information retrieval systems. Cleverdon [1966] identified the following six significant factors that can be measured.

1. **Coverage** — the extent to which all relevant material is included in the system.
2. **Time** — how long it takes from when the query is submitted to when the system responds.
3. **Presentation** — the form in which the system's output is displayed.
4. **Effort** — the labor on the user's part, either mental or physical, to use the system.
5. **Recall** (R) — the proportion of the material relevant to the request that was retrieved.
6. **Precision** (P) — the proportion of the material retrieved that is relevant to the user's request.

In the usual discussion of IR system evaluation [Salton 83, Van Rijsbergen 79], items 1–4 are usually passed over as being easy to measure, and the emphasis is placed on examining the last two items and measures related to them. Coverage is simply a matter of the content of the documents in the database in relation to the subject of a query. Time is often a matter of the kind of hardware, the system load, and the efficiency of the file organization or database system, as well as other factors.

Items 3 and 4 are related since the form of the presentation can cause the user to expend more or less effort depending on its effectiveness at displaying and capturing information. Suffice it to say that effective presentation of information will reduce the amount of effort that the user must expend while interacting with the system. The most effective system would have multiple ways of displaying information so that the user can use the form that he prefers. For example, many people find point-and-click menu selection interfaces very easy to use, while those that are accomplished typists find that using a mouse or trackball is a hindrance and prefer control key combinations to perform

the same tasks. Novice users often find menu-based systems easy to use, since they provide a structure that helps them avoid making mistakes and learn the function of the system. But as they become proficient with the system and know where information of interest is in the menu structure, the menus become an annoyance, especially if they have to go several layers deep to get to a particular selection.

Some systems have been evaluated with the idea of the user's effort in mind. One way this is done is to count the number of tokens that the user must enter. An example [Oddy 74] of this type of comparison is between THOMAS and MEDUSA, a medical information retrieval system. The tokens for MEDUSA are command names, concepts, system-assigned codes, and logical connectives The tokens for THOMAS are concepts, special words (YES, NO, and NOT), numbers from the user displays, and null messages (no comment). The average effort, over 32 queries, to use THOMAS required the user to enter 9.5 tokens; MEDUSA required 33.25 tokens. Both systems had approximately the same level of effectiveness. This kind of comparison gives a rough idea how easy one system or the other is to use.

Factors 5 and 6, are the two primary measures of IR system effectiveness. These measures as well as others are computed from the "contingency" table in figure 2.2.

	Relevant	Not Relevant	
Retrieved	r	n-r	n
Not Retrieved	R-r	N-n-R+r	N-n
	R	N-R	N

Figure 2.2: "Contingency" table for computing evaluation measures.

Precision is defined to be r / n and recall is r / R. Recall is calculated in an experimental situation when standard test collections that have known relevance judgements for the test

queries are used. These two measures are usually combined into a precision/recall (P/R) graph, which shows the precision at standard recall points, usually at intervals of 10 percent. Figure 2.3 shows a typical P/R graph. A P/R graph for a particular strategy for a given collection represents the results of all the queries in the test collection averaged together. The details of the averaging techniques can be found in Van Rijsbergen [1979] or Salton [1983].

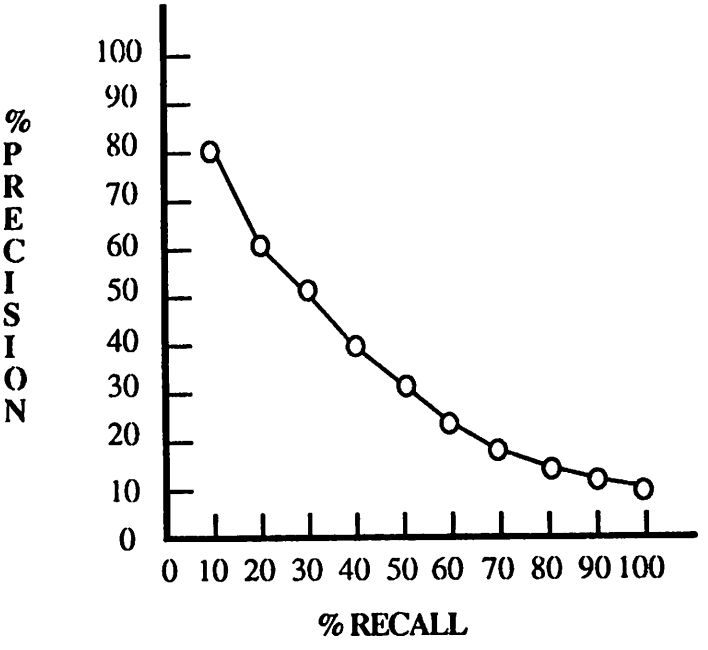

Figure 2.3: Typical precision/recall graph.

A measure that can be used instead of recall is fallout, which is (from figure 2.3) n-r / N-R. This measure makes more sense in an operational system where the number of documents relevant to a particular query is unknown. However, since comparative studies are done using test collections, fallout is seldom used in IR literature for comparing the effectiveness of different systems or search strategies. There are a number of other reasons why fallout may be preferred, which may be found in VanRijsbergen [1979] or Salton [1983].

The measures of recall and precision can be combined to give a single valued measure of effectiveness for a particular search strategy for a given collection. One way is to

take the average of the precision values at various recall points. Another composite measure is the E-measure [Van Rijsbergen 79], which is given by

$$1 - \frac{1}{\frac{\alpha}{P} + \frac{(1-\alpha)}{R}}, \tag{2.5}$$

where $\alpha = 1 / (\beta^2 + 1)$ and β represents the relative importance attached to precision and recall. A value of 1 for β indicates equal importance to precision and recall. Values greater than 1 indicate greater emphasis on recall; less than 1 on precision. Smaller values of E indicate greater retrieval effectiveness.

Another single-valued measure that has been proposed is search length, which is defined to be the number of non-relevant documents that the user must examine before his information need is satisfied. If the system produces only relevant documents, then the search length will be zero. This measure can used in systems that provide an ordering of the documents retrieved or as in THOMAS [Oddy 74, 77] produce one document at a time for evaluation. A variation of this measure is to count the number of non-relevant documents presented before the first relevant document is presented. This is used to estimate, in THOMAS, the amount of work expended in formulating the query. Ofori [1982] also used this measure in the estimatiton of the amount of effort that was expended in the course of a search session.

While testing different search methods with standard test collections gives a basis for comparing their relative performance and for testing the effects of new strategies or weighting methods, a few things should be kept in mind. First, the concept of relevance for a document is subjective. Different users have different needs, which will be reflected in the kind of document that they choose as relevant. Second, the relevance judgements of test collections were made by domain experts; each expert developed one or more queries and looked through the collection to find the documents that were relevant. Therefore, the

relevance judgements may reflect the biases of the experts or overlook what might appear to be relevant to an unsophisticated user. Another consideration is how the relevance judgements were made. In some collections, the experts had a fairly comprehensive idea of the documents in the collection, so they could make reasonably knowledgeable evaluations. In other collections, the experts derived queries, submitted them to a retrieval system, evaluated the results, and then followed citations from the retrieved documents to others. Even with these considerations, the use of standard test collections represent the only way developed so far to perform repeatable experiments.

2.3 Retrieval Problems

The advantages of the traditional IR systems are that they are easy to implement, relatively fast, and those that use automatic indexing are domain independent. However, there is a major problem, and that is the overall performance of these systems is low. The average precision of the advanced statistical systems varies from 50% (unusual) to 25% (more typical) taken at recall intervals of 5% from 90% to 5% [Fagin 87]. One of the causes of this problem is that traditional systems are limited to only a single way of responding to every user and every type of problem. This limitation manifests itself in two important ways, in query formulation and in the search process.

2.3.1 Query Formulation Problems

Query formulation is the process of expressing an information need, which is in the mind of the user, in the best possible way in the form required by the system. Certainly, the best way is one that provides the system with the necessary information to retrieve a high proportion of the documents that the user wants and a low number of unusable ones. The perfect search would retrieve all and only those documents that the user wants or needs. The results of a search can only be as good as the description of the items being

sought. Therefore, if the user cannot adequately define what he is looking for, the system cannot produce adequate results. There are two major obstacles to the user in expressing his information need in an effective way. The first is the form of the query required by the system, and the second is the ability of the user to express precisely the information need. Some users may know exactly what they want and can describe it precisely, while others may not be able to do so.

2.3.1.1 Form of the Query

Most commercially available systems require that the user cast his need into a formal notation using Boolean logic. Users not familiar with Boolean logic find it difficult to use. The NOT operator, particularly, causes problems, although it is essential for getting good results [Vigil 83]. Furthermore, the Boolean connectives AND and OR do not have a sufficiently precise meaning with regard to the relationships between words. This lack of precision can confuse the casual user. For example, when two words, such as, "information" and "retrieval," are ANDed together, does the user mean that the two words form the phrase "information retrieval" or, simply, that they both are required to be present. The OR connective also presents some ambiguity. Words combined with OR are usually taken to be synonymous, but OR can also be used to list the components of a higher level concept. For example, the concept of "additive primary color" can be expressed by the list: "red OR green OR blue." AND and OR can also be used interchangeably in everyday usage, as in the sentences: "I do not like peas and carrots," and "I do not like peas or carrots."

To overcome some of these problems, some systems allow the user to specify proximity information in the query, but these additional operators can be a source of confusion adding further difficulty to query formulation. It is clear that when two words are specified to be immediately adjacent that the user is specifying a phrase, but what the user

means is less clear if the proximity is 5 words or 20 words. Perhaps the user means that the words must occur in the same sentence or the same paragraph. Some systems that retain the full text of the document provide more meaningful sentence and paragraph proximity operators. The addition of more operators, however, requires that the user make more of an investment in learning how to formulate queries, which he may not want to do.

Systems using automatic indexing are easier to use, allowing the user to enter his query as free text. This text is then indexed, generating the same internal representation as used for the documents. Although this form is easier to use, it suffers from the loss of information regarding relationships between words. It also can pick up spurious words, depending on how the user phrases his request. For example, if the request leads off with the phrase, "I am interested in...", the word "interested" will be part of the representation of the query, even though it has nothing to do with what the user is looking for.

2.3.1.2 Content of the Query

Inability to express the content of the query has two causes. The first cause is that the user lacks knowledge of the terms and their interrelationships of the domain he is searching. It may be that the user has a definite idea of what he wants, but cannot find the appropriate terms. For example, a scientist or engineer may change his primary field of interest and cease to keep up with developments in his former field. At some time, he may need to look for information in his former field, but may not know new terminology. A searcher that does not have a good understanding of the area in which he is working cannot select the necessary words to compose a well defined query. For example, a person searching in a medical domain may be interested in all the information about a particular class of diseases. The documents describing those diseases may or may not refer to the general class name when discussing a particular disease, and if the documents in the collection are automatically indexed, then that class name will not be in the document

representatives. If the user does not provide specific disease names as well as the general class name, a number of potentially relevant documents may not be retrieved.

The second cause is that the user does not have a definite idea of what he wants but perhaps, only a vague notion or hunch. This is not an unusual experience. It stems from the user trying to describe something that he does not know. An example might be someone who wants to know about the topic of "color." Is the user interested in the physics of colored light, the psychological effects of color, the physiological aspects of color vision, or color theory as it relates to art to name a few of the possibilities. The information desired may actually draw from all of these areas, but at the time the user comes to the system, he may not really know exactly what he wants. In this case, the user may need to examine results of a few searches to help him clarify his need. This kind of difficulty is the basis for THOMAS [Oddy 77] and RABBIT/ARGON [Patel-Schneider 85]. Both of these systems assist the user by showing him examples of what his current query will retrieve, so that he may alter them to better match what he really wants.

2.3.2 Search Methods

Search methods in Boolean retrieval are directed at getting a set of documents small enough so that the user can conveniently examine all the titles and abstracts. Since the query expression acts as a filter on the collection selecting only those that match, no ordering of the documents is done. Each document is considered as equally relevant. Consequently, the user must examine all of the documents in the retrieved set to assure himself that no document of value is missed.

The feedback for manipulating the query is indirect, consisting primarily of tracking the size of the retrieved document set, and occasionally looking at some of the documents in the current set. Only when the size of the set is in the neighborhood of 10 to 20 will the user, if present, look at all of the documents. If a search intermediary is doing the search-

ing without the user in attendance, he must use his judgement based on the knowledge of the topic he has gained from the user and from his own background to determine whether or not the results are adequate.

2.3.3 Search Intermediaries

The difficulties in using traditional systems is most often overcome by employing people called search intermediaries or search analysts [Barraclough 77, Marcus 78]. These individuals have specialized knowledge in 4 areas:

- Knowledge of command languages for different systems.
- Knowledge of the subject coverage of particular databases.
- Knowledge of how to use Boolean logic to formulate a query.
- Knowledge of how to elicit information from the user about the information need.

They remove from the end user the burden of having to concern himself with the details of the command language, the selection of which database to search, and how exactly to formulate a query. The problem with this approach is that the person with the information need is not doing the searching. No matter how good the intermediary is, there is no way that the end user can really convey the exact nature of the need. The preferred arrangement is the person with the need working with the intermediary as the search is being carried out. In this way, the intermediary can get immediate feedback on the progress of the search. This, however, is often not the most economical arrangement, since it can slow down the intermediary and increase the connection charges.

Another difficulty is that intermediaries are not always available at the time when the user may be looking for information. For example, university students often search for information during the evening and on weekends, and people who use services such as Compuserve tend to search at night when phone rates are at their lowest. Another consideration is that with the advent of optical disk technologies, such as CD-ROM, large

amounts of information may be available to many people at a relatively low price for use in their homes or businesses. People using search systems based on CD-ROM technology in their home or business may not have access to intermediary services, such as are available in large research institutions or corporate technical libraries.

2.4 Intelligent Text Retrieval

With the success of knowledge-based systems in areas such as medical diagnosis, computer system configuration, and others, the developments in natural language processing techniques, and computer aided instruction, it appears that artificial intelligence (AI) may have techniques to contribute that will make IR systems more effective and overcome the problems described in the previous section. This is especially attractive, since research in improving statistical systems using purely statistical means appears to have reached a peak [Belkin 87a].

Knowledge-based, often called "expert", systems technology seems to be applicable since there are search intermediaries who are experts in conducting retrieval sessions and since there is no standard method to conduct a retrieval session. Intermediaries develop their own sets of heuristics or rules of thumb. These heuristics deal not only with the details of the actual search, but also with methods for eliciting the information need from the user, and transforming it into a query. Bates [1979a, 1979b] presents some heuristics related to search. These heuristics can be encoded into rules that form the basis of the typical expert system [Hayes-Roth 83].

The work in story understanding, fact retrieval, and translating natural language database queries also appears to be especially appropriate. The focus in story understanding [Wilensky 84] and fact retrieval work [Kolodner 83] has been to develop representations for the meaning of narrative text so that paraphrases of the story can be generated and questions can be answered about the actions and intentions of the characters. The focus of

the database query work has been to provide user-friendly interfaces that do not require a user to know the formal query language of the database in order to get information. As an example see [Salveter 87, Waltz 78].

An example of a sophisticated natural language interface system is UNIX Consultant (UC) [Wilensky 83], the purpose of which is to provide online help in using the UNIX operating system. One of the features that makes this system sophisticated is the ability to analyze the user's statements to infer his goals. For example, a user enters the query:

```
I'm trying to get some more diskspace.
```

An appropriate response is:

```
Delete files that you do not need,
```

or

```
Ask the system manager for more space.
```

The latter implies that the system is aware of the user's desire to save files, and that getting more space in his account will satisfy that desire.

Based on these ideas, one could envision an intelligent retrieval system that would assist the user in expressing his information need so that the system could understand it and determine an appropriate response. This idea of applying AI to IR has led to several definitions of what an intelligent retrieval system might be. One definition of an intelligent retrieval system is given by Sparck Jones [1983]:

> ...an intelligent retrieval system would have an inferential capability so that it could deploy prior knowledge to establish, if not by certain reasoning at any rate by plausible reasoning, a connection between a request and a candidate relevant document.

Brooks [1987] expands on this definition by incorporating more awareness of the user into the requirements of the system. She states that given the following three conditions

1. A user who has come to an information service with a problem for whose resolution or management, information is required.

2. The user is unable to specify exactly the information needed, since this is describing what the user does not know.

3. Access to a number of document databases.

The system must, by use of its knowledge of its world (of documents, users and topics), and its knowledge of the specific user and problem, infer documents that will enable the user to better resolve or manage the problem. The inclusion of awareness of the user increases the complexity of the system dramatically, since it now must handle users that have varying levels of domain knowledge and computer experience, as well as widely differing search goals.

An intelligent text retrieval system would use artificial intelligence techniques to improve each of the four system elements defined previously. AI techniques for text representation and indexing elements are intimately connected. Often, when one picks a representation scheme the indexing or parsing method comes along with it. There are a number of difficulties with using the currently available techniques. First is the problem of dealing with enormous amounts of domain knowledge. Not only does one have to consider the number of possible concepts that a general information retrieval system has to deal with, but also the number and types of possible connections between them. The typical approach has been to restrict the application to a narrow domain. This reduces the volume of information to a level at which domain knowledge representation can be built by hand. For example, UC only deals with the UNIX file system. This kind of approach is not appropriate in IR since real systems often must provide access to hundreds of thousands of documents. Applicable AI models for representing domain knowledge are semantic nets [Quillian 68] and rule representations [Tong 87].

Another problem is selecting the appropriate representation for document meaning. At this point there is no universally accepted method of representing the meaning of natural language. There are a number of possibilities, which include conceptual dependency theory [Schank 75], Montague semantics [Dowty 81], and logic representations [Simmons 87].

Besides the difficulty of representing the knowledge contained in documents and their domain is the difficulty in searching to find relevant information. The methods on how to do this are not well established. Fact retrieval systems [Kolodner 83] use sophisticated matching algorithms on their internal representation structures that may not be efficient enough to work in an IR setting. Furthermore, the intermediary knowledge that does exist about searching, that could be used in construction of an expert system, is oriented toward optimizing the performance of commercially available systems.

Van Rijsbergen [Van Rijsbergen 87] has taken some initial steps in characterizing the logical foundation necessary for intelligent retrieval decision methods, although the concept of "logical relevance" appears as early as 1971 [Cooper 71]. Instead of considering retrieval as being based on a similarity or a likelihood, it is considered as an inference that allows the system to determine if a particular document implies the user's query. Associated with this inference is a measure of its certainty. If the measure of the certainty is sufficiently high, then the document will be selected for presentation to the user as a candidate relevant document.

The significant point about this reformulation of the document selection process is that it relies on an accumulation of evidence to determine the certainty of the implication. This evidence may come from a variety of sources. One of the difficulties with the formulation is the definition of the semantics and its representation. This is an open research problem. The traditional keyword approach represents a primitive approach to representing the semantics of text. However, keywords are only one possible source of evidence. Other possible sources are citations and other kinds of user defined links.

Even if one could develop a system that retrieves documents intelligently based on the natural language description of the information need, the nagging query formulation problems remain. The use of natural language as the means of expressing the need by no means solves the problem. Each user will have a different degree of ability with natural

language. The user must still take time to carefully draft his query or the system will have to have the ability to handle ill-formed and perhaps ambiguous need statements. Furthermore, the system will have to deal with users that do not have a clear idea of what they want, besides not being able to express it.

The work of intelligent retrieval has for the most part, as traditional retrieval work has, focussed on the objects of retrieval, documents, queries, and inference methods, and not on the end user. Consideration of the end user has a significant impact on the design of the interface element of the system as well as the search element. It introduces a new dimension to the kinds of knowledge that the system must consider in order to be effective. But, by doing this the system builder can take advantage of the expertise of the search intermediary in helping the user refine his information need.

2.5 Analysis of Systems

The systems to be reviewed in this section are organized by their major emphasis, which corresponds to what element of an information retrieval system they are trying to improve. The systems are organized by what is considered to be their major feature or emphasis relative to the elements that compose an information retrieval system. The initial section concentrates on the representation of meaning and the organization of the text elements.

2.5.1 Representation of Meaning

Research into the improvement of the representation of the content of a document fall into two primary categories, those that use natural language processing (NLP) techniques and those that do not. One motivation for attempting to use techniques that do not use the NLP techniques is to reduce or eliminate the need for the large amounts of domain

knowledge that is required. Another is to retain the efficiency that automatic indexing provides.

2.5.1.1 Non Natural Language Approaches

The primary work in this area has been by Belkin [Belkin 82a, Belkin 82b, Belkin 84]. The underlying motivation for this work is to model the user's "anomalous state of knowledge." The attempt to resolve this perceived anomaly is what causes the user to come to an information retrieval system. The actual representation of the user's information need and representation of the document content is based on the proximity of word pairs in the text. Words that are adjacent are given a score of 12; those that are in the same sentence are give a score of 4; and those in adjacent sentences are give a score of 3. These values are then taken and used to derive a graph representation of the top 40 associations. The graphs can then be used in two ways. The first is to get feedback about the query from the user. The graphs show what the user thinks is important based on his expressed interest. The graph can be altered if it does not reflect the real information need. The second use is for search, where the graph representation is compared to documents that have been processed into the same graph representation. Various heuristics are used to match substructures in both of the graphs [Belkin 86].

Other work in representation has focused on the addition of citation information as an extension of the normal vector representation [Salton 83]. Citation information, however does not contribute to the representation of meaning of the text, but serves to connect it to other pieces of text that are judged to be relevant by the text's author. This topic is expanded in section 2.5.2.

2.5.1.2 Natural Language Approaches

An approach of using NLP in text retrieval is FASIT (Fully Automatic Syntactically based Indexing of Text) [Dillon 83]. This approach attempts to be completely automatic and functional without semantic information, using only information about the structure of English to extract meaningful phrases from text. This is accomplished in a two stage process: concept selection and concept grouping. Concept selection is done in three steps, which are: assignment of words to syntactic categories (tagging), disambiguation of multiply tagged words, and concept selection. Concept grouping is a two stage process. The first is formation of canonical form and the second is grouping by canonical forms. For details see Dillion [1983]. Since FASIT does not require domain specific knowledge, it is very general, and can be used in many domains, which is its primary advantage. It does not, however, perform significantly better than traditional stem-based systems.

More recently, in a system named ADRENAL [Croft 87a] consideration has be given to the incorporation of semantics. The system does not attempt to build representations of every document as they enter the system, nor does the system require large amounts of domain knowledge. Instead, it uses a general representation of the semantics of science and technology called REST (REpresentation for Science and Technology). The system also does not attempt to understand the content of documents and queries, but only garners enough information to make a judgement about the relevance of a document to a query. This process is made more efficient by not comparing the representation of the query to every document. Instead, a retrieval is made using traditional techniques, and the deep comparison, using the NLP techniques, is made with the results of the retrieval.

Other research in applying NLP techniques to IR has been performed by Spark Jones [1984], Smeaton [1986] as well as others.

2.5.2 Organization of Text Units

The first group of systems fall into the category of "hypertext" or "hypermedia" systems; the distinction being that latter incorporates more than just text. The emphasis in these systems is the organization of the units of information, where the units may be one of a number of types, such as text, graphical, pictorial, or audio. There is little emphasis on the representation of the meaning of the unit itself.

An early hypertext-like system is ZOG [McCracken 84]. In this system the basic unit of information is a 24 line by 80 column full screen of text. These textual units are then linked to form a network. ZOG has been used with success to implement a naval operations manual. Another similar system is the help system in the GNU EMACS [Stallman 87] text editor. In this system, the information is basically structured as a hierarchy with added cross reference links.

This system was applied to an information retrieval system and called BROWSE [Fox 80]. In these systems the only way to find information is to follow the links. BROWSE included a method of performing a "parameterized" content search which is carried out in the context of the user's current location. The primary organizing structure is the Computing Reviews classification structure. The level that the user is at determines the constraints on the search.

One difficulty with these systems is that the user, especially the novice user, has little context from which to determine where he is in the net. This can lead to a variety of behaviors, such as frequent backtracking or returning to the top of the classification structure to start over. The user is required to maintain a mental model of where he is in the system. This can be quite difficult in a large highly interconnected network.

Another difficulty is that these systems are by and large hand-built. While ZOG provides an integrated editor, there is no provision for the automatic addition of text to the system.

A more sophisticated system, which is representative of more recent implementations of hypertext systems is Textnet [Trigg 86]. The emphasis in this system is the support of scientific authoring and information sharing. Because of the emphasis in this system, there are 80 different kinds of links. The two broad classifications of them are Normal and Commentary links. The Normal links define the structure of a document, provide citation links, and allow other users to attach their own opinions. The Commentary links provide the means to attach information that is related to the style of the document. While there is no provision for the automatic inclusion of text to the system, it is an open system. The intended use of Textnet is as an authoring system to develop ideas. As with ZOG, no provision is made to include a normal search facility. User's must follow links to find the information that they desire.

Another similar system is the Hepatitis Knowledge Base (HKB) developed at the National Library of Medicine [Williamson 85]. Its intent is to provide quick access to textbook knowledge on viral hepatitis. It is organized as a three level hierarchy. The bottom level contains very specific detailed information, along with references to journal articles. As one moves up the hierarchy, the information becomes less detailed and of a summary nature. Built on top of this is a system called ANNOD that allows the user to make content based searches.

Dynamic Book [Weyer 82] is similar to the HKB in that it provides access to textbook information. Its purpose was, however, to study how people search for information. It provided a number of methods to allow users to find the answers to factual questions. It also was organized hierarchically but was limited to only three levels that correspond to chapters, sections and subsections. These levels had simple title-like sentences or phrases describing the content of the corresponding unit of text. The significant point about the Dynamic Book is that it used a sophisticated multi-window interface to give user different

views of the information. A user could trace down the hierarchy or could look at the titles alphabetically.

THOMAS [Oddy 77] has a unique way of organizing the text. Instead of relying on a schematically rich set of links, it uses untyped associations between authors, concepts, and documents. The program is not concerned what the associations are between elements, only that the elements are associated. Based on the user's responses, THOMAS selects portions of the network as an image of the searcher's interest.

2.5.3 Query Formulation Assistance

Many systems have been developed with the goal of helping the user find the best terms to describe his information need. The basis for most of these systems is a thesaurus. These, at the minimum, give synonyms, broader, and narrower terms, for a particular domain, if they exist. They may also provide other information such as the frequency of a term in the collection.

An early system along these lines is Eureka [Burket 79]. This system is a full text retrieval system that also kept an inverted file of every important word in a document. The thesaurus available to the user was personal and under the complete control of the user. From the information contained in this thesaurus the system could automatically make substitutions for synonyms. In this way the system maintains a model of the user's knowledge in the domains that he searches.

Most systems use global domain knowledge as a basis for providing assistance. This knowledge along with published search heuristics [Bates 79a, Bates 79b] has lead to a number of systems that are built as expert systems. A number of the more sophisticated systems will be discussed in the section on AI systems. Shoval in an unnamed system uses semantic nets [Quillian 68] to organize and select terms for user approval. Semantic nets are a natural way to organize this kind of knowledge. The system used the initial

terms of the query as a starting point for a "spreading activation" through the net. Spreading activation works by taking the initial starting points, in this case terms from the user query, and traversing the links connecting these initial points to all the immediate neighbors. These are then shown to the user for approval. If they are judged negatively, then no further activation is done from them. Of particular interest are when activations from different points intersect. Shoval's system maintains lists of terms that are found and presents them to the user for his evaluation.

Another system that uses an AI approach is CANSEARCH [Pollitt 84]. The basis for this system is the MeSH thesaurus, specifically limited to cancer research (oncology). Instead of using a network representation, the thesaurus is coded as frames which specify the possible selections that may be made. The control of the system is based on rules that handle four major activities: interaction with the underlying retrieval system; generating queries for the search system; selection of frames for presentation; and processing the results of term selection.

Much of the success of this system is due to the nature of the MeSH thesaurus. Since it is the indexing language for the documents, the searcher cannot choose terms that do not exist. Furthermore, the use of a touch screen eliminates the occurrence of misspelled terms. This can be a significant problem, since medical terms can be quite complicated. The system also acts as a guide to help the search select the right terms.

Two more recent and slightly more sophisticated systems are EP-X [Krawczak 85] and CoalSORT [Monarch 87]. Both of these systems model the knowledge of a highly constrained domain, environmental pollution and coal liquefaction respectively, as a frame-based semantic structure. The user is given assistance as a result of the system's examination of the knowledge-base. The initial entry points are, as in Shoval's system, the initial terms provided by the user. Each system uses the relationships between the different concepts to help the user refine his query. Although not specifically stated, it appears that

both systems use Boolean retrieval to get documents. Since the EP-X system, is hooked up to the Chemical Abstracts service, it approaches the performance of an operational system. CoalSORT, however, is specifically a prototype, having access to only 100 documents.

The significant feature about CoalSORT is its user interface. It is designed to show the user the structure of the knowledge base. At all times the user is aware of what he has seen and what he has selected as key concepts of the information that he is looking for.

2.5.4 Interactive Query and Search

The usual approach in text retrieval is to work on producing a good query by using available tools and then submitting it to the system and evaluating the result of the search. An integration of these steps results in a finer grained user/system interaction. An example of this approach is the program called THOMAS [Oddy 77]. To begin, the user submits a few terms to the system and the system shows the user one document that appears to be the most related to the initial statement of interest. The user then evaluates the document by indicating what keywords are relevant, what ones are not to be considered further, and the relevance of the document as a whole. From this, the system adjusts its image of the user's area of interest in the document collection and retrieves another document.

This kind of need refinement/search differs from traditional browsing in a number of ways. The first is that the system retains the control of the interaction. It selects the next document to view. Second is that the system constructs a model of the user's need. Systems that rely on browsing as the primary means of search simply provide a structure for the information and an interface to get at it. They may keep track of the node that the user has visited, but since most of them do not have any representation for the content of the nodes, they cannot build up a model of what the user has examined. The third is that

the user is not shown the structure of the information contained in the system. He is simply presented with a sequence of documents for evaluation.

Another system that uses a similar approach is RABBIT, later called ARGON [Williams 84, Patel-Schneider 84]. The paradigm that RABBIT proposes is *retrieval by reformulation*. The system's knowledge base is composed of a collection of heterogeneous facts that is organized using a frame-based knowledge representation language, KANDOR. Retrieval begins by the user specifying some category that he is interested in. The system responds by presenting some instance of that category. The user can then critique the instance in six ways. He can specify one of the following.

Require - adds an attribute to the query

Prohibit - adds the relative complement of an attribute to the query

Alternatives - suggests a series of alternatives and inserts those selected

Describe - allows the definition of an embedded query

Specialize - refines a generic category by allowing the selection of one or more subconcepts.

Predicate - applies a predicate to the value of an attribute.

The interface shows the user the components of the developing query, an instance of what the query currently specifies, and a list of other matches. This gives the user immediate feedback as to what he has specified.

2.5.5 Search Oriented Systems and Research

There have been a variety of approaches used for improving and performing search using AI techniques. The work in improving search has dealt primarily with using limited natural language techniques to provide more evidence to statistically based search methods. This work has involved two major steps, getting the evidence, which mainly consists of noun phrases, and incorporating it into the search method. The motivation for much of this

work is the existence of well-studied parsing techniques. Consequently, there are robust techniques that can be applied in a domain independent way.

Croft [Croft 86] expanded the basic probabilistic model by adding a factor that gives extra weight to documents that contain groups of related words derived from the user's query. Later work [Croft 87] uses more sophisticated natural language techniques to examine documents after they have been retrieved by probabilistic means to see if they have phrases that indicate relevance. Smeaton [1986] used parsing techniques to extract phrases from queries. Sparck Jones and Tait [1984] have investigated the use of parsing techniques to generate variants of search term phrases. Fagan [1987] has also pursued these ideas and has extended the cosine correlation to integrate phrase information.

Another technique that has been tried is the use of a connectionist system to select search strategies for particular queries [Croft 85b]. This work is based on the observation that even though one search strategy may be generally better than another, that for a particular query the less generally effective strategy may perform better. If a way could be found to select the best strategy for a given query, the performance of a retrieval system could be improved dramatically. Unfortunately, there was no feature or combination of features of the queries that could be used as a discriminator.

A system that performs conceptual information retrieval without using natural language techniques is RUBRIC [Tong 87]. This system is essentially an extension of a Boolean query retrieval system. Queries are composed of two kinds of rules. The first kind of rule maps occurrences of specified text strings into concepts. The condition part of this kind of rule uses AND, OR, and adjacency operators. The action part is the establishment of a concept with an associated degree of certainty. The second kind of rule connects concepts.

The primary purpose for RUBRIC is the monitoring of streams of message traffic such as occur on a newswire like the Associated Press. This kind of environment is the

inverse of the typical IR environment in which there is a slowly growing (i.e. fairly stable) collection of documents and a dynamic query environment. RUBRIC, however, has a stable collection of queries and a highly dynamic heterogeneous group of documents. Consequently, much effort is put into constructing the queries, so that they will give good results. The queries also represent the specification of a reusable source of domain knowledge.

A system that is not per se an information retrieval system, but in many of its aspects resembles one is GRANT [Cohen 87]. The particular problem that this system attempts to solve is matching research proposals to potential funding agencies. The information in the system is organized as a semantic net. Topics and agencies are linked by a wide variety of association links, including structural associations such as "heart `IS-A` organ" and setting relationships such as "heart `IS-SETTING-OF` heart attack." Other example links are `AFFECTED-BY`, `COMPONENT-OF`, `CAUSED-BY`, and `RESEARCH-TOPIC`. In all there are 24 links and their inverses.

Search is carried out by "constrained spreading activation." The constraints determine what paths are to be followed through the net. Some are simple such as stop at nodes that have a high (more than 16 links) fan-out, or stop if a path has more than 3 links. Other constraints are more sophisticated and are heuristic rules called "path endorsements". These may be either positive, which indicates that the path is to be preferred or negative, which indicates that the path is to be avoided. An example positive endorsement is:

```
If   request-funds-to-study( X ) and
   IS-A( X, Y ) and
   agency-funds-studies-on( Y )
Then the path is good.
```

A negative endorsement, for example, would have the IS-A replaced with an INSTANCE-OF relationship. The positive endorsement captures the concept that a funding agency might fund studies that are specializations of their stated interest, and the negative one cap-

tures the concept that agencies are not likely to fund studies that are more general than their stated interest.

The performance of GRANT has been judged acceptable by its users. The typical results are a list of 15 to 20 possible agencies, of which two or three are judged relevant by the user. In tests, 27 proposals were run against the knowledge base producing a recall rate of 67% at a precision rate of 29%, which is equivalent to the performance of IR systems in general.

Using a system like this involves a significant amount of knowledge engineering to construct the semantic net. This work entails analyzing the domain in which it will operate to determine the appropriate concepts and relationships.

2.5.6 AI Systems

Many of the systems described previously have used AI techniques to attack one IR problem or another. The following systems are broader in their scope, in that they attempt to provide a system solution to the problem of making retrieval systems more effective. Of particular interest, is the architecture of these systems; how are they organized to meet the requirements of an *intelligent* information retrieval system. Factors for consideration are the following.

- How are the traditional functions of a retrieval system implemented? These considerations are partially taken from [Chiarmella 87].
    - Implemented completely as a rule-based system.
    - Implemented as hybrid rule-based and procedural system.
    - Implemented using a different kind of AI architecture.
- How are these functions partitioned into different tasks?
- How are the tasks controlled? This depends on the kind of overall organization that has been selected.

2.5.6.1 IR-NLI

IR-NLI (Information Retrieval - Natural Language Interface) [Brajnik 85, Brajnik 87] is a prototype system for investigating the construction of cooperative man-machine interfaces for information retrieval. The basic direction of the work is to simulate the functions of a search intermediary in an expert system. The input to the system is natural language, the output is a search strategy, which in this case is a sequence of commands to be submitted to a Boolean query retrieval system.

The system is composed of five major modules. The Understanding and Dialogue Module is concerned with developing the internal representation of the query based on user input and available domain knowledge. The Reasoning Module embodies the knowledge of the intermediary and is responsible for the management of the dialogue with the user, planning and replanning the search strategy. The Formalizer Module simply translates the strategy into search system specific commands. The User Model Builder constructs stereotype-based user models, which serve as inputs to the Reasoning module and the Understanding and Dialogue module, so that they can adjust the interface and search strategies to fit the particular user. The History Manager maintains a long term database of user models.

The user models maintained by the system, which were based on the approach described in this thesis (the proposal of which is [Croft 85b]), consist of information about the user such as his education level, experience with information retrieval systems, assessments of knowledge about particular domains, and familiarity with particular databases. No information is kept about the content of previous queries or user-supplied domain knowledge.

The interface of the system is designed to operate in a textual mode without the use of sophisticated graphics. The initiative resides solely with the system, so that the user

simply answers questions proposed by the system. The style of the interaction is adjusted to match the user's stereotype.

The architecture of the system contains some novel features with regard to how the rules are organized. There are three kinds of rules in the system: domain rules, matching rules, and conflict rules. The first kind are used to represent the intermediary's knowledge and are used to perform the functions assigned to the reasoning module. The matching rules are used to activate groups of rules, called *blocks* which contain only rules of one kind, in response to different conditions. The conflict rules change the way conflict resolution is accomplished. They may tell the rule interpreter to choose a rule by its weight, by its number of conditions (most constrained), or by some other criteria. The blocks of rules are aggregated into *classes*, which contain at least one matching block and one or more domain and conflict rule blocks; they are given a mnemonic name to indicate their purpose. Classes are activated in a construct called a "task" which consists of a class, a list of input parameters, and a termination predicate. The parameters and predicate may be empty and, if so, the class will operate on all concepts in the internal problem representation and run until specifically terminated.

The internal problem representation consists of concepts that are part of the query; search logic which is the Boolean combination of concepts; objective which is either precision, recall, or sample; limitations which are date, language, and treatment; output specifications which are format, maximum number of items, ordering, and display mode (e.g., offline print, online display, etc.); and search mode which specifies search on controlled vocabulary or on any field.

2.5.6.2 IOTA

IOTA [Defude 85, Chiarmella 87] has the same goals as IR-NLI. The input is a restricted natural language statement of need that is composed of noun groups connected by

Boolean connectives. The output of the system is documents. The architecture of the system is somewhat different from IR-NLI. Whereas IR-NLI implements all its functions as rules, IOTA is a hybrid system using rules as the control structure for the system. The rule actions implement large pieces of functionality, for example the query parser is one action. The basic control paradigm is a goal-directed approach with limited backtracking. The overall operation of the system follows a set sequence. The interface is textual with the user answering various questions proposed by the system.

Significant points about IOTA are that it has the ability to classify words that are unknown to it by means of inferences based on the context in which the word is used. It attempts to use information about the type of user as input to control decisions of the system. This information is derived solely from the query.

2.5.6.3 CODER

CODER (Composite Document Expert/Extended/Effective Retrieval) [Fox 87] is a system that is currently being developed. Its initial focus was to study how to handle "composite" documents [Fox 86], which are composed of different sections such as the text, a header, and citations. Later, it was expanded to serve as a testbed for studying various approaches to applying AI techniques to IR problems [France 86]. Of all the systems reviewed so far, it bears the most similarity to the system described in this thesis. The kernel of the system is a common data area called a blackboard and a controller called a strategist, which form a blackboard/strategist complex. In the current design of the system there are two blackboard/strategist complexes implemented, one for updating the document base, and the other for handling queries to the system. The blackboard contains a number of areas that maintain information relevant to the current problem, and are divided into two major groupings of subject areas and priority areas. The strategist is where the control of the system resides. It performs a variety of functions that can be described as blackboard

maintenance and control of the experts. The basis for the control decisions has not been detailed. The functions are to determine what experts are relevant to the particular state of the problem, and at a lower level, on what processor can they be executed.

The blackboard serves as communication medium for a community of experts, each of which implements a particular function, for example some of the experts in the current implementation are a p-norm search expert, a p-norm query builder, a time/date handler and an interface manager. To be completed are a browsing expert, a user model builder, and various problem description experts. The experts are implemented in a dialect of Prolog, so that internally they all operate in a goal-directed manner.

The implementation state of the system is currently unknown; no example search sessions have yet been published.

2.5.7 GRUNDY

The program GRUNDY [Rich 79] occupies a somewhat odd position in the analysis of information retrieval systems. The purpose of the program was to experiment with the development and use of user models. The bases of these models are stereotypes, which are representations of typical kinds of users. An individual user model is composed of the stereotypes that apply to the user with a high degree of certainty. The problem domain for the program was book recommendations. That is, the system built a model of the user and compared this model to representations of the books in the database to find ones that had a close match to the user. This description is identical to that of an IR system. Essentially, GRUNDY, by means of a dialogue, built a model of the user according to its indexing language, the stereotypes, and used this model as a "query" against the books that were indexed using the same language. Relevance feedback was performed via the user's review of the books.

What is important about GRUNDY is that it demonstrated how user models could be built, represented and maintained. It did not address how these models could then be used to control the behavior of the system. This has been addressed in the system described in this thesis and to some degree in several of the systems reviewed previously.

2.5.8 Graphical Display of Information

A number of IR systems use a graphical interface to provide a rich visual environment in which to display information and its structure as a means of assisting the user in either expressing an information need, understanding what has been retrieved, or for searching for information. The main intent is to provide the user with sufficient context so that he cannot get "lost" easily. This is especially important when browsing complex structures where the user can easily lose track of where he has been.

An early system that uses a graphical user interface is CALIBAN [Frei 83]. The primary goal of this system is to present system functions in a coherent and easy-to-use way. This is accomplished by structuring the services of the system as a hierarchy. The hierarchies are presented to the user graphically on a high resolution interface. Selection of what system services to use can be made using a mouse. The basic source of information is, as with all the query formulation assistance systems, a thesaurus. However in this system, the user may browse through the tree representation and actually get to documents. Search in this system is based on the concept of Query-By-Example [Zloof 77]. The user fills out a form that matches the structure of the documents, which includes author, publisher, free index terms, classification, and thesaurus terms. The connection between browsing and query formulation is manual. The user must browse the thesaurus while in query mode and select terms that will be added to the query form.

The TOPIC/TOPOGRAPHIC [Thiel 87] system approaches the problem of information presentation by "informational zooming." The TOPIC part of the system deals

with the representation of text meaning. It is a frame based approach with extensions to capture other kinds of relationships. These extensions generalize the representation to a semantic net. TOPOGRAPHIC is the interface part of the system and consists of a multiple window environment that shows various levels of detail. The amount of detail is user controlled by means of selection parameters. In this environment, there are three basic commands: zoom, select, and browse. Zoom means to move to greater or lesser levels of detail; it is not a graphical zoom that makes the information on the screen larger or smaller. Select means to choose attributes to limit the information shown when zooming to a level of greater detail. Browse means to get to some object that is connected to the one currently in view. Each one of these commands is assigned to a button on a three button mouse. For example, the user may be looking at a window that shows a graphic representation of a portion of the frame hierarchy. He can select attributes of a specific frame and have it displayed in another window.

Another approach is the concept of an "information space" [Caplinger 86]. This approach views data as points in an N-dimensional space, the dimensions being an organization of the attributes of the object. For example, some organizations are alphabetical ordering by author or title, another possibility is a classification scheme such as the Library of Congress system. The approach is very similar to the vector space model of the Smart system [Salton 83]. These attributes can be used to construct a space model to show its structure as a two or three dimensional representation on a graphic screen. The sample document base used was a collection of documents from INSPEC, from which four dimensions were derived. These were *classification* which is based on an assigned code, *level* which is based on the source of the document and its intended audience, *treatment* which is based on an assigned value from the list – bibliography, market survey, practical, general, application, new development, theoretical, and experimental, and *length.* of the

document Using these as the dimensions different graphical representations could be constructed. To browse, the user moves his point of view through the space.

2.6 Summary

In this chapter the basic problems of text retrieval and a variety of systems directed at solving those problems have been presented. Most systems have taken one aspect of an IR system as a focus for developing a solution. The most popular approach has been to use domain knowledge as a basis for developing a more well defined query. Only a few systems have taken a system oriented approach and have attempted to tackle more than just a single problem. With this background in mind, the basis for the approach that I^3R takes can be developed.

CHAPTER 3

THE BASIS OF I^3R

3.1 Introduction

In this chapter the requirements for an information retrieval interface are discussed. Emphasis is placed on determining what facilities must be provided to help the user to specify his information need as precisely as possible and to provide the system with the means to respond to the need as effectively as possible.

3.2 Discussion

There are two primary ways of deciding what capabilities an intelligent information retrieval interface should have. The first is a systems analysis perspective in which the basic problem is considered and then ways of ameliorating it are determined. These aids then determine what the functionalities of the system should be, and what additional information is required to support them.

The other way of considering the problem is to look at it from an intermediary standpoint. Since there are people, search intermediaries, who act as the interface between the end user and the retrieval system, we can ask ourselves how they perform their task. What sorts of information do they require in order to get adequate results? Answers to these questions provide the basis for determining how the system should operate.

3.2.1 Systems Analysis Perspective

The basis of I^3R is to provide the user with facilities to overcome the problems of query formulation and to provide the system with multiple ways to respond to the user. These two aspects are intimately related. If the user cannot relate to the system his infor-

mation need in as much detail as he possibly can, then no matter how sophisticated the available search or inference techniques are, they cannot produce the best results. Conversely, if the system does not possess ways of using the information provided by the user in more than just a superficial way, then the time and effort spent by the system and the user to develop a detailed model of the need is wasted. Therefore, effort should be expended to improve both aspects of the system's functionality. Since the information shared by the processes of query formulation and search is a model of the user's information need, called a request model, in an operational sense the purpose of I^3R is to develop the most complete request model possible and to provide ways of using this model to retrieve text that is relevant to the need that it describes.

The basic *process* of retrieval is:

1. build an initial request model,
2. search for and retrieve documents by inference,
3. evaluate the results and revise the request model accordingly.

It is important to view information retrieval as a process rather than as solving a problem. The user may make use of the system many times to obtain information to solve a problem or satisfy whatever information need caused him to come to the system. The time between the second and third steps may be minutes or days depending on how thoroughly the user evaluates the information retrieved. The particular information need may be a current awareness type request, where the user periodically comes back to find new information. Consequently, the system must retain the user's request models, so that he may come back to work on them again if he so desires.

The foundation for the facilities to assist the user is the use of statistical retrieval techniques, which have the following advantages.

1. They are more effective than Boolean query/exact match search systems [Salton 83].

2. They are domain independent.
3. They can be implemented efficiently [Croft 84].
4. They have a sound theoretical basis.

By using these well-proven techniques, the system can be guaranteed to have a reasonable minimum level of performance.

These techniques also provide the first step in overcoming the problem of query form, since they allow the user to specify a query as free text. However, some provision must be made to allow the user to specify important phrases and related words. This should be done in a manner that does not burden the user with a specification language. One way to do this is to use a graphic interface that allows the user to select the phrases that are important and to combine words in specified ways. While some systems have done this, most require that the user augment the query manually by using cut-and-paste operations. The user should be able to select the concepts that are interesting or relevant from whatever source that is available and have them automatically added to the request model. This also means that the system must keep track of all the evaluations that the user makes in the course of the session. There should be, therefore, a module that builds and maintains the request model. This frees the user from having to pay attention to the form of the query and allows him to concentrate on the content.

Another possible method for determining relationships between words in the query is to use natural language processing (NLP) techniques. However, as mentioned previously, these techniques generally require substantial amounts of domain knowledge to be effective. Systems that have attempted to employ these techniques have been used in very specific and limited domains. This limitation negates the domain independence advantage of statistical systems. Two possibilities exist to overcome this drawback. One is to use a specialized sub-language, such as was done in IOTA [Chiaramella 87]. This burdens the user with having to pay close attention to the query form. The other way is to restrict the use of NL techniques to those that appear to be useable without significant amounts of

domain knowledge, such as are used in FASIT [Dillon 83] which is primarily syntactic in nature or ADRENAL [Croft 87], which uses a high level semantic representation for science and technology. This approach appears to be promising, but is at this point, a developing research area.

The developing work in the use of restricted NL techniques raises another consideration for the requirements of I^3R. That is, how can new functionalities be added at such time that they are demonstrated to be feasible. Not only will NL techniques continue to mature, but new search techniques based on different representations may be developed; for example, searches based on ASK representations [Belkin 86]. Consequently, the design of I^3R must be flexible enough to enable the addition of large pieces of new functionality, as well as allowing existing functions to be upgraded.

In order to maintain domain independence, the system is not required to have pre-existing domain knowledge; although this information is extremely valuable in helping the user refine his information need, and is the basis of many of the systems reviewed previously. In many areas there are already well-defined thesauri and classification systems which serve as a ready source of domain knowledge, whereas in others there are virtually no formalized domain knowledge sources. For example, the medical field is very well covered by MeSH thesaurus. Because this is used as the sole indexing reference for the National Library of Medicine's MEDLINE system, it is revised periodically. Along with this, other sources of medical terminology are available. One is the Current Medical Information and Terminology, CMIT, that gives the causes, signs, symptoms, other information on about 4000 diseases [Rada 87]. Another source is the Systematized Nomenclature of Medicine, SNOMED, which contains about 50,000 terms and is used for indexing patient records. Combining these sources into a single source has been investigated [Rada 88].

Computer science, on the other hand, represents a field that is not as well covered by thesauri or other classification systems. Articles submitted to the ACM for publication, generally, have categories assigned from the Computing Reviews classification system. The categories are relatively high level so specific topics are not assigned. Furthermore, it has only been thoroughly revised once in its history. It has been partially expanded [Waltz 85] to cover the field of artificial intelligence more adequately. These classifications are, however, used only for ACM publications. Articles submitted to the IEEE or British Computing Society, for example, are not classified using this system. Therefore, the system will have to rely on the user for specific domain knowledge related to his need, but it also should be able to take advantage of existing domain knowledge when available. It is important for the system to be able to capture, retain, and display domain knowledge.

Domain knowledge collected from the users represents their particular views of a subject. It may be more or less accurate depending on the expertise of the person entering it or the source from which it is obtained. For example, a user may be consulting a reference book on the particular subject as he enters his query. The domain knowledge collected from the users can be used as a system-wide resource to help other users. Since this knowledge can be of varying degrees of sophistication, it should be left up to the user to decide whether or not it is relevant to his information need. However, users of less expertise may require assistance, so the system should offer advice as to what it considers to be related information. Because domain knowledge is such an important resource and is managed differently than the request model a separate domain knowledge manager is required.

Categorization of the domain knowledge as to whether or not it is expert knowledge requires that the system evaluate the user that is interacting with the system. The ease or difficulty of this task is influenced by the operating environment of the system, since this determines the kinds of users that it will confront. In some settings, like a public or un-

dergraduate library, the system will deal with a large population of diverse kinds of users that will use it infrequently. In other settings, like a law office, there may be only a small rather homogeneous community of users who make moderate to heavy use of the system. The kind of information that a user wants can also vary significantly. Some users may only need summary kinds of information such as is found in an encyclopedia. Others may come back to a problem again after using information obtained at a previous session to gain a better understanding of their problem.

Not only are there different system usage patterns depending on the setting, there are different levels of expertise in different areas among the users. Some users may have a great deal of experience with computers: some may have very little. Some users may be very knowledgeable in a specific domain; some might have a general knowledge of many domains. The goals of the search might also be very different. One user may be looking for a specific piece of information; another may be seeking broad background information.

This consideration brings up further questions, such as what can be known about the user and what knowledge is useful. The previous discussion indicates that a number of things can be known about the user. The current request model as well as all previous request models and their associated relevance judgements are available. The domain specified by the user is available. To further help the system, it would be desirable to know the user's expertise in the use of computers and the use of IR systems, so that the interface of the system could be adjusted to suit his abilities. If the user is a novice, guidance could be offered if the user is having trouble. If the user is an expert then the system would be less likely to offer help, but could make more facilities available.

Attempting to evaluate the level of the user's domain knowledge is more difficult. Since there is not any guaranteed base of domain knowledge, there is no standard by which the user's domain knowledge can be measured. Perhaps structural analysis could be made;

looking at the complexity, interconnectedness, or depth of the knowledge to determine its classification.

In any case, it is apparent that a user model builder is required to determine what the models apply and also to maintain information gathered about the user from previous interaction.

Depending on the level of the user's expertise with the system, it may called upon to explain its operation. This requires that the functions of the system be implemented as "transparent" boxes rather than as "black" boxes so that their decision criteria may be examined. If a uniform way of implementing the functions is chosen, then the task of explanation can be accomplished by a single explanation function that "understands" this representation.

Another aspect of the system's operation to meet the varying needs of users is how it searches for information. Previous work [Croft 85] has shown that using more than just a single search method can significantly increase the performance of a system. While it has been demonstrated that, at this time, it is not possible to select the best search method based on attributes of the query, the general behavior of search methods can still be exploited. For example, cluster searches [Van Rijsbergen 79] tend to be useful in precision oriented searches and retrieve different documents than a probabilistic search does [Croft 79]. This leads to the conclusion that the use of multiple search strategies would lead to better results and therefore greater user satisfaction.

Another important technique that can be used to help the user in both refining his information need and in searching for relevant information is browsing, which can be defined as an informal, user-directed heuristic search through a well-connected collection of records in order to find relevant information. As has been pointed out [Bates 86], browsing takes advantage of two well known cognitive abilities. The first is the greater ability to recognize what one wants rather than describing it, and the other is the ability to

skim or perceive at a glance. Instead of the user having to attempt to describe something that he may not have a firm idea of, he can simply pick out what is relevant and let the system keep track of the description.

Incorporation of browsing makes demands on the structure of an IR system. The first is that the organization of the documents has to have significantly more structure and interconnection than is normally found. Concepts already are organized by the domain knowledge structure. This leads to an examination of what organizing principles should be used to generate this structure. One possibility is the use of available classification systems such as the Library of Congress cataloguing system, or more domain specific ones like those previously mentioned, MeSH and Computing Reviews. The choice depends on the breadth and depth of the collection. If the collection is very broad, then the standard classification systems will be appropriate. If the collection is narrow, additional classifications must be made. It is well known that citation information is a valuable source of information with regards to accessing related information and is easily obtained.

The second demand that browsing makes is that the user must be able to view the organizational structure of the information. This need, plus the requirements of request model and domain knowledge management, leads to the requirement for a separate user interface manager that can provide a more sophisticated interface than has been traditionally provided.

The facilities that have been discussed, other than the explainer, do not require the use of natural language generation. Therefore this capability should reside in the only function that needs it. The interface manager can determine where to put the explanations that are generated.

In the discussion so far, the need has been described for the following kinds of functions or modules:

1. Request model construction,

2. Natural language processing,
3. Domain knowledge management,
4. User model construction,
5. Explanation facility,
6. Search selection,
7. Browsing advice,
8. User/Machine interface management.

Each of these modules can operate relatively independently of each other, which means that each one on its own, can determine what it needs to do or what steps to take. However, more than one module at a time may wish to present information to or request information from the user. If this activity is unrestricted, the user could be faced with a chaotic and confusing situation. This can reduce the user's ability to effectively respond, since he has to consider what information is presented or what response he must provide. While the system may be able to operate in parallel or switch tasks easily, the user can really only consider one thing at a time. Therefore, what the system presents to the user should be done in a logical order. This implies that the system needs to have a control function that is responsible for mediating the interaction between the system and the user. This control function can be seen as a dialogue manager that determines what functions can "talk" to the user at any given time.

A system analysis of a more general information provision mechanism was done by Belkin, et. al. [1983]. Its functional decomposition of the information provision mechanism is presented in the next section and is compared to the analysis present in this section in section 3.3.

3.2.2 Intermediary Concept

The preceding discussion has examined the requirements of the problem from a systems analysis perspective. That is, given the nature of the task, what will the system have to do in order to handle in an effective way the problems posed. The same task has been analyzed from an "expert" analysis point of view, examining how search intermediaries perform their task [Brooks 84, Daniels 85].

Recall that a search intermediary is a person who uses the currently available information retrieval systems on behalf of a user to find information. The primary reason for their task is to compensate the shortcomings of traditional retrieval systems. The intermediary, then, becomes the interface to the system for the user. This means that the end user has only to discuss his information need with the intermediary and can ignore the details of system operation.

The method of Belkin, Brooks, and Daniels analysis was to record interviews conducted by a search intermediary with a person seeking information. These interviews were analyzed to determine their structure in order to find out the steps that an intermediary goes through or goals that he must achieve to carry out a search session, and the knowledge required to support this activity. The steps fit into a framework of ten goals or functions that a general information provision mechanism (IPM) [Belkin 83, Belkin 84] performs. This information provision mechanism is a system that is intended to be more general than a text retrieval system. It may incorporate fact retrieval and other kinds of problem solving assistance as well as text retrieval. The information it contains is intended to be very broad, since it is envisioned to be available as a public utility, essentially, an online public library. The ten IPM functions are:

1. **Problem State** — Determine position of the user in the problem treatment process;
2. **Problem Mode** — Determine appropriate mechanism capability

3. **Problem Description** — Generate description of problem structure type, topic, context, etc.;

4. **User Model** — Generate the description of user type, intentions, beliefs, etc.;

5. **Dialogue Mode** — Determine appropriate dialogue mode for immediate situation;

6. **Relevant World Builder** — Choose and apply appropriate retrieval strategies to world model;

7. **Response Generator** — Determine propositional structure of response to user appropriate for immediate situation;

8. **Input Analyst** — Convert input from the user to the mechanism into structures appropriate for the functions;

9. **Output Generator** — Convert mechanism's internal response structures into structures appropriate for the user in the immediate situation;

10. **Explainer** — Describe mechanism operation, restrictions, capabilities, administration to user at appropriate points in the dialogue;

11. **Blackboard Analyst** — Moderates the interaction of the other functions with regard to access to the blackboard.

The primary focus of this work, so far, has been to develop in detail the user model and problem description functions in the context of a text retrieval system.

While many other systems have embraced the intermediary concept, their foundation has been either published sources [Bates 79a, Bates 79b], or analysis of a single intermediary. The work by Belkin, et. al. represents the most comprehensive and detailed empirical studies to date.

One of the major results of this work has been the examination of the dialogue structure. Their analysis shows that the dialogue shifts in and out of different areas of focus in a loosely structured fashion, and does not proceed in a fixed step-by-step manner. This dialogue can be described as following a set of goals that are further divided into subgoals. The course of a session is driven by the intermediary attempting to fulfill these goals. The dynamic nature of the dialogue can be partially explained by the intermediary initially considering a goal to be satisfied and then realizing that the topic represented by a

goal needs more refinement. The implication of this work for I^3R is that the control structure of the system must be extremely flexible.

This work is complementary to the specification and design of I^3R. It is interesting to see that work directed at a very similar problem, but derived from a different point of view produces a design that is very similar to that of I^3R. The functional correspondence of I^3R and the IPM is shown in figure 3.1.

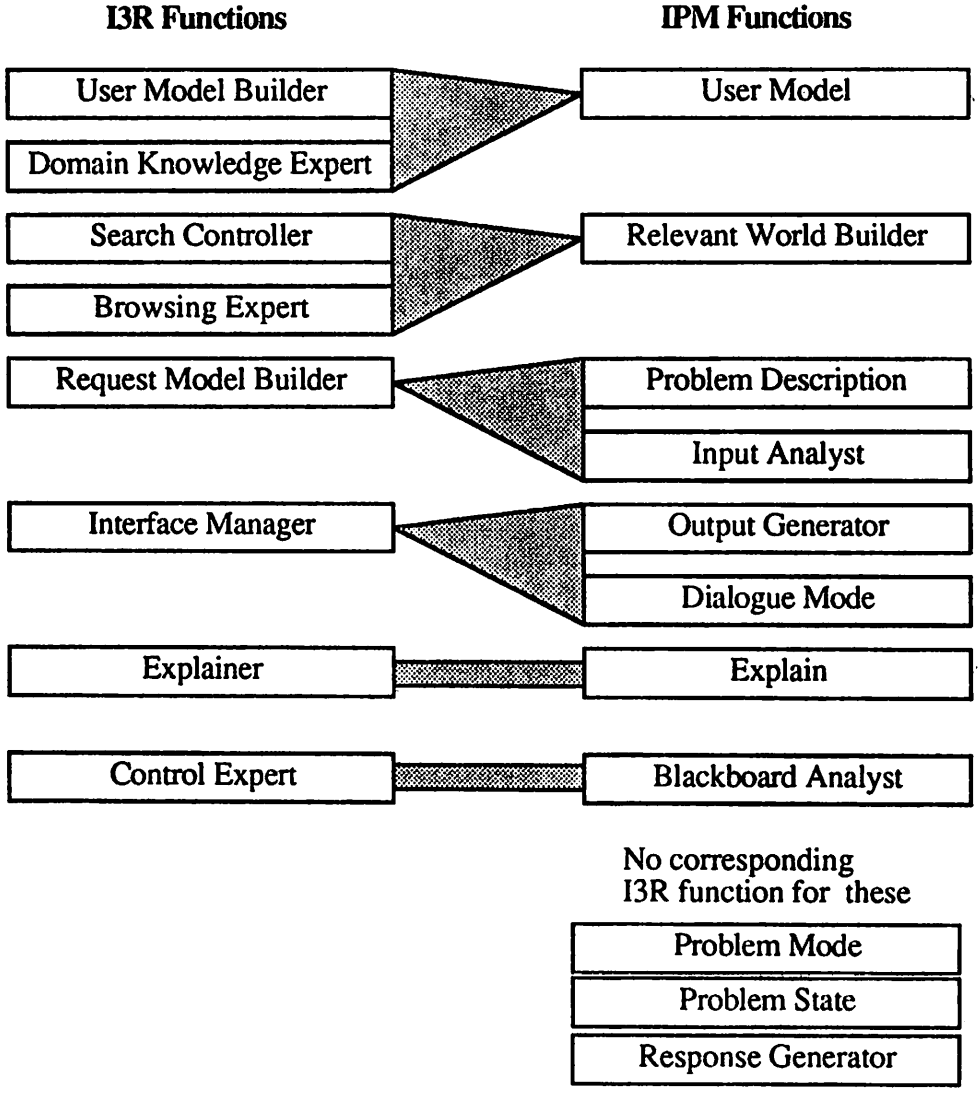

Figure 3.1: I^3R / IPM functional correspondence.

3.3 Conclusion

In this chapter, the functions necessary for an intelligent information retrieval interface have been described. Similar functions to these have also been derived by others from the perspective of simulating the behavior of a search intermediary. This similarity leads to the conclusion that the approach embodied in I^3R is sound. I^3R provides additional capabilities that a human intermediary does not, particularly browsing and a graphic oriented interface.

CHAPTER 4

DESIGN AND IMPLEMENTATION

4.1 Introduction

In this chapter the issues of design and implementation are discussed. First, the overall structure of I^3R is presented and is contrasted to the architecture of the Hearsay II speech understanding system from which it was derived in order to highlight the design decisions. Second, the implementation of the system architecture is presented in detail. Third the contents of the long term memory is described. Finally, the implementation of the individual experts, with the exception of the browsing expert, is discussed.

4.2 Design

4.2.1 Requirements

The requirements for an intelligent interface for information retrieval, developed in the previous chapter, can be summarized as follows:

1. Independently operating functions are required, that in the process of solving their own problems, contribute to the process of finding documents relevant to the user's expressed information need. Seven functions have been identified as necessary.

2. A separate control function is required to mediate the flow of information or dialogue between the experts and the user.

3. The system must be able to integrate new functions into its operation as they are developed.

4. Individual functions are required to be alterable, so that they can be changed as new developments occur or as tuning is required.

5. Decision criteria of each function is required to be available for use by an explanation facility.

6. A sophisticated graphical and textual interface is required to allow the user to select interesting elements such as documents and concepts, as well as, allowing the user to input information by the keyboard.

4.2.2 Architectures

A variety of architectures were considered to meet the requirements of an intelligent interface. Those considered were AI architectures, since many of the problems being examined required the use of AI techniques and representations for solution.

The architectures considered were traditional rule systems, Prolog architectures, and blackboard systems. Rule systems were considered since they meet the requirement that their decision criteria are available for use by an explanation facility, and they meet the requirement that they can be incrementally developed (requirements 1 and 4 are variations on this theme, differing in the granularity of the alteration).

4.2.2.1 Traditional Rule Systems

Traditional rule systems meet many of the system requirements, especially allowing incremental changes to the functionality of the system, and making the system's decision criteria available to an explanation mechanism.

A rule system consists of a body of rules and a working memory. The rules are composed of two parts: a condition part and an action part. These are also referred to as the antecedent and the consequent, respectively. The condition part of a rule is composed of one or more individual predicates that are implicitly ANDed together. For a rule to be eligible for execution, all of the predicates must be true. The action part is composed of actions that alter the working memory. The working memory consists of information that represents the state of the problem being solved, as well as control information. How the working memory is organized and the information representation is organized is dependent on the particular implementation.

The basic operation of a rule system is as follows.

1. All rules are examined to determine if any have a condition part that is true. If not the system stops.
2. All those rules whose condition part is true form the conflict set.
3. Using some criteria one rule is selected to "fire" or execute its action part.
4. The selected rule's action part is executed
5. Go to step 1.

The difficulty with using these kinds of systems stems from the way that control of the system is implemented. This is usually done in the way that the rule for execution is selected from the conflict set. The following are some of the ways that this is done.

1. User assigned priority values.
2. Number of conditions in the condition part.
3. Physical ordering of the rules. The rule that occurs first in a file of rules is selected first.

These methods are acceptable when the system in not too large (< 150 rules). If it gets beyond that size, it requires a great deal of effort on the part of the developer to make sure that the system operates as expected.

One way to assist in organizing large rule systems is to partition them. Partitioned rule systems are a variation of the traditional rule system. The primary difference is that the rules are partitioned into sets that are activated and deactivated by a control in their condition part. When the control condition is matched, the entire set of rules is enabled and eligible for selection; when it is false the set is disabled. The control condition, in form, is no different than any of the condition. It is its purpose that is different.

The utility of this kind of system is that it provides a way to develop groups of rules that respond to major changes in the situation. The recognition of the change can be done by rules that are in essence control rules. The actions of these rules assert or retract the information that controls the other sets of rules. For example, one way of organizing the

rules into different functions is to partition the rules into sets by adding an extra condition to the condition part. When the condition is true the set of rules defined by that condition is enabled. The sets of rules can be controlled by rules that are devoted to manipulating the phase information. This kind of architecture in a more complicated form is how the IR-NLI [Guida 1984] system is implemented.

While it appears to be easy to add new large scale functions by simply adding a rule set that implements it and modifying the control rules, there are other difficulties to consider. These stem from the ways that rule systems are implemented to avoid the continual firing of rules where condition parts remain true through a number of cycles. The behavior that is desired in the implementation of a rule system is for a rule to respond to a set of conditions only once when that set becomes true and not respond to that exact specific set of conditions. If the conditions change and then become true once more, the rule would then fire again. Essentially, a rule should respond the same way that a Petri net transition [Peterson 78] works. In rule systems that are built using the RETE pattern net [Forgy 82], this is accomplished by maintaining markings in the net of the conditions that are true for a given rule. When the rule fires, the markings of those conditions for that rule are removed. These markings are usually in reference to a numbered "fact," which is a tuple in the working memory. This means that a rule will respond only once for a unique set of facts. However, the partition information is also kept as a fact in working memory that gets retracted (removed) when the partition is deactivated and reasserted as a fact with a new number when the partition is reactivated. This means that many rules that have fired previously, will fire again because they are responding to a new set of facts. The only difference in such cases is the new control fact. There are ways to program around this, but they require that more control information has to be embedded into the rules and the representation of information in the working memory.

4.2.2.2 Prolog architectures

Prolog was considered as a candidate implementation model for the system. Prolog implements a particular kind of rule system. It is a goal oriented top-down system. The "rules" are specified in terms of goals and subgoals. For example, (The following syntax is not true Prolog, but is meant to be illustrative.) an abstract rule would be:

```
<goal 1> <- <subgoal 1> <subgoal 2> <subgoal 3>.
```

This specifies that goal 1 is satisfied or achieved by subgoals 1, 2, & 3 being satisfied. A Prolog program consists of rules of this sort. Processing is started by trying to establish a goal that is on the left hand side of the "<-" symbol in a rule. The system then tries to satisfy this goal by satisfying the subgoals, and the subgoals' subgoals, etc.

Prolog is a powerful language, and is very convenient for solving problems that have or can be cast into a goal oriented structure, such as parsing language. It is however very difficult to implement multiple independent lines of reasoning in a single Prolog system. Prolog is also very restrictive in terms of control, since there is only one control paradigm, goal-directed processing with backtracking. While this control paradigm may be suitable for certain classes of problems, it is unsuitable for many others. It has been used as the implementation language for individual components in the CODER system (which is similar in structure to I^3R)[Fox 87] in a distributed processing environment where each expert runs as a separate process and communicates by means of message passing.

4.2.2.3 Blackboard systems

The architectural model that best fulfills the requirements is a blackboard-like architecture [Erman 80, Nii 86a, Nii 86b]. This architecture can be defined as a group of independently operating knowledge sources (KSs) that have access to a global data structure called a blackboard (BB). The blackboard is structured as a series of levels that repre-

sent abstraction levels of the problem that the system, as a whole, solves. The horizontal dimension of the Hearsay II blackboard is time; in other systems the blackboard may model spatial information as well. Figure 4.1 shows the high level architecture of Hearsay II.

A hypothesis is an individual KS's interpretation of the information posted on the blackboard. The direction of inference can be bottom-up, if the interpretation is from a lower to a higher abstraction level; top-down, if the interpretation is from a higher to lower abstraction level; or laterally, if the interpretation is within the same abstraction level.

A knowledge source is a separate function that is completely self-contained. The only requirements placed on the KS are that it have a precondition program and that its input and output conform to the hypothesis representation of the blackboard. The precondition program is similar to the condition part of a rule in a rule based system. It differs from a rule condition part in that instead of evaluating to true or false, it produces an estimate of the contribution that running the KS will make.

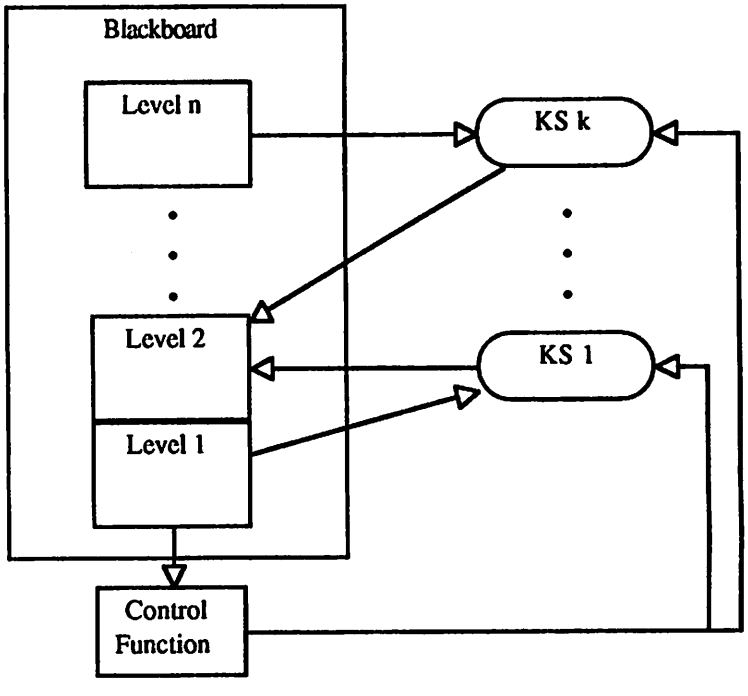

Figure 4.1: Hearsay II high level architecture.

This estimate is then used by the control function in determining what KS to run. A knowledge source's internal workings are hidden from view of any other knowledge source. Furthermore, each KS, generally, has all of the information it needs to make its interpretation.

A blackboard system functions in the following way. A change is made on the blackboard by the posting of a new hypothesis. The blackboard monitor informs those KSs that are potentially interested in the change that it has occurred. The interested KSs place a pre-condition program in the scheduling queue. The scheduler orders the queue by priority based on the focus-of-control database and selects the top action to perform. If the action is a precondition program, it is executed and the KS places an instantiation of an action program in the queue. If the action is an action program, also called a KS instantiation, it is executed which results in the posting of a new hypothesis on the blackboard, and the cycle starts again. The system stops when it has found a solution to the problem. Recognition of a problem solution is made by a KS that functions at the highest level of abstraction.

Control decisions embodied in the ordering of the execution queue, also called the agenda, are primarily based on attempting to minimize the time needed to solve the problem, which, in the Hearsay II case, was to produce a high level interpretation that accounted for all of the input. Because the system was data-directed, the potential for a multitude of possible interpretations was great, especially at the intermediate levels. Since the sequence of operations is not of particular importance, the scheduler can decide to process KS instantiations in any order it chooses, usually working from well established parts of the solution, in a process called "island driving."

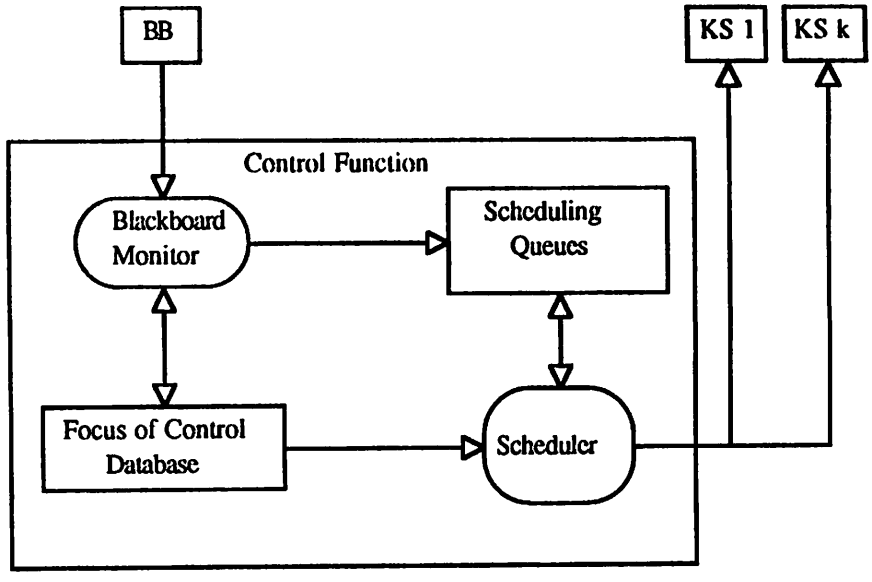

Figure 4.2: Hearsay II control function.

4.2.3 Sufficiency of Blackboard Architectures

The blackboard architecture meets the previously specified requirements in the following ways. First, the blackboard architecture allows easy integration of new large scale functions into the system. The only requirements are that the new KS conform to the blackboard interface. This means that when the KS alters the blackboard, the blackboard monitor receives sufficient information so that it can inform any other interested KSs of the change. It also means that the new KS must provide the blackboard monitor with sufficient information to indicate the changes in the blackboard that interest the new KS. Accordingly, separate knowledge sources fulfills requirement one and partially fulfills requirement three (section 4.2.1).

Secondly, the blackboard architecture separates the control of the overall system from the KSs. This means that each KS can concentrate on solving the problem for which it is designed, without having to consider the overall perspective of the problem. The overall perspective is localized in the control function. By doing this the control of the system can be easily altered and can be easily explained. If the control of the system were

dispersed into the individual KSs, integration of a new KS would be very difficult, since knowledge of that new KS would have to be put into all of the existing KSs. The separate control function fulfils requirement two and partially fulfills requirement three.

Since the KSs are independent, they can be adjusted or altered so long as the adjustment or alteration does not grossly affect its overall function. This means that they can be refined to better meet the needs of the system as the system and the KSs operation is observed over a period of time. This meets requirement four.

Requirement five is best met by a rule based system. It can be met by a blackboard system depending on its implementation. The original Hearsay II system was implemented in an Algol-like language called SAIL (Stanford Artificial Intelligence Language) and so the operation of the individual knowledge sources would be difficult to explicate. If each the knowledge source was implemented as a rule system, its decision criteria would then be explicitly available to an explanation mechanism. This also has an added benefit of making the knowledge sources easier to alter.

Requirement six can also be met by a blackboard system depending on its actual implementation. There is no specified way that a blackboard system must be implemented other than what has been described. The graphical interface manager could be integrated as another of the KSs, but this would interfere with its operation. Its purpose requires it to respond to the user as rapidly as possible. Consequently, it needs to run independently of the control function and the other knowledge sources. What it displays can be determined by the messages that it receives from the rest of the system.

4.2.4 High Level Design of I^3R

Since the blackboard architecture can meet all of the requirements of an intelligent information retrieval interface, it was chosen as the basis for the design of I^3R. There are,

however, a variety of differences between the kind of problems that the blackboard architecture was originally designed to solve and the information retrieval process.

The kind of problems that blackboard architectures are primarily designed to solve can be represented by multiple abstraction levels, such as is the case with the original Hearsay II. Other kinds of problems include battlefield situation assessment, where the purpose is to identify threat forces and predict their likely actions. The abstraction level in a military situation might be, at the bottom, various kinds of electronic emissions in different parts of the spectrum such as radar or microwave. These correlate to different kinds of units. These units are part of larger kinds of units and so on until the structure of an entire battlefield is represented.

The information retrieval process has no overall hierarchical structure, even though there are hierarchies in the representations of some of the underlying information. Because of this, and other differences some changes are needed in some of the design elements of the system. Figure 4.3 shows the high level architecture.

The first change is in the central shared data structure. Instead of being organized as a hierarchy levels, it is a heterogeneous structure composed of different "places." Consequently, it is referred to as a short term memory (STM) rather than a blackboard. These "places" are where the various knowledge sources build models that contain the information that they have derived from the session. Each model is in a form that is most appropriate for the particular function that the knowledge source serves. For example, the request model builder (RMB) uses a hash table of record structures that represent terms to model the user's request. The domain knowledge expert (DKE) uses modified semantic nets for the user's and the global domain knowledge. Each model will be discussed in the context of the expert that builds it.

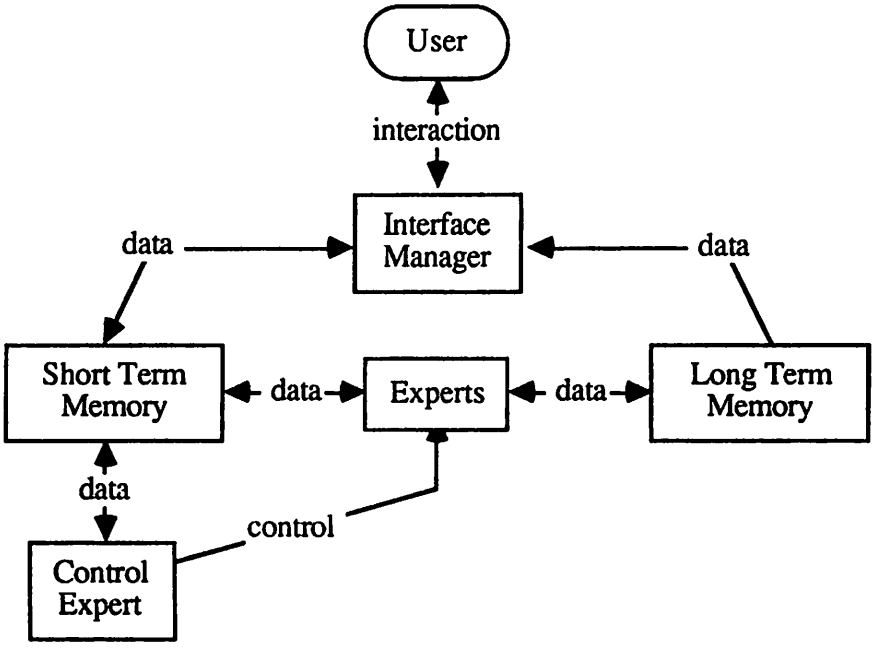

Figure 4.3: High level I^3R design.

In Hearsay II each knowledge source had its own knowledge resources that were not shared with other knowledge sources. In I^3R, a significant amount of knowledge is shared among the various experts. Much of this knowledge is permanent and is kept in a long term memory (LTM). It is altered by the experts, and is available for use by the interface manager. It consists of the document collection, the global domain knowledge, and the user histories. The long term memory will be described in greater detail in the next section.

The control expert determines what experts can operate during any time in the session. Its purpose is to coordinate the activity of all of the experts so that the dialogue is consistent and logical. The basis for its decisions is a high level plan of the end-user/intermediary dialogue.

The interface manager handles the control of the user interface. Its basic functions are to manage the graphic presentation of information to the user and obtain user input.

The experts correspond to the functions specified in the requirements developed in the previous chapter. Each one is implemented as a rule based system, and will be described fully in subsequent sections.

4.2.5 Long Term Memory

As stated previously, the long term memory (LTM) is composed of three major parts, the document collection, the global domain knowledge and the user histories. Conceptually, the document collection and the global and user domain knowledge are considered to be fused into one structure, called the concept/document knowledge, since this is how the browsing expert presents information to the user. Different experts make use of different parts of this knowledge. The domain knowledge expert is most concerned with conceptual structures, the search controller is most concerned with the document knowledge, and the browsing expert makes use of all the concept/document knowledge.

4.2.5.1 Domain Knowledge / Document Collection

The information in the concept/document knowledge is organized in three levels, the concept level, the document level, and the journal issue level. This is shown in figure 4.4. This figure does not show all of the links that exist in the knowledge, but shows the hierarchical organization of the levels; groups of concepts form a document; groups of documents form a journal issue. The hexagon symbol at the top labeled "J1" represents a particular journal, such as CACM, or IEEE Computer. It is included to indicate that the concept/document knowledge hierarchy could be extended. The organization of the concept/document knowledge in this hierarchical and combined way is one of the original contributions of I^3R. Other systems kept these sources of information separate. The primary reason for combining the concept/document in this way is to facilitate browsing. It

also provides a convenient structure to support multiple search strategies and the acquisition of domain knowledge.

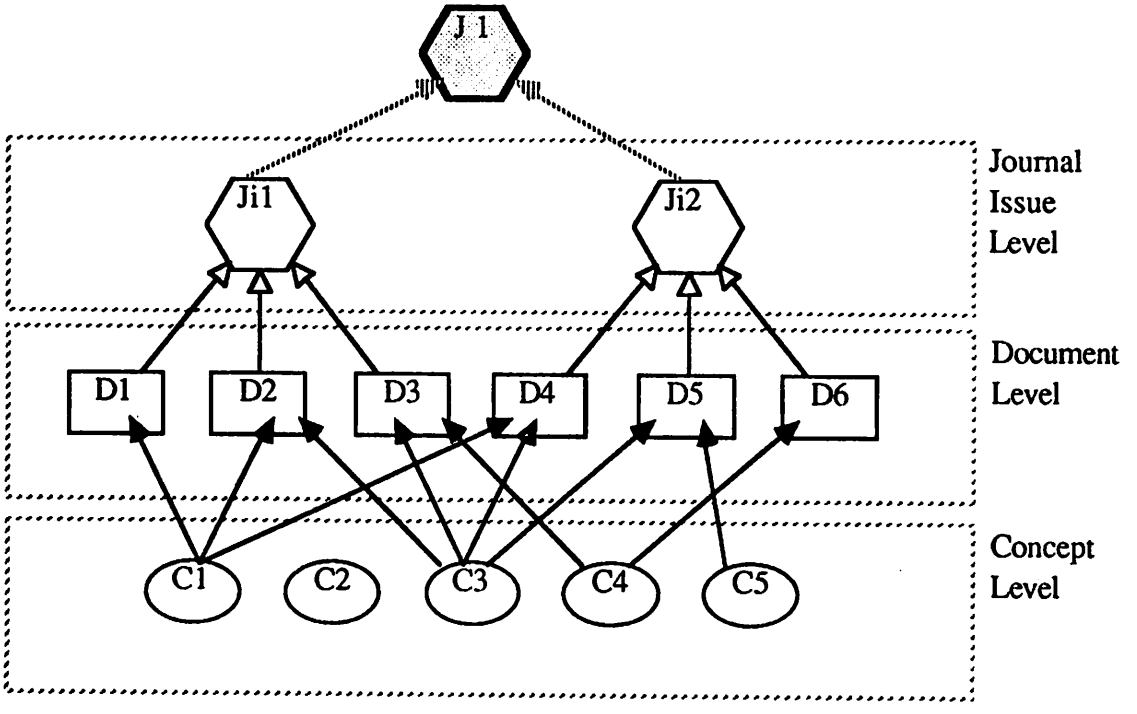

Figure 4.4: Basic structure of document collection, showing the relationships between the levels. A concept (ellipse) can be in many documents (squares) and a document can be in only one (generally) journal issue (hexagon).

4.2.5.1.1 Concept level

The concept level is the lowest level, since concepts are the fundamental indicators of meaning. In attempting to form a query, an end user selects concepts that he feels will best represent his information need. Due to the fact that automatic indexing is used, only single word concepts are included in document representations. Authors are included at this level. An argument can be made that authors represent a different kind of information than concepts, but, in practice, end users search using authors in the same way that they use terms, so they are included with the rest of the single word concepts.

The organization of the concepts constitutes what is normally called a thesaurus in other text retrieval systems. It is a semantic net with a variety of link types, including *synonym, related, broader, narrower, component, part-of, phrase,* and *nearest neighbor.* The synonym link connects concepts that are identical in their usage in a given domain. When concepts are not close enough in meaning to be called synonyms, but are still allied in their usage they can be connected with a related link. Broader and narrower links represent the standard generalization hierarchy. For example, mammal is broader than rodent. Component and part-of links represent the standard aggregation hierarchy. Phrase links connect multiword concepts, like "Artificial Intelligence," to their single word constituents. Nearest neighbor links connect single word concepts (also often referred to as terms) that are statistically similar. This relationship was discussed in detail in chapter two. These links are divided into two types, nearest neighbor from, and nearest neighbor to, since the relationship is not necessarily symmetric. For example, term A's nearest neighbor might be term B, but term B's nearest neighbors are terms C and D.

In the Rubric system [Tong 83] certainty factors are attached to the links between concepts to provide a basis for document ranking. This is because Rubric uses the structure of its domain knowledge directly to infer the relevance of a document. In I^3R the domain knowledge structure is not used directly as a criterion for retrieval, therefore, no certainty factors are attached to the links. There are two reasons for this. First, for many of the link types, a numeric strength does not make sense. For example, it is not particularly meaningful to say that concept A is narrower than concept B with a certainty of 0.8. Either concept A is narrower than B or it is not. What determines this is either the domain in which the two concepts are used or the opinion of the person structuring the knowledge. If the exact nature of the relationship cannot be determined, the "related" relationship can be used. Secondly, the way that the domain knowledge expert searches for concepts to recommend does not require certainty factors. This method will be

described in a subsequent section on the domain knowledge expert. Figure 4.5 shows an example of how a portion of the conceptual knowledge might be structured. The node labeled "document collection" indicates that the attached single word concepts are components of the document representatives.

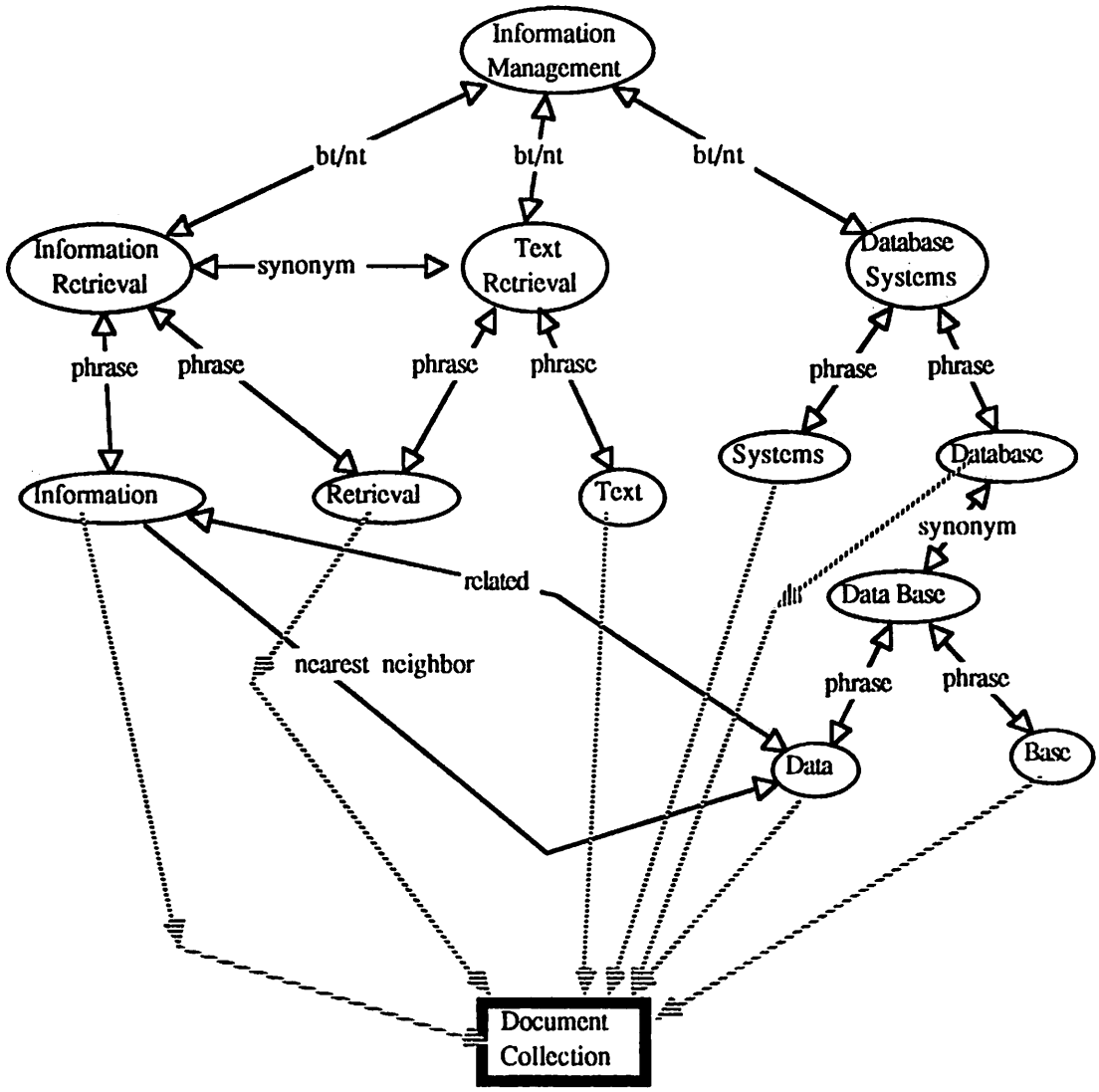

Figure 4.5: Sample Conceptual structure.

The concept level, during a session, is a combination of three knowledge bases, the global domain knowledge, the user's domain knowledge, and the text database. The global domain knowledge is derived from published thesauri or other classification information.

In the case of our test data, for example, it was derived from the old and new Computing Reviews classifications. The user's domain knowledge is gathered from him as he interacts with the system. When a user is developing his query initially or is examining retrieved text, he may make connections between words. For example, a user examining a document may indicate that the words "concurrent" and "processes" form the concept "concurrent processing." This phrase is entered in to his domain knowledge model by the Domain Knowledge Expert. He may then think of the concept "parallel processes," and decide that it is not close enough to be a synonym, but is certainly related. Figures 4.6 through 4.9 show the user selecting these phrases and connecting them with the "related" link.

The information from the text database simply gives the mapping from terms to the documents in which they occur. These links are represented by the dotted links in figure 4.5.

4.2.5.1.2 Implementation of the Concept Level

The semantic net representation of the concept level is implemented by means of a hash table of record structures. Hash tables are a convenient structure in Common Lisp. They are expanded in size automatically when they reach a specified percentage of their total capacity, the equality test used to determine if a hit has been made can be user specified, and there is a special function, `maphash`, for iterating through all the values in a table. The definition of record structure used for storing each entry is:

```
(Defstruct  (Concept
              (:Include Common-Content-Info)
                ;; the above is used for DK entry
              (:Conc-Name nil)
              (:Predicate Concept?))
```

```
Id        ;either a term number or if a phrase a list
          ;of term numbers
Stem      ;either a stem or a list of stems
Text      ;full text of the concept
Synonym   ;list of synonyms
Related   ;list of related words
Broader   ;list of broader terms
Narrower  ;list of narrower terms
Phrase    ;text of single words that make up phrase or
          ;list of phrases that this word is in)
```

Each concept is stored using its full text as the key to the hash table. So, when the domain knowledge expert wants to find all of the phrases that a single word concept is a part of, it executes the function `Get-DK` which is defined as follows:

```
(Defun Get-DK (Connection Word Table)
       (Apply Connection (Gethash Word Hash-Table)))
```

where `Connection` is the particular link, `Word` is the concept of interest and `Table` is either the user's domain knowledge or the global domain knowledge. The `id` field provides the connection into the document representations, which are stored in VMS/RMS files.

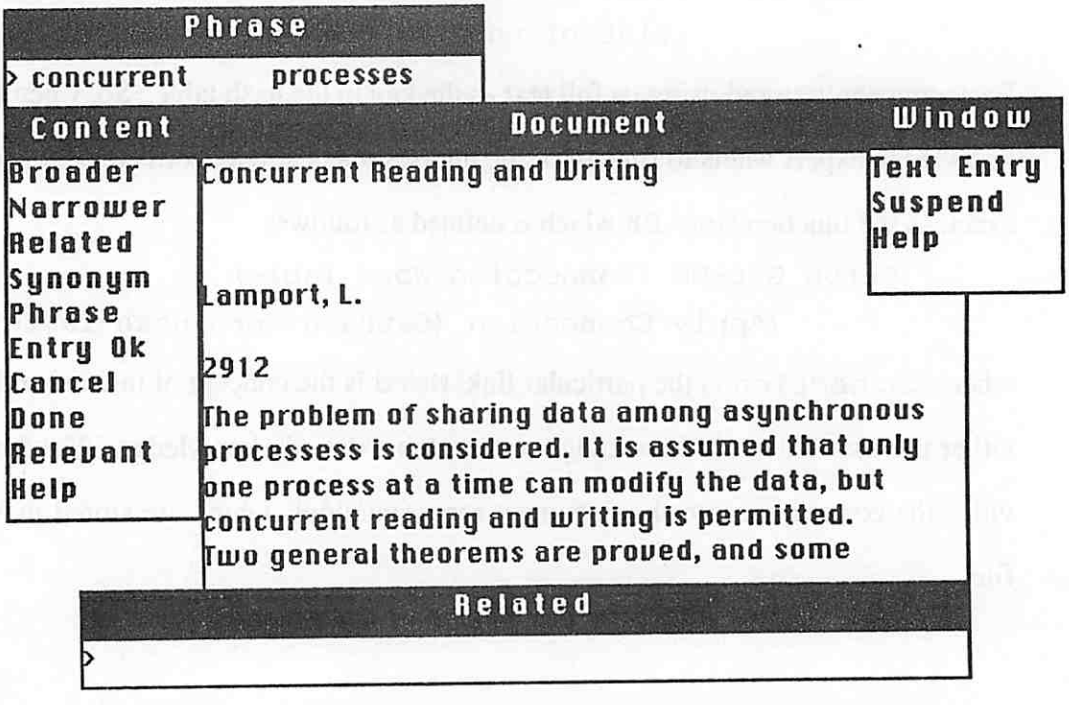

Figure 4.6: The user is making a connection between "concurrent processes" and "parallel processes." The first step is to select **Related** from the Content menu. This causes the related window to appear. The user then selects **Phrase** from the Content menu, causing the Phrase window to appear above the document. Selecting the words "concurrent" and "processes" using the mouse causes them to appear in the Phrase window.

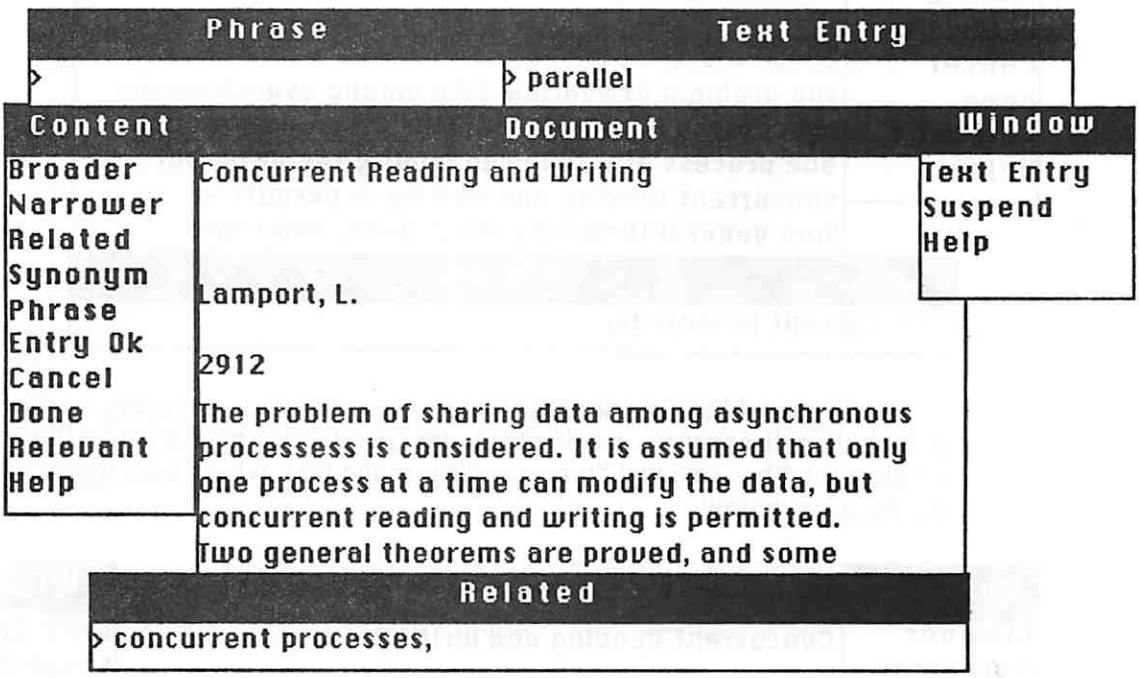

Figure 4.7: After selecting **`Entry OK`** from the Content menu, the phrase "concurrent processes" is transferred to the Related Window. The user then selects **`Phrase`** again from the Content menu and then **`Text Entry`** from the Window menu to allow him to enter the word "parallel."

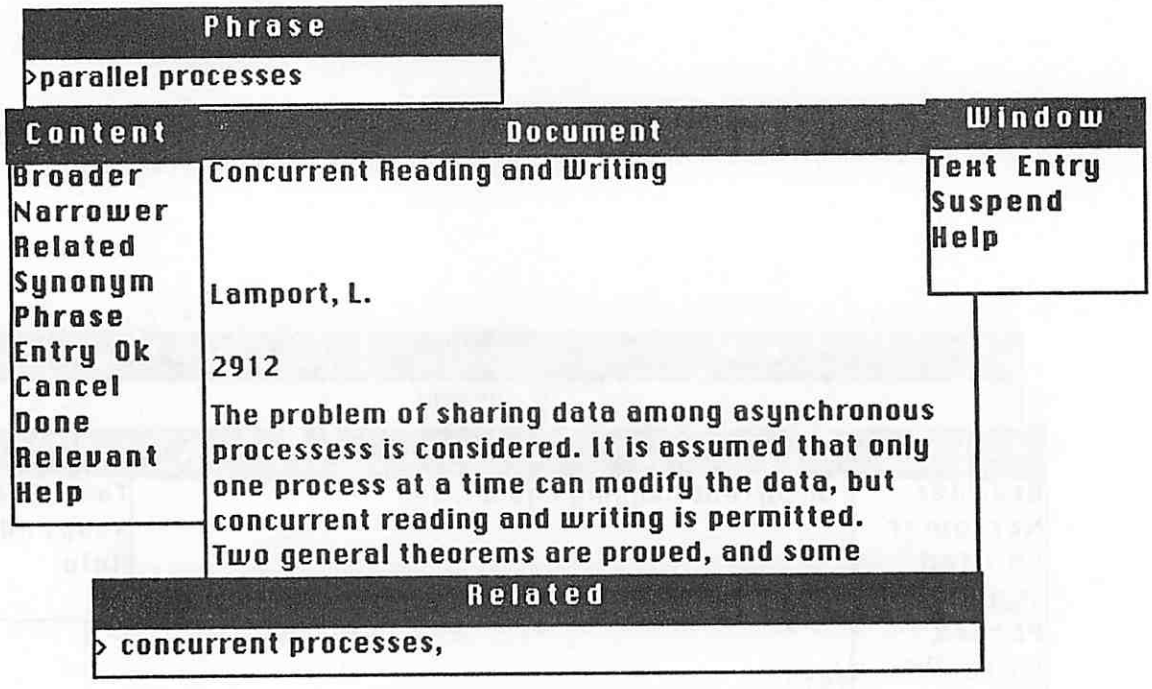

Figure 4.8: The user has keyed `return` in the Text Entry window (which then disappears), causing the word "parallel" to appear in the Phrase window, and has selected "processes" from the text, which also appears in the Phrase window.

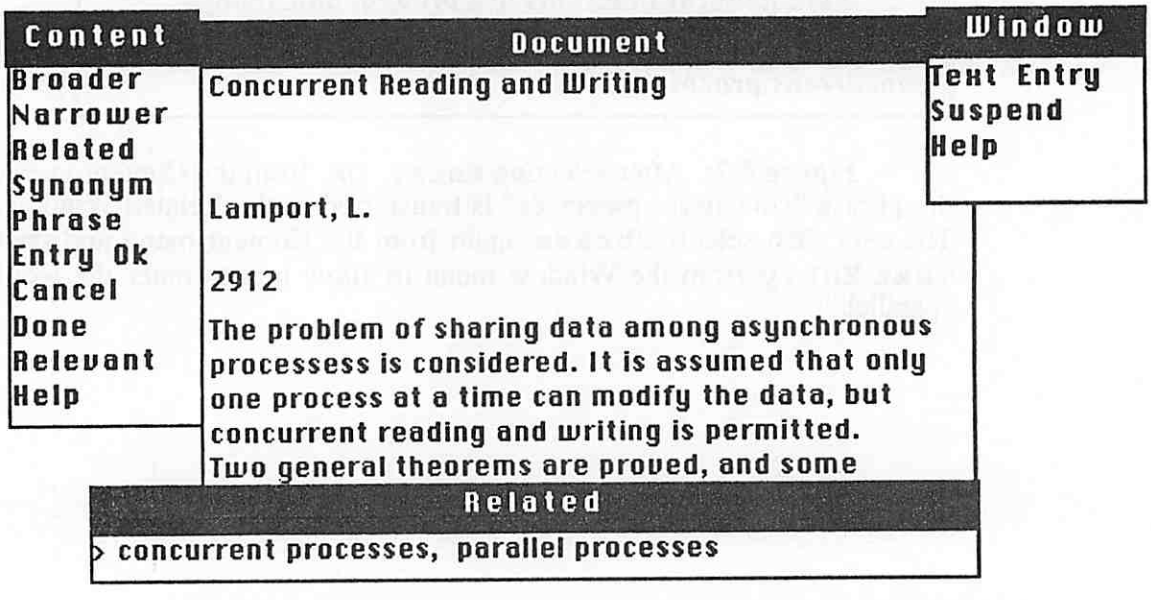

Figure 4.9: The user selects **Entry OK** from the Content menu, which causes the phrase to be transferred to the Related window. When the user selects **Entry OK** again, the Related window disappears and the domain knowledge is entered into the user's domain knowledge model.

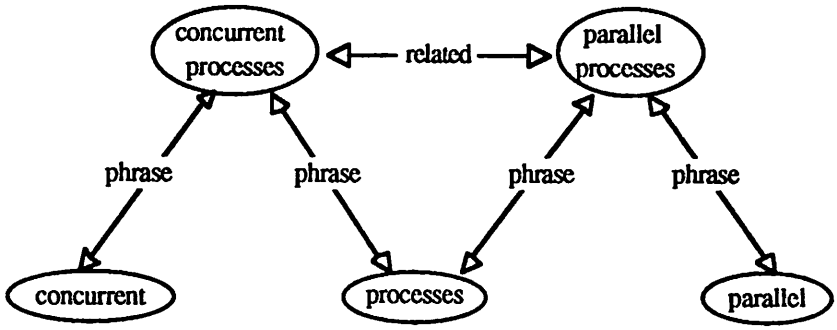

Figure 4.10: Representation of the domain knowledge added to the user's model

4.2.5.1.3 Document Level

The next level is the document level. In this system, documents are the fundamental units of information. It is in a document that the user will find the desired information. Each document is represented by a collection of concepts that indicate its content. These are generally derived from the title, abstract, and whatever keywords are supplied by the author. Because of the automatic indexing method [Porter 80] used to derive the document representations, they consist only of single word concepts (terms). The document level is, like the concept level, highly interconnected. Figure 4.11 shows the kinds of links that can be found. A document nearest neighbor link, which is similar to the concept nearest neighbor link, is used. It is based on the overlap of the terms contained in representations of two documents. Only the most highly related documents are linked by the nearest neighbor link. The algorithm used for generating these links was described in chapter two.

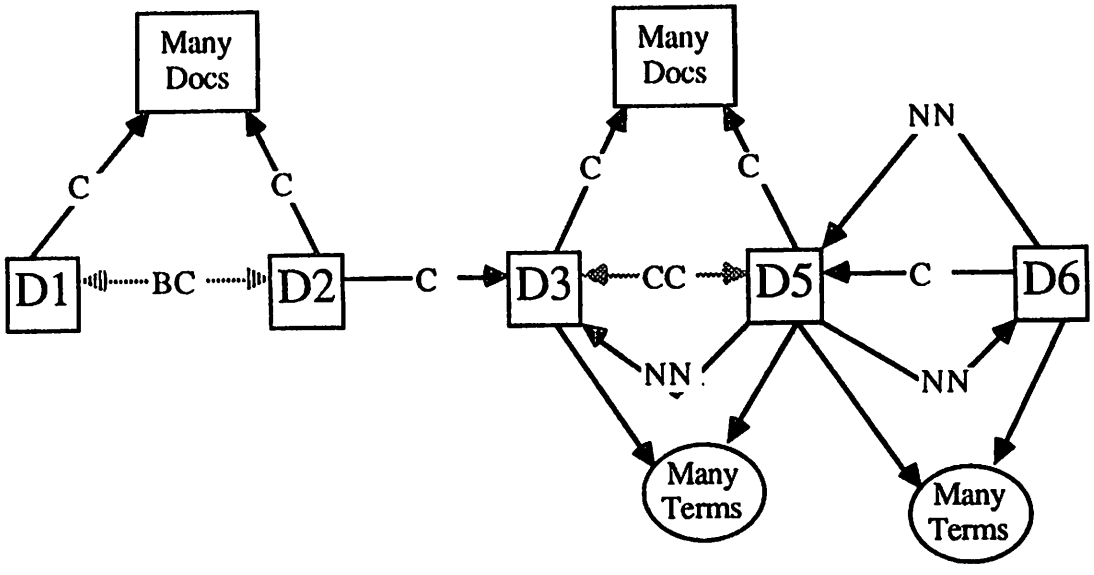

Figure 4.11: The Document level. The markings on the links are as follows: BC = Bibliographic Coupling, CC = Cocitation, NN = Nearest Neighbor, C = Citation.

Each document also contains citation links, which connect it to other documents; these are important since they represent author judgements of what other documents are related to the contents of their documents. They can be used directly to facilitate finding other documents, or they can be used to generate two other kinds of document-document similarity links called bibliographic coupling (BC) links and co-citation (CC) links [Salton 83, Mansur 80]. The first is based on the overlap of two documents' reference lists, and the second is based on the number of times that two documents appear together in reference lists of other documents. Due to some deficiencies in the test collection that is currently used by I^3R, the BC and CC links are not generated. Specifically, the citation information available for each document only references other documents in the collection. Consequently, accurate determination of bibliographic coupling links could not be made, since much many references were missing.

Even without bibliographic coupling and cocitation links, the document level even in the collection used is still highly interconnected. Figure 4.12 shows a region of the test collection, containing documents that deal with tree data structures. In a few cases, the

nearest neighbor links link the same documents as the citation links do. In others, they link documents whose connection by citations is more circuitous, or is not made at all.

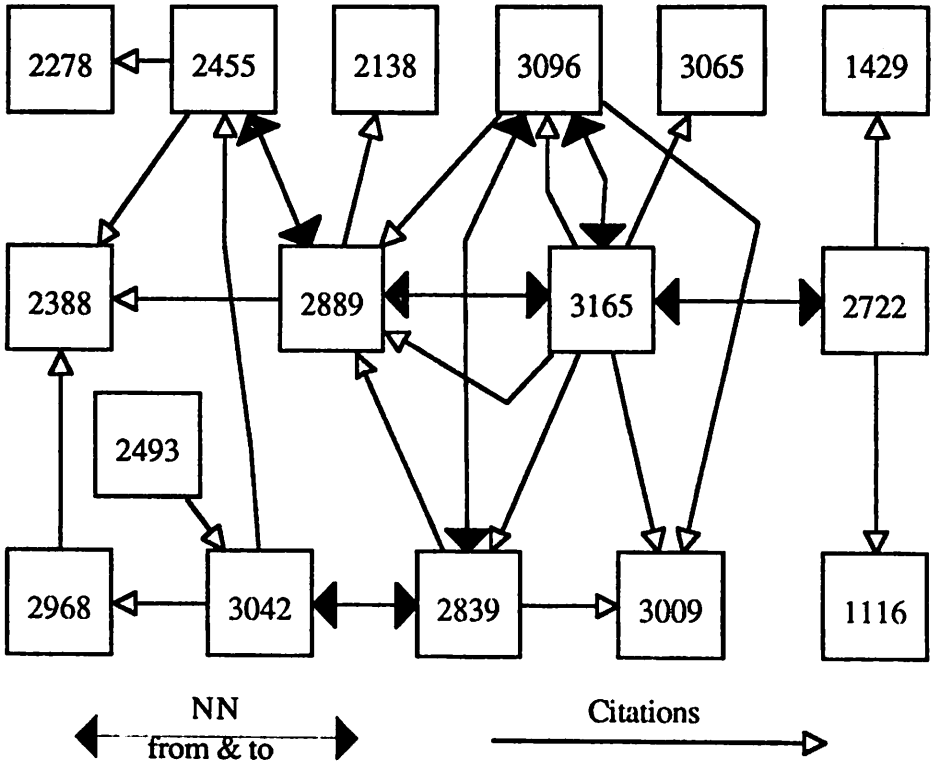

Figure 4.12: Document neighborhood taken from the CACM collection. This neighborhood will be used in the fourth scenario in chapter six.

4.2.5.1.4 Implementation of the Document level

The document level is implemented using VAX/RMS file structures. There are 9 files which are:

1. DOC_TERM – holds the basic document representations,
2. DOC_DOC_CITATION – holds the citation links,
3. DOC_DOC – holds nearest neighbor links,
4. DOC_TEXT – holds the full text of the document as well as the date,
5. DOC_AUTHOR – maps documents to authors,

6. TERM_DOC – hold the inverted file of #1,
7. TERM_TERM – holds term nearest neighbors,
8. TERM_PHRASE – maps the text of terms to the term numbers,
9. COLLECTION_INFO – holds information about the size of the collection and the most frequently occurring term.

These files are accessed directly by the search program. The rest of the system uses access programs that are written in C that can be called form Lisp. Originally, these files were stored as relations in a relational database, VAX/RDB. This database system, however, was very inefficient, causing the searches to take about five minutes. Searches in the current system take about 50 seconds.

4.2.5.1.5 Journal Issue Level and Above

The journal issue level is based on the observation that many journals will have from time to time whole issues devoted to a specific topic. These issues might be the proceedings from a particular conference, or tutorial articles on a specific area. Journal issues may also have sections which are topic specific. The primary motivation for including this level is to support browsing. A typical browsing heuristic is: "If this document is interesting, then are there any more documents in this issue that are interesting."

There are other possible connections above the journal issue level. The most obvious in a collection that has more than one source of information is connecting the journal issues together into journals. The journals can then be categorized by higher level classification structures like the Library of Congress system.

In the current system, the journal issues are not represented by a separate file structure. This information is available from the DOC_TEXT file which has fields representing the month and year that the document appeared.

4.2.5.1.6 The Test Collection

The test collection used to test the implementation of I^3R consists of 3204 documents taken from the Communications of the ACM, from the years 1960 to 1979.

4.2.5.2 User Histories

The other part of the long term memory is the user histories, which consists of two parts, the user's domain knowledge and the records of the user's previous searches. The user's domain knowledge is organized the same way as the global domain knowledge. The user record has the following structure.

```
(Defstruct (User-Record
              (:Conc-Name UR-)
              (:Predicate UR?))
           (User-Name nil)
           (Session-Records nil)
           (Domain-Knowledge))
```

This record is stored in the VMS file system in a file with the name `<username>.model`. The user name corresponds to what the user's account name is on the particular machine.

The session records contain information that was in the short term memory when the session was suspended or ended. Briefly, a session record consists of:

1. The stereotypes that were in effect when the session was closed. (These will be discussed in the next section on experts under the user model builder.)
2. The request model consisting of the concepts that were judged relevant, any relevant phrases, and the documents that have been judged relevant.
3. If the session was suspended, the browse map if there was one, and the state -history.

A session record has the following structure.

```
(Defstruct (Session-Record
              (:Conc-Name SR-)
```

```
   (:Predicate SR?))
(Date nil)
(Stereotypes nil)
(Initial-Need nil)
(Document-Evaluations nil)
(Term-Weights nil)
(Browsing-Path nil)
(Session-Complete))
```

4.2.6 Experts and Short Term Memory Structure

In the current implementation of I^3R, six of the eight specified experts were implemented. The natural language expert (NLE) was not implemented since the needed techniques are still under development [Croft 87]. The explainer (EXP) was not implemented since it represented a significant piece of work in itself and it was felt that the utility of the I^3R could be demonstrated without it. The browsing expert will be discussed in chapter five.

4.2.6.1 Control Expert (Scheduler)

4.2.6.1.1 Purpose

One of the major differences between the traditional blackboard system and I^3R is the purpose of the control expert or scheduler. In most blackboard systems, the purpose of control is to constrain the system's activity to processing those KS instantiations that will contribute the most to solution of the problem. In other words, the scheduler's purpose is to conserve the resource of time.

In information retrieval, the purpose of control is to constrain the course of a session, so that it appears logical and consistent to the user, rather than being chaotic. In this sense, the control function could be considered a dialogue manager. What the control

function determines is the relative importance of the activity of the system experts during a given part of the session.

The control function must be organized to provide a flexible dialogue with the user. The work by Belkin [83], Brooks [83], and Daniels [85] has shown that, while the same basic steps are accomplished in every session, the order may vary considerably. Consequently, the control function has to be organized so that it can handle this variability. Their analysis goes fairly deep, examining the changes in focus down to the utterance level. In I^3R these lower level utterances are determined by the individual experts. For example, the domain knowledge expert would decide what concepts should be presented to the user for approval, the control expert would determine when the domain knowledge expert should be engaged in this activity.

Because the actual requests for and presentations of information, which correspond to the utterances of a human intermediary, are determined by the different experts, the control expert does not have to engage in a sophisticated process to determine the course of a session. Therefore, the control expert can be implemented as a state/transition network with relatively few states that encodes the general plan or sequence of activities for the information retrieval process. In this network, there are two types of states, intermediate and leaf states. The leaf states are where priority orderings for the experts are determined. The transitions are also of two types, normal and exception. Normal transitions encode the standard path through the states; exception transitions are taken when the session is not proceeding as expected. Figure 4.13 shows the structure of the network. Selection of what transition to take is based on parameters derived from the stereotypes determined by the UMB and by completion of certain functions; for example, the user has entered his query.

Another interpretation of these states are that they are goals to be met. When all the goals are met then the session is complete. Associated with each state, then, are the criteria

to determine when that goal is satisfied. At present there are two criteria, the number of relevant documents expected to be found and the number of searches expected to find them.

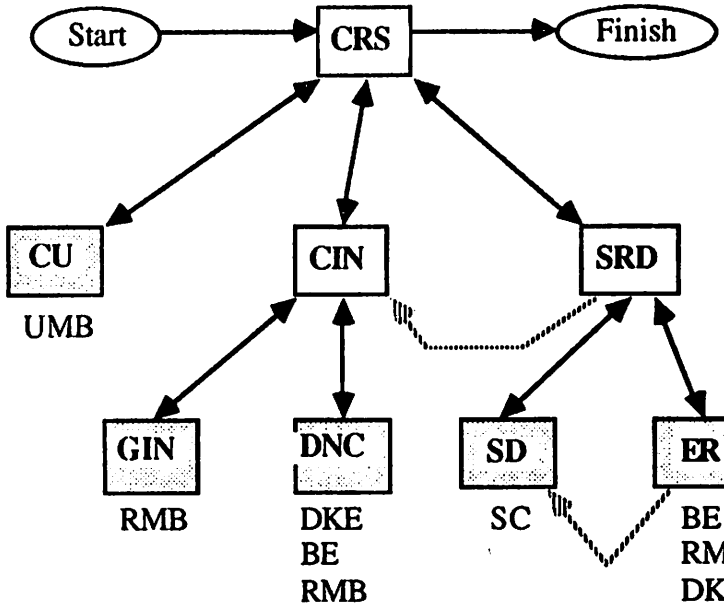

Figure 4.13: Control Expert States. Shaded states are leaf states where the relative priorities of the experts are established.

4.2.6.1.2 Conceptual Operation

The top level goal of Conduct Retrieval Session (CRS) simply represents the goal of the system, and it is met by the three subgoals of Characterize User (CU), Characterize Information Need (CIN), and Search for Relevant Documents (SRD) being satisfied, or the user indicating that he wants to quit or suspend the session.

The purpose of the first subgoal, Characterize User, is to allow the UMB to engage the user in a dialogue to determine what stereotypes apply to the user for the session. This allows the search and document expectations to be posted in the STM. Posting of these signals the satisfaction of the goal. Figure 4.14 shows the values that the control expert uses based on the user's stereotypes. These values are based on the following assumptions.

1. A domain expert will be able to specify the information that he is looking for precisely.

2. A domain novice will *not* be able to specify the information he is looking for with the same precision as an expert.

Because of the inability of the domain novice, it will require more searching to find relevant information. Furthermore, the domain novice may not recognize relevant documents. The values chosen reflect the abilities of these kinds of users.

Search Emphasis	**Domain Knowledge Expertise**	
	Novice	Expert
Exhaustive (recall oriented)	D = 15 S = 4	D = 20 S = 2
Selective (precision oriented)	D = 5 S = 2	D = 5 S = 1

Figure 4.14: Summary of control expert expectation values based on user stereotypes. D is the number of *relevant* documents expected, and S is the maximum number of searches needed to find the relevant documents

The next goal is Characterize Information Need (CIN) which is divided into two subgoals of Get Information Need (GIN) and Develop Need Context (DNC). The first goal (state) only allows the RMB to operate, during which the user enters his query in one of the query entry forms supplied by the RMB (see section 4.2.6.3). Completion of the state is marked by the internal form of the initial need being posted. The DNC state is characterized by an interaction between the domain knowledge expert (DKE) and the user where the DKE suggests, for user approval, additional concepts to be added to the developing information need. Any terms approved will be added to the request model by the RMB.

The browsing expert (BE) may also be active during this state, DNC. Whether or not it is depends on the user model and the history of the session. If control expert returns

to this state as a result of failure to retrieve the expected number of relevant documents in the number of searches allowed or if two searches in a row fail to retrieve any relevant documents, then browsing will be enabled. It will also be enabled if the user is categorized as a system expert. The state is completed when either the DKE has no more terms to suggest, or the user quits browsing.

Once the states of GIN and DNC are finished, then so is CIN, and the system passes to Search for Relevant Documents (SRD). If the search and relevant document expectations have not been met the control expert passes to the Search for Documents (SD) state where the search controller selects an appropriate strategy. When finished the SC posts the results in the STM and passes them to the interface manager (IM) for evaluation by the user.

The control expert (CE) then goes to the Evaluate Results (ER) state. Here the user will determine what documents and concepts are relevant. This information is of interest to the RMB, DKE, and SC. The user is also able to browse at this point depending on the context of the situation. If the user is browsing the RMB and the DKE will record judgements made during the process. The ER state is finished when either the user exits browsing after having evaluated at least one document as relevant or after having evaluated the search results.

After the results of the search are evaluated and the user has not found the expected number of documents the exception transition back to SD is taken, so that the SC can use the revised request model for a new search. If the expected number of searches has been taken and the expected number of documents has not been found, the CE will take the exception transition from SRD to CIN and then take the normal transition from CIN to DNC. This transition embodies the idea that the information need is still not defined sufficiently and needs further refinement by the DKE. If no new concepts are added the system will

suggest that the user browse if he has not already done so. If concepts are added, the CE will return to the SD to make use of the revised need model.

4.2.6.1.3 Implementation

The Control expert is implemented using Lisp symbols to represent the states (or goals) and rules to represent the transitions. State related information is kept on the property list of a state. This information includes the expert priority list for those states where the experts are operable. It also includes the substates that must be completed for the state to be completed; for example, the state CRS has three substates, GIN, DNC, and SRD that must be completed for it to be completed.

Since the transitions are implemented by rules, it is easy to add new transitions to the control expert. For example, say that the user model builder is expanded so that it also monitors how the user interacts with the system. At some point, the user model builder may wish to ask the user about how he works with the system, since the user's activity differs from that expected by the system based on the original model. A transition can be added that will take the system state back to the Characterize User state, so the user model builder can pose questions to make new determinations. Addition of this transition only requires a new rule that would respond to the conditions of the system and expectations set up by the UMB.

4.2.6.2 User Model Builder

The purpose of the user model builder is to determine where the user fits into the kinds of users that the system expects to find. Recognition of these stereotypical users is embedded in the rules that make up the UMB. The second purpose is to perform housekeeping functions for other experts by maintaining the user models kept in the long term memory.

In I^3R there are three kinds of information about the user that are important to maintain. The first is the user's domain knowledge. Since this information is primarily used by and manipulated by the DKE, the UMB simply performs the housekeeping functions of storing it at the end of a session and retrieving it at the beginning. The second kind of information is the characterization of the user along three lines, experience with computers, experience with the domain of the search topic, and interest in exhaustive or precise search. The third is the user's history, which is a record of the search sessions that the user has with the system. This information is used primarily by the request model builder, so here too the UMB performs housekeeping.

The primary reason for the inclusion of the user model builder is to demonstrate how a user model can be used to modify the behavior of the system. This is different from the use that user models were put to in GRUNDY, where the purpose was to assess the user and find books that would match or fit with that assessment. To this end, what the model should contain is determined by what behaviors are important to modify. Adaptability based on user models is another significant contribution of this thesis to the design and implementation of information retrieval systems.

The behaviors in I^3R that are important to modify are how the search controller responds to the information need, and how the interface changes with the respect to the ability and experience of the user. The second aspect is made up of a number of the kinds of choices that the user has for domain knowledge entry, the amount of information to be displayed on the two browsing maps, and how quickly the search controller will initiate a search while the user is browsing.

The UMB determines what stereotypes apply to the user by questioning him directly. It asks directly whether the user is interested in an exhaustive or a specific search to determine the interest in recall or precision. It poses a number of choices to determine the user's domain and system experience. These choices are for domain experience:

1. Know very little.
2. Have read a few news magazine articles on the subject.
3. Have read a few science magazine articles on the subject.
4. Have read a textbook on the subject.
5. Have read a few journal articles on the subject.
6. Have written journal articles on the subject.
7. Have written a textbook on the subject.

And for system experience are:

1. Seldom use a computer.
2. Use a word processor.
3. Own a personal computer.
4. Have never user an information retrieval system before.
5. Have used an information retrieval service before.
6. Frequently use an information retrieval service.

To these the user can answer yes or no; the default answer is no. If the user answer yes to questions 5, 6, or 7 of the first group, he is considered a domain expert. If he answers yes to 3, 5, or 6 of the second group, he is considered a system expert. From these determinations, the UMB posts on the STM on the `*User-Model*` place, its evaluations in the form of a list. For example, `((Domain Novice) (System Expert) (Search Recall))`.

4.2.6.3 Request Model Builder

The primary purpose of the Request Model Builder (RMB) is to keep track of the information provided by the user that pertains to the developing query. This includes the initial definition of the query, the single word concepts (also called terms) and multi-word concepts derived (called phrases) that the user considers important from the initial query, from concepts presented by the domain knowledge expert, from documents presented by

the search controller, and from browsing. Each term is stored in a hash table of records that is keyed by the term's number and its stem. The following is the record structure kept in the table.

```
(Defstruct
   (Term-Representative
      (:Conc-Name TR-)
      (:Predicate TR?))
   (Full-Text nil :Type String)
   (Stem      nil :Type String)
   (TermNo      0 :Type Integer)
   (Cfreq       0 :Type Integer) ; Collection Freq
   (Qfreq       0 :Type Integer) ; Query Freq
   (RelFreq     0 :Type Integer) ; Relevant Freq
   (User-Judgement nil)
   (Source nil)
   (Concept nil)
   (DK-Checked nil)
   (B-Recommended))
```

The RMB also performs stemming of any input text (see the example in chapter two), as well as providing different ways of initially specifying the information need. These specifications are:

1. free text,
2. a known document,
3. a simple Boolean query — terms occurring on the same line on the input form are implicitly ORed together, the separate lines are implicitly ANDed, and NOT is not allowed,
4. a complex Boolean query — an arbitrarily complex query using all three operators and parentheses.

Choices 3 and 4 are provided because many users are used to this form of specification or they may have previously developed queries on another system that they wish to transport. Boolean queries in either form would be processed in the following way [Croft 86a].

1. The component words are stemmed.

2. The query is transformed into a tree which is then used to generate candidate phrases for user approval. For example, the query, (parallel OR distributed) AND (processes OR programs) yields the phrases: parallel processes, parallel programs, distributed processes, and distributed programs.

3. The candidate phrases are presented to the user for evaluation. This step is required because the combinatorial nature of step 2 can produce phrases that do not make sense.

4. Phrases approved by the user are added to the request model.

The RMB also maintains a list of documents, on the STM place `*Doc-Evals*` that have been seen by the user. Each document is represented by a record:

```
(Defstruct (Document-Representative
              (:Conc-Name DR-)
              (:Predicate DR?)
            (ID 0 :Type Integer)
            (User-Judgement nil)
            (Nearest-Neighbors nil)
            (Terms nil)
            (B-Recommend nil))
```

Documents that have been judged relevant have their component terms retrieved and put in the appropriate field; documents that have been retrieved and are not judged relevant do not have their terms retrieved.

4.2.6.4 Domain Knowledge Expert

The DKE is one of the major functions in I^3R devoted to query refinement. Its responsibilities are twofold. The first is to build a model of the user's domain knowledge. The second is to search the available domain knowledge models for concepts that are related to those supplied by the user while describing his information need.

An underlying assumption in the operation of the DKE is that the user is the final authority on what is and is not relevant to his need. Therefore, the DKE acts as an advisor

to the user while the query is being developed. This means that the DKE will only suggest concepts for the user's approval, and will never automatically add any.

The DKE has, in the current design of the system, two sources in which to look for concepts, the global domain knowledge and, if present, the user's domain knowledge. There will be at least a minimal amount of information in the global domain knowledge consisting of term nearest neighbors. The order in searching for candidate concepts, is to search the user's knowledge before the global knowledge.

Searching for concepts proceeds by taking the terms in the request model that have not been checked previously by the DKE, and using them as entry points into the knowledge. From these entry points a modified form of spreading activation is performed. The links emanating from them are chosen in a specific order to find candidate concepts. The basis for the order is to find the words that are most likely to be, in the mind of the user, associated with the information that he is looking for.

The first links taken are the nearest neighbor links. These are relatively rare, therefore, if they exist it is very likely that the associated term is "about" the same topic. This not to say that the nearest neighbor is synonymous, but by reason of the association hypothesis (see section 2.2.2.3.2.3) it will be dealing with the same topic. The second link followed is the synonym link, since synonyms are defined to have the very same meaning. In effect, these words have to be interchangeable. These words increase the coverage of the request model, but do not expand meaning of it. Synonyms are especially important in fields where there are a number of synonymous terms for the same concept. For example, in graph theory, point, vertex, and node mean the same thing, and line, edge, and arc are also synonymous. The third link followed is the narrower link, which tends to make the query more specific. For example, a user may be interested in trees (botanical usage) and the DKE might find deciduous trees and coniferous trees as narrower terms. The user may or may not be interested in the distinction.

The next group of links to be followed tend to expand the query beyond its original meaning. The fourth link followed is the related link, which gets terms that bear some general association to another term or are a cross reference. The fifth link to follow is the phrase link that connects single word concepts to any phrases that they are members of. The last link that is followed is the broader-than link, which finds more general terms

This activation method differs from that used in most semantic link based systems. In those systems the purpose of the activation is to determine what relationship, if any, that the entry points have either to the other entry points or to some other designated set of points in the network. In earlier systems, such as Quillian's original one [Quillian 68], the activation sought to find any path between the entry points, and then to explain the path, thereby giving a description of the relationship of the points.

Each of the words used as an entry point is marked. If and when the DKE looks for additional terms, only those that have not been examined previously will be used as new entry points. Before the DKE further examines the domain knowledge for additional concepts , there is expected to be a search or some browsing activity providing some concept that can be used.

The other major activity of the DKE is updating of the user's domain knowledge model. This is done when the user is initially developing his query and when he is examining documents. During each of these activities, depending on the user model, the DKE will allow the user to enter different kinds of domain knowledge. A novice user is allowed to pick out phrases of interest, whereas an expert can enter all types.

Figures 4.6 through 4.9 show the interaction that the user has with the system to enter some domain knowledge. The actual operation of the system works in the following way. As the user selects the relationships, the full words are placed in list structures that reflect the relationships and words. These list structures are kept in fields of a record that is

part of every record that holds the content of what is displayed on the interface. This record is defined as follows.

```
(Defstruct
   (Common-Content-Info
       (:Conc-Name CCI-)
       (:Predicate CCI?))
   Display ;pointer to the display structure
   Relevant-Display
   Relevant-Display-Content
   DK-Display ; holder for DK acquisition displays
   DK-Type ;type of DK being acquired
   DK-Words ;words or phrases being related
   DK-New-Word ;holder for a word entered from keyboard
   DK-Phrase ;holder for phrase being built as part of
             ;another relationship
   Expanded ;a flag to indicate whether this node has
            ;been expanded in browsing
```

When a user selects a relationship to enter, the first thing that happens is a window is generated to display what the user is entering; the pointer to that display is held in DK-Display. This field may actually contain up to three display pointers, depending on what is being entered. This is the case of the situation shown in figure 4.7. When the user selects a word from the display, it is first pushed into DK-Word, and then displayed in the DK entry display that is either at the top or the bottom of the main document, concept, or query display. When `Entry OK` is selected the whole DK entry, for example, `(related distributed parallel)`, is pushed onto the list kept on `Relevant-Display-Content`. If `Cancel` is selected, the fields containing the information are cleared and the display is removed.

When `Done` is selected for a document or a concept the connections are sent from the interface manager to the system as a list of connections, as in:

```
((related (phrase concurrent processes)
```

```
                (phrase parallel processes))
    <other connections>)
```

After the domain knowledge is received by the DKE, each of the component phrases is stemmed and the words are entered into the domain knowledge with the original phrase put into the phrase field of the individual word record. Then, the phrases are put into the user's domain knowledge with the single words that make them up put into the phrase field of the record and the related phrase in the related field.

The only information that goes into the model is that which the user enters while examining a document or the query. The concepts that the user approves while he is evaluating the suggestions that the DKE makes go only into the request model. There is no migration of information from the global domain knowledge to the user's domain knowledge. The purpose of the user's model is to record the relationships he makes that are different from those in the global model.

4.2.6.5 Search Controller

The purpose of the search controller is to select the search technique or techniques that are appropriate given the user's interest in precision or recall, and the history of the session. This is an innovative feature of I^3R, since most systems are limited to a single search strategy. The system has two basic kinds of searches at its disposal, a probabilistic search based on the term independence model, and a cluster search. The cluster search in this system has a two variations depending on the links used to define the clusters. The primary cluster search uses nearest neighbor links; the other variation uses citation links. Other variations of the cluster search based on bibliographic coupling links, or cocitation links could be included, as well as searches based on different retrieval models such as the vector space model with the cosine correlation [Salton 68] or the extended Boolean model [Salton 83].

As has been mentioned previously, there is as yet no way to select retrieval strategies based solely on attributes of the query [Croft 84]. Therefore, other kinds of information must be used. This is the prime motivation for determining the user's interest in recall or precision. Besides the user's search interest, attributes of the search techniques should be taken into consideration, but there is little information to work with in relation to this. One piece of information is that cluster searches tend to retrieve different sets of documents than probabilistic searches for the same query [Croft 80] (see figure 2.1 and scenario three in chapter 6).

In order to make use of the available information heuristics have been developed to select what strategy to use in a particular situation. In developing these heuristics, knowledge of how human intermediaries perform searches is of no use, since their experience lies in manipulating Boolean queries in commercial systems. The heuristics of the search controller can be summarized as follows.

- Initial Searches
    - If the user is precision oriented, use a probabilistic search. The motivation for this is to use a well test search method as the basic technique.
    - If the user is recall oriented, initiate both a probabilistic and a citation cluster search. This will give the user the greatest number of documents to choose from. A large volume of documents is the goal in a recall oriented search.
- Subsequent Searches
    - If the previous search failed to retrieve more than two relevant documents, use the other search technique. The search had very low precision, so try another technique.
    - If the previous search was successful, more than 2 relevant documents, use it again with the modified request model. If a search has achieved a minimally acceptable performance, stick with it.
    - If two searches in a row fail to retrieve more than 2 relevant documents, signal this failure. The control expert will then put the system in a state where the browsing expert has priority and will suggest that the user browse. If all of the search techniques fail, then let the user do

some of the searching manually. This will cause the request model to be altered, so that the searches may work better later.

The search controller maintains a record of each search, so that it can keep track of all the documents that it has retrieved, the kind of search used, and the precision of the search.

The searches are implemented by means of a C program that performs all of the searches currently implemented. The search controller gets from the request model the term numbers, the term's collection frequency and occurrence in relevant documents, relevant phrases, and documents that have already been seen. This information, as well as what kind of search to perform is passed to the search program.

4.2.7 Interface Manager

The interface to the system is handled by an interface manager that operates independently from the rest of the system. The interface manager communicates with the rest of the system by placing messages on and reading messages from two places on the short term memory. It is primarily window oriented, and the following are the kinds of windows it supports:

- **System Messages Window** — displays textual messages from the system to the user.
- **Choices Window** — display choices to the user which may be selected by the mouse.
- **Query Entry Window** — a window produced by a text editor that allows the user to enter a query in a variety of different ways. These different ways are defined by forms that the user completes.
- **Menus** — there are two types, window and content. Window menus allow the user to manipulate a window by scrolling the contents, suspending it, etc. Content menus let the user make choices about the content of a window. For example, getting the bibliography of a document, or selecting the domain knowledge link that he wishes to use to connect two concepts.
- **Text Entry Windows** — these allow the user to enter character strings. Used in acquiring domain knowledge.

- **Document Window** — Shows the document title, author, abstract, and reference. Also used for displaying the query after it has been entered.
- **Document List Window** — shows the titles of documents. Used for search results, bibliography lists, and citation lists.
- **Concept Window** — displays the concept as well as all of the other words it is connected to.
- **Concept List Window** — displays concepts chosen by the domain knowledge expert to the user for approval.
- **Context Map** — Gives a graphic "road map" of where the user has been while browsing.
- **Neighborhood Map** — Shows the immediate neighborhood of a node in the Context Map.

The content of the windows is controlled by the experts via the content of the messages that they send. For example, depending on the UMB's categorization of the user, different choices for domain knowledge entry are made available by the DKE. A domain novice user is only allowed to select phrases he is interested in, whereas an expert can enter domain knowledge using any of the links and can enter text as well.

The interface manager is basically composed of two parts, one part to receive messages from the experts and display appropriate information, and another to receive information from the end user. The output portion of the IM runs as it is called from the main part of the system. This is done in several places. The first is during the part of the cycle when the Control Expert is determining what state the system is in. This allows it to pass any control information to the IM. The second place that the IM is called is after all the rules of the experts have been executed. This is when information that was the result of rule execution is passed to the IM for action. The messages passed to and from the IM have the following format:

```
(Defstruct (Inter-Process-Message
                (:Predicate IPM?)
                (:Conc-Name IPM-)
   (Message-No 0 :Type Integer)
   Msg-Id
```

```
Choices
Value-Type ;Choice-List, Done, Concepts, Text, etc.
Values)
```

These are placed in a list on an blackboard place named `To-IM`, and they are received back in a list on a place called `From-IM`. The interface basically takes each message from the place, `To-IM`, each cycle, decodes it and performs the action specified by the message-id. These actions consist of displaying information or changing parts of the interface, such as adding or removing choices from a menu, for example. Some of these actions provide a response back to the system and some do not. The system looks at the `From-IM` place on every cycle before the experts determine what rules to place on the agenda, so that they can act on information collected from the user.

The input part of the interface is primarily interrupt or event driven. Each mouse click or keystroke, depending on the window in which it occurs, causes an interrupt routine to be executed. These routines execute at a higher priority than the rest of the system, so the response to the user is fast. The only exception to this is the query input editor, which runs as a separate process while the rest of the system is suspended.

4.3 Implementation of the Blackboard System

Many of the design decisions in the way that the blackboard system was implemented were based on building the system inside of a single LISP image. More specifically, each expert does not run as a separate process communicating with a process that manages the blackboard. There are an number of reasons for this. As mentioned previously, on a DEC VAX minicomputer running the VMS operating system, a LISP process consumes a significant amount of resources. Running more than one process slows down the operation of the system considerably.

Descriptions of a blackboard architecture often make the statement that knowledge sources *look* for changes on the blackboard. This implies that they *actively* examine the

blackboard when they are not performing some other computation, and suggests a polling implementation. Polling in this situation would be a very inefficient implementation. A more accurate description would be that the knowledge sources *respond to changes* on the blackboard. This suggests an interrupt-driven or message-sending implementation, where the knowledge sources are idle until something of interest happens. The difficulty of this style of implementation is that the process that monitors the blackboard must know what interests the various knowledge sources. In the Hearsay II system, this information was simply that a hypothesis was posted on the level that a knowledge source examines. What to do with the notification was up to the KS. I^3R follows the message-sending view and provides the underlying process that runs the system with the information about what interests I^3R experts

4.3.1 Rule organization and execution

Because of the requirements to support explanation and incremental development, each expert in I^3R is implemented as a separate rule system. Rules can be interpreted as a condition/action pair or an antecedent/consequent pair. There is a subtle difference between the two interpretations. The first is an operational view and corresponds to the "`If-Then`" construct found in most programming languages.

The other interpretation is a logical view, and means that the antecedent and the consequent are related by modus ponens. This interpretation means that the rule is a specification that is interpreted by an underlying mechanism. The rules can be used in both directions. An example is the BNF specification of a programming language. If the specification is used as a generator, starting with the start symbol and working toward the terminals it is a top-down interpretation. If it used as a recognizer, working from the terminal symbols to the start symbol, it is bottom up interpretation.

In I^{3}R the rules were implemented using the condition/action interpretation. The primary reasons for this choice were the following:

1. Rules represent the high level control structure in each expert, where the actions are performed by algorithmic processes; for example, stemming or search. It was not intended to implement the entire operation of each expert using rules. In some cases, it would have been grossly inefficient to do so; for example, the search processes.

2. Ease of implementation with regard to the blackboard. Since it was desired to have more than one rule fire, if possible, during a major cycle of the system, use of commercially available systems was infeasible.

4.3.1.1 Rule form

The form of a rule is relatively simple, it is a 4-tuple consisting of:

```
(<expert name> <rule#> (<conditions>) (<actions>))
```

The expert name and the number simply indicate where the rule belongs. The actions and conditions are the meat of the rule. The conditions are a list of 4-tuples and the actions are a list of triples. The condition 4-tuple is:

```
(<BB-place> <action-name> <predicate-name> <arguments>).
```

The essential meaning of a condition is that if a certain action is taken on a specified place, then check the blackboard place with the given predicate and arguments. The action triple is:

```
(<BB-place> <action-name> <arguments>).
```

Similarly, this means perform the specified action on the BB-place with the specified arguments.

The actions and the place names provide the mechanism for notifying experts when something has happened that is of interest. At each blackboard place a list is kept, indexed by action names, of the actions that are interesting to particular experts. This information is extracted from the condition part when a new rule is added to the system. This list is accessed by the rule execution code when an action is performed by using the `bb-place` and `action-name` part of the condition. For example, the bb-place called `*Domain-`

Knowledge* would have on its list of actions an action called Add-DK. This action is interesting to the request model builder, since the new domain knowledge may contain words that are not already in the request model. When Add-DK was performed a message would be sent to the RMB of the form (Add-DK *Domain-Knowledge*). This would cause all the rules of the RMB that have this action/place pair in any one of its conditions to become a candidate rule for selection.

4.3.1.2 Rule processing

When the rules are stored in the system for an expert, they are processed in the following way. First, each rule is stored in an array at the location that corresponds to its rule number. Then the condition part is broken up into individual conditions. These are used to form an inverted list of action-place indexes that point to the rules that contain them. To illustrate this, an expert, E1, has the following three rules:

```
(E1 1 ((P1 A1 <predicate><args>)
       (P2 A2 <predicate><args>)) (<action part>))
(E1 2 ((P1 A1 <predicate><args>)
       (P2 A3 <predicate><args>)) (<action part>))
(E1 3 ((P1 A1 <predicate><args>)
       (P3 A3 <predicate><args>)) (<action part>)).
```

When these rules are read in,each one is put in an array indexed by its rule number. The condition parts are then broken up into the following triples:

```
(P1 A1 1) (P2 A2 1) (P1 A1 2) (P2 A3 2) (P1 A1 3)
(P3 A3 3)
```

These are combined into lists:

```
((P1 A1) (1 2 3)) ((P2 A2) (1)) ((P2 A3) (2))
((P3 A3 (3))
```

These lists are put into a hash table, keyed by the place/action pair. These place/action pairs are also used to key operations on places back to the experts that are interested in them. For example place P1 would have on its action list the following:

```
((A1 (E1 <other experts>)) (An (<experts>))...)
```

So, when an action is performed on a place, the experts that have rules that have that place/action combination in their condition part can be notified, by means of a message, of that event.

When an expert is allowed to operate by the control expert, it looks at its list of incoming messages notifying it that some action has occurred on a place that it is interested in. For each one of these messages, the expert retrieves all of the rules, by means of the index table, that have the place/action pair in their condition part. The system forms a triple that consists of the rule number, the number of conditions it has in its condition part, and the number of messages that refer to it. After all the messages have been processed, the list of triples is sorted; the major criterion being the number of conditions in the condition part, and the first minor criterion being the number of messages that refer to the rule; the second is the rule number. The system then goes down the list and takes the first rule whose condition part is true and places it on the agenda for potential execution during that cycle.

This ordering is how the system performs conflict resolution. It is based on conflict resolution heuristics that give priority to the most constrained rule, the rule that is responding to the most activity in the system, and the numbering of the rules.

The actions and the predicates are implemented as Common Lisp functions. Some of these may call-out to programs written in C.

4.3.1.3 System operation

A detailed description of the system operation is the following.

1. The scheduler determines what state it is in, which specifies what experts can operate during this cycle of the system. This is done in the same way that the rest of the experts determine what rules will be eligible for execution. The stopping criterion for the scheduler is the attainment of a leaf state.

2. Any messages from the interface manager are put in the short term memory and appropriate action-place pairs are sent to the experts.

3. Each expert that is allowed to operate by the control expert looks on its list of messages that contains action-place pairs that correspond to condition parts of its rules.

4. Every rule that is indexed by an action-place pair is retrieved and is ordered by the rule conflict resolution criteria.

5. The rules are then checked in order to find the first rule that is true. This rule is placed on the agenda for execution.

6. Once the agenda is set, it is executed in the order determined by the scheduler in the following way.

    a. The first rule is executed.

        i. Any BB-place that has been altered is put on a list of altered places.

        ii. For each place that is altered a message consisting of an place/action pair is sent to each expert "interested" in that place and action.

    b. The rest of the rules on the agenda are executed in order.

        i. A rule is checked to see if any of the altered places are in its condition part. If so, then the condition part is rechecked to see if it is valid.

        ii. If it is valid, then it is executed in the same fashion as the first rule (steps 6.a.i & 6.a.2).

        iii. Get the next rule.

        iv. If any more rules, go to step 6.b.i.

7. Any messages for the interface manager are sent .

8. Goto Step 1.

4.3.2 Interface Manager

The interface manager (IM) represents a significant portion of the code that implements I^3R. In its initial design, it was to run as a process separate from the main part of I^3R. This proved infeasible, since running two Lisp processes on a single processor severely degrades performance. It was desired to implement the IM in Lisp to retain the flexibility and superior development environment that Lisp provides.

With these constraints in mind, the structure of the interface manager can now be described. Figure 4.15 presents a schematic view of the IM structure. The `*Display-Table*` holds pointers to all of the records that define a major window on the screen, which are displays such as documents, concepts search results, the context map, etc. Neither menus nor domain knowledge entry windows are considered major windows and are associated with a particular major window. The first six records hold pointers to the windows that the system always has, which are the system messages window, the questions window, the neighborhood map, the context map, query display, and the relevant documents display.

The other displays are lists of concepts, lists of documents, single concepts, or single documents. Each display is represented by a record that holds information related to the VMS UIS (User Interface System) display and a pointer to the content of the display. The content is either an array of pointers to documents or concepts, or is a single document or concept. The documents and concepts items are kept in hash tables, the concepts are kept in their respective global or user domain knowledge hash tables, and the documents are kept in a document hash table, which is keyed by document number.

Each display is represented by a record structure that keeps the size of the window, the associated interrupt functions, pointers to the menu displays, the coordinates of the object on the browsing map, and a pointer to the content of the display. The content is either a document, a document-list, a concept, or a concept list. The system messages window have no associated content and the choices displays are a special case of a menu display. The interface manager data organization is summarized in Figure 4.15.

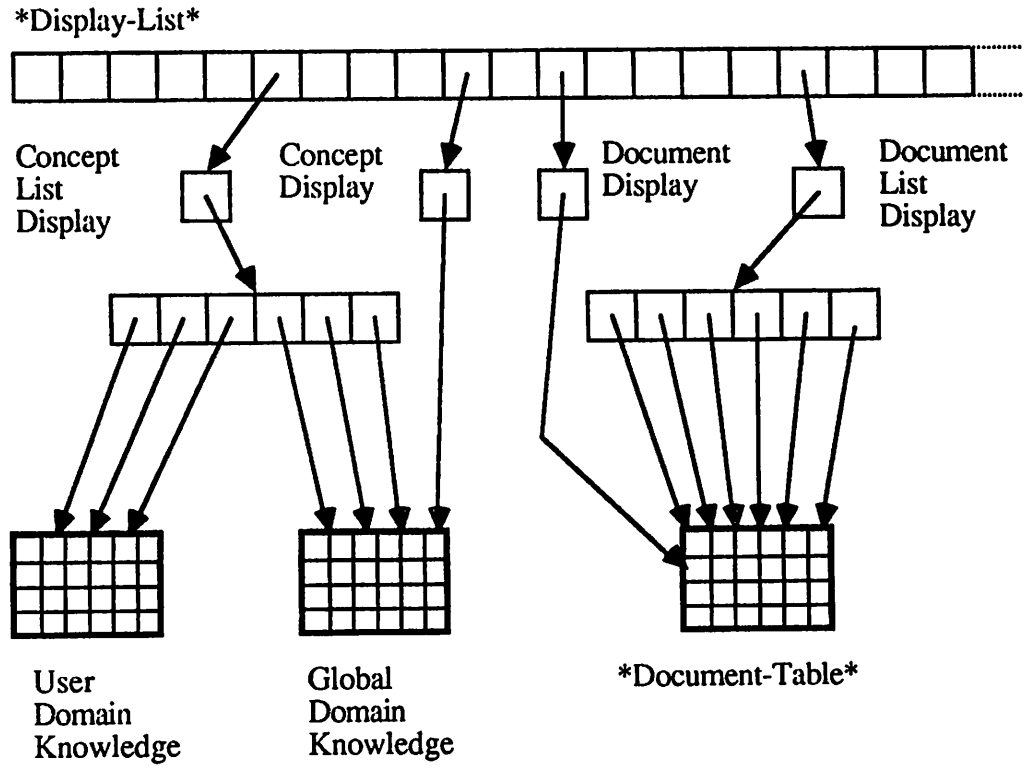

Figure 4.15: Organization of Interface Manager data.

4.4 Summary

In this chapter, an architecture for an intelligent interface for information retrieval has been presented. It meets the requirements established in the previous chapter in the following way. New large scale functions can be added easily by means of adding new experts to the system. This can be accomplished simply by adding the expert's name to the priority lists of the control expert in the appropriate states, and writing the rules. Each expert can be incrementally developed by adding or changing rules. Rules also make explicit the decision criteria, so that the operation of the system can be explained. The system provides different ways for the user to enter his query and allows him refine his query by having a flexible interface that supports the entry of domain knowledge and browsing. The system can use multiple search strategies to find candidate relevant documents. Finally the system allows the user to take complete control of a session by browsing.

CHAPTER 5

BROWSING EXPERT

5.1 Introduction

In this chapter, the browsing expert is examined. Browsing is an alternative method for both query refinement and information search. It gives the user direct access to the concept/document knowledge so that he may explore to find the information he needs. The end user is assisted in the browsing process by recommendations made by the expert and by graphical displays that provide the user with context, showing where he has been and the immediate neighborhood of his current location. Browsing, in concert with the automatic construction of the request model and multiple formal search techniques provide a synergistic retrieval environment that overcomes the disadvantages of using browsing as the sole means of information search.

5.2 Definition

Browsing is an informal or heuristic search through a well connected collection of records in order to find information relevant to one's need. The searcher evaluates the information currently being displayed to determine its value relative to the information he is seeking. Once this evaluation is made, the user then selects the next item to display and evaluate. Browsing is also a feedback process, but it differs from traditional relevance feedback in two major ways. The first difference is granularity; the user only examines one item at a time evaluating its relevance and selecting another item for view, instead of having to examine the results of an entire search. The second difference is the locus of control; it is the user that determines the items to be examined rather than the system.

5.2.1 Advantages

Bates [Bates 86] points out the advantages of browsing in the context of term selection by showing how it takes advantage of two cognitive capabilities. The first one is the greater ability to recognize what is wanted over being able to describe it. This concept is evidenced in the production artist renditions of criminals by means of "paste up" facial sections, where an individual picks out facial features that they recognize of the individual being sought. The second capability is being able to skim or perceive at a glance. This allows a searcher to evaluate rapidly a large amount of material, determining what is useful in it. Browsing makes use of these abilities by showing the user examples of information that match his current model, as expressed to the interface, for evaluation. A system that allows a user to browse and to do so quickly may provide information that the user wants and may not have been able to describe, and quite possibly all the information that the user needs. In this way, browsing addresses the content specification problem of query formulation described in chapter 2.

Besides being an alternative to a formal or parameterized search, browsing serves to acquaint a user, unfamiliar with a domain, with the structure of its information. This tutorial use helps those users that cannot find the right words to express their information need or that do not know how terms in a particular domain are used. This kind of browsing is dependent on having a high quality thesaurus, which may or may not be available. Some domains, like medical science, have very well defined structures that are embodied in thesauri such as MeSH (Medical Subject Headings). Other domains do not have such a readily available source of knowledge, but domain knowledge collected from domain experts can be a source for this kind of information.

5.2.2 Use in Other Systems

Because of these advantages, a number of systems use browsing as a means of finding information. These systems include hypertext and text retrieval systems, database systems [Motro 85], and object oriented programming systems [Goldberg 1983]. In the hypertext and text retrieval systems browsing has been used for both query formulation assistance and finding document information directly. Some prototype systems developed to tackle the query formulation problem are CANSEARCH [Pollitt 83], CALIBAN [Frei 83], and CoalSORT [Monarch 87]. These systems take an existing classification system and automate it to give a user online access in order to select terms for a query. The structure of the knowledge is organized as frames with various kinds of relationships, depending on the field to which the system was applied, connecting them. In both CALIBAN and CoalSORT the user is required to manually construct a query. CANSEARCH is oriented directly at producing a query. Consequently, it leads the user to a greater extent than the others to specify certain types of information. As the user makes evaluations and selections, the system includes them in the developing query.

There have been a number of text oriented systems that use browsing as a method of search. An early system is ZOG [McCracken 84], which is designed to be a general purpose human-computer interface. The fundamental mode of operation is menu selection with the basic unit of information being a screen-full of text called a frame. The information in a ZOG system is handbuilt using the built-in editor. The main organization is a tree, but there is no restriction on how frames may be linked. Search is done by traversing the frames until one finds the information one desires. ZOG's advantages are that it is easy to use, requires very little training, and is fast.

BROWSE [Palay 81] is a system built on top of ZOG and more oriented to document retrieval. The set of frames combined document abstracts with a concept classification hierarchy, author, and journal information. It overcomes some of the limitations of

ZOG by including a partial map of the frame structure. This map is limited to only the concept hierarchy. A further improvement is the addition of a search capability. At any time the user could make a selection to perform a parameterized search of the documents. However, the query for the search still has to be manually constructed by the user.

Browsing is also found in the hypertext or dynabook systems [Kay 77, Weyer 82, Trigg 83]. These kinds of systems are characterized by text units of approximately paragraph or page length connected by various kinds of links. One class of links organizes the text units hierarchically into subsection, section, chapters, and papers or books. Other kinds of links provide citation referencing and editorial commentary. TEXTNET [Trigg 83] maintains over 50 kinds of links. Once the user has entered the system, he is free to meander through the network examining the text. These systems also provide a sophisticated user interface, giving the user a number of ways to get information about the document that he is in.

A major problem with these systems is that browsing is, generally, the *only* way to find information in them. Either no facility or only a rudimentary one is provided to perform a search. Searching through that many units of text by browsing only would be a formidable task. In one case, the dynamic book [Weyer 82], an index structure similar to that of a book is provided so that the user can "jump" from one place to another. However, this prototype was constructed from a history text, and therefore, the amount of information and the subject matter was constrained.

THOMAS [Oddy 77] is an interface program that employs a different type of browsing. In this work, the system does not rely on a pre-existing complex, highly connected database of documents. Instead, a model is built by the system of the user's interest as the user evaluates the information presented to him. The model is different from the kind built by CANSEARCH in that it is domain independent. This aspect of THOMAS is

significant since it relieves the user of having to manually construct a query, other than the initial few terms.

Another difference in the operation of THOMAS from the previous characterization of browsing is that the system takes the initiative, after the user entered a few initial terms, in selecting what to show the user. Even though this would seem constraining, the system is quite flexible in its interaction. It can determine when the user is not making progress, and ask him to reevaluate previously seen abstracts. It is similar to the ZOG systems in that it presents the user with only one item of information, a document abstract, at a time for evaluation. However, the user can indicate that individual elements of the abstract are relevant, rather that only being able to evaluate the whole item.

5.2.3 Disadvantages

The disadvantages of browsing as a means of search are that given the complexity of the concept/document database it is very easy to get lost. This is one of the major problems of ZOG-like or hypertext systems. Once the user is deep into the database, he may have forgotten how he got there and has only the current frame for context. This results from these systems being "memoryless," and providing only a simple interface and a structure. Another disadvantage is that browsing is labor intensive. The user may have to examine many pieces of information before finding anything that is relevant. In I^3R this is overcome by the availability of formal search techniques that can be used after the user has judged a number of documents or concepts relevant.

5.2.4 User Heuristics

How does a user browse? How does he get started, and how does he determine what to view? In order to determine what item will be displayed next for evaluation, the

searcher uses heuristics. Some examples of browsing heuristics that a searcher might use in a document retrieval system are:

1. If the current document is interesting:
    a. What else has been written by its authors?
    b. Are any of its references interesting?
    c. Are any of the documents that reference it interesting?
    d Are any of the documents in the same journal issue, conference proceedings, etc. interesting?
    e. Are there any documents that are very similar to it in the database?
2. If the current term is interesting:
    a. Does it have any synonyms, narrower terms, etc.?
    b. What documents is it used in?

All of these heuristics depend on the kind of links maintained by the system. For example, if references are not maintained, then heuristics 1b and 1c cannot be used. The richer the set of links, the more ways that the user can move through the database. By having a rich set of links, the system is responsible for helping the user understand what the links mean, how they might be used, and where he is in the network formed by them.

Browsing can also be seen as a form of constrained spreading activation in a semantic net [Cohen 87]. The heuristics, in this case, constrain the choice of paths that the user selects from. Each path is evidence that a document or concept is related to another document or concept. For example, if two documents share a number of very common terms, these documents are likely to be related only on a very general level. If one of the documents cites the other, the likelihood of them being related is greater.

Since the concept/document database has a large variety of links, the user has many possible ways to navigate through the information. Furthermore, because of the way that the concept knowledge is fused with the document knowledge, the user is given much more latitude to find information of interest. The user can explore the structure of the do-

main knowledge in a variety of ways. For example, he can look at all the phrases that a particular word is used in, as well as how the concepts are used in the documents. This is in contrast to the BROWSE system [Palay 81] where the user is restricted to the tree structure and the few cross reference links of the hand-coded domain knowledge.

5.3 Browsing Operation

The Browsing Expert (BE) provides assistance to the user in three major ways. First, it makes recommendations about nodes connected to the current node that it considers likely to be useful. Second it remembers where the user has been, so that he can retrace his steps to return to interesting nodes that he has seen in order to pursue different paths. Third, in concert with the Interface Manager, it provides visual context, so that the user can avoid getting lost in the complex and potentially confusing structure of the database.

The BE, in a manner similar to the Domain Knowledge Expert, acts in an advisory capacity, allowing the user to be the final judge of the usefulness of the information displayed. The advice is given to guide and not restrict the user options; the user can always ignore the advice of the BE, and elect to go off in a direction of his own choosing.

5.3.1 Browsing Interface

The advice given by the BE is reflected by what is displayed on the browsing maps. These maps are generated by the IM and consist of the neighborhood map and the context map. The neighborhood map (figure 5.1) gives a picture of the nodes that are immediately adjacent to the current node of interest; the context map (figure 5.2) gives the view of a larger area around the node of interest, so that the user can get an idea of the path that he has been pursuing. Both maps consist of nodes representing concepts, documents, lists of documents, and connectors connected by links marked as to their type or frequency (in the case of document to concept links). Nodes are filled with different patterns indicating

whether they have been visited, recommended, or judged relevant. The shape of the node indicates what kind of item it represents; circles represent documents, squares represent lists of documents, octagons represent concepts, diamonds represent connectors, and over sized boxes represent reminders. Reminders are markings made by the user on the maps to indicate some node that he would like to come back to and examine at another time.

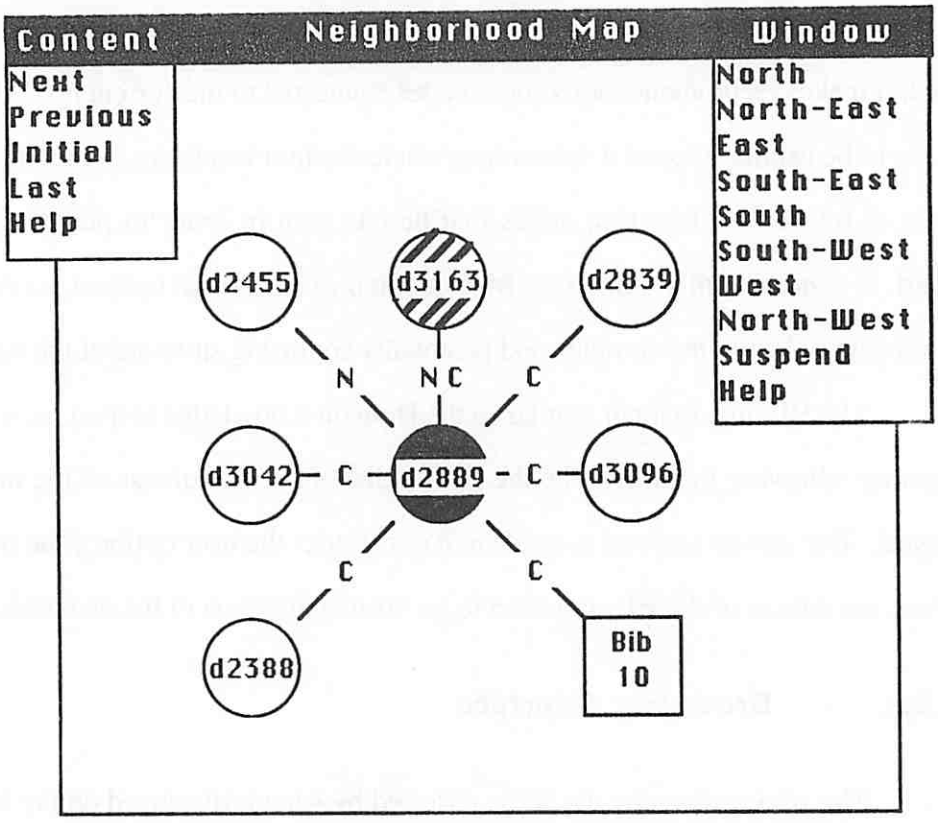

Figure 5.1: Sample Neighborhood Map. Markings on the links indicate the kind of link: C = Citation, N = Nearest Neighbor.

When the user views the node again, the reminder is removed. This gives a cue to the user if he is scrolling around on the maps that there is something that at one time caught his eye.

The possible number of links that can emanate from any node, representing a concept, is potentially large. A moderately frequent single word concept (also called a "term") may be found in as many as fifty documents, even in the CACM test collection. Besides this, a concept can be linked to many other concepts by the different links. For example, a

term such as "algorithm" in a computer science document collection could be a part of many phrases.

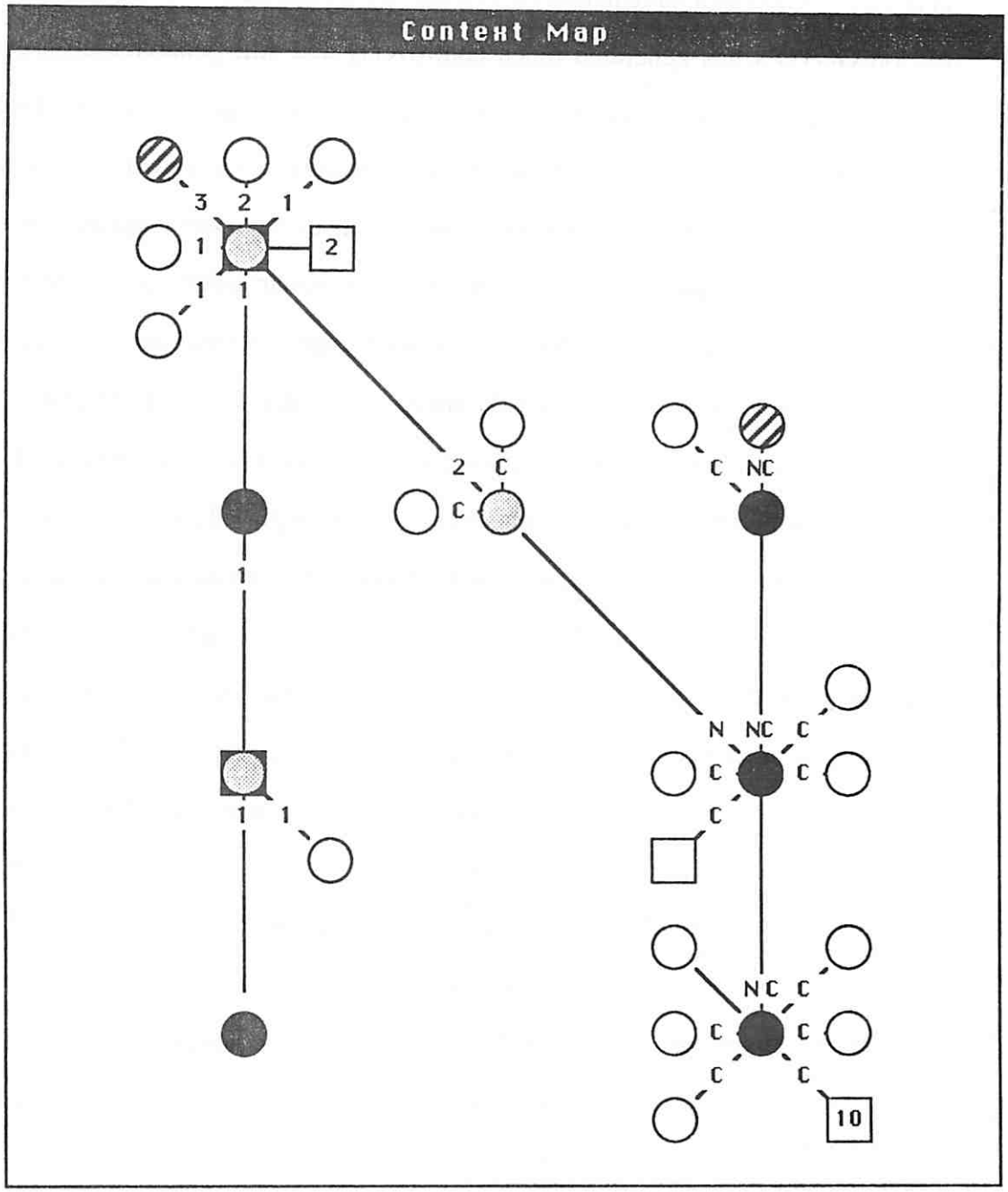

Figure 5.2: Sample Context Map.

The situation for documents is similar. With reference links, a document may have anywhere from just a few links, in the case of a short correspondence with one or two

references, to over a hundred, in the case of a review article (see [Fox 87b]). A document also may be cited by a large number of documents; for example, Dijkstra's letter [Dijkstra 68] on GOTO's has generated much controversy and still generates discussion (see editorial comment after [Klein 88]), so it is referenced many, many times. Furthermore, a document may typically have as many as twenty to thirty terms as part of its representation.

This potentially large number of links cannot be effectively displayed on a graphic screen. Therefore, the number of nodes that can appear around a node of interest is reduced and the organization of the screen is made regular so that the information can be effectively managed and displayed. The maps are organized as a square grid (see figure 5.3), which is not displayed. Only one object is displayed in a cell whether it is a link or a node. This limits the number of new nodes that can be displayed around a central node to a maximum of seven, since every node (but the initial node in the map) has an input link. Nodes in a neighborhood are drawn one cell away, with the links and their labels being drawn in the adjacent cells. Different symbols indicate different kinds of elements; circles represent documents, octagons represent concepts, squares represent lists of documents, hexagons represent journal issues, and diamonds are connectors. These symbols are shaded to indicate their status. Black indicates a relevant element, gray indicates an element that has been viewed by the user, diagonal bars indicate a node that is recommended by the BE. An oversize square around another node is a mark that the user can place to indicate a node that he might wish to come back to. In addition, the nodes are labeled; document nodes have the document number, concept nodes have the concept, list nodes have the number of items that they represent.

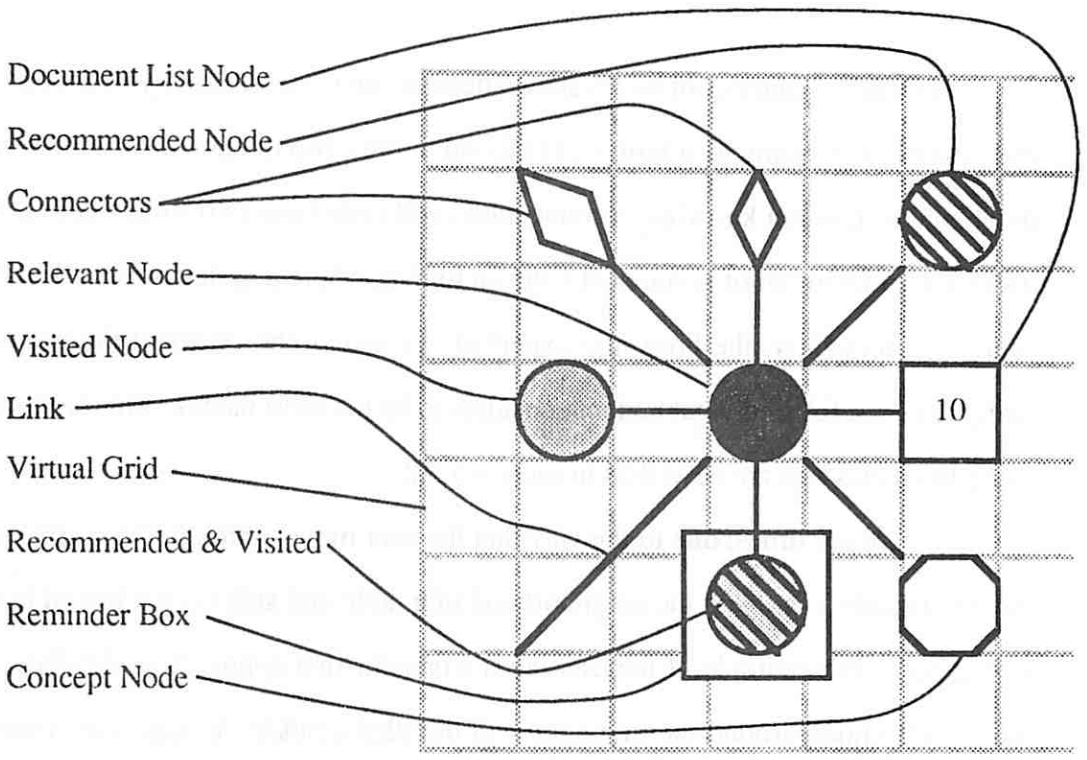

Figure 5.3: Grid for browsing maps showing different node markings, but without labels.

Organization of the browsing maps in this way was done with two considerations in mind, cognitive economy and machine efficiency. It is desirable not to overwhelm the user with a mass of nodes, such as can be the case even for an average document. It is not unreasonable to assume that a document might contain 10 terms and 10 references, making 20 links to other items. To display this many nodes requires a great deal of space on the screen. If a node is represented by a 1 inch diameter circle, the 20 related nodes have to be drawn on an approximately 6 inch diameter circle around the center node. Furthermore, consideration must be given to the space needed for labeling the links.

When the user traverses a link to examine the contents of another node and desires to see the neighborhood, the link must be extended and space must be found to draw the neighborhood. Initially, this is not a problem, but as the user continues to examine different nodes, it becomes very difficult to manage the space. Furthermore, many of the nodes may not be examined, so their presence only contributes to the visual clutter.

The actual number of nodes shown depends on the connectivity of the node and the user model. For example, a term that is found in only two documents, and has no nearest neighbors or domain knowledge connections will only have two links. A system expert will have a maximum of seven nodes shown to him, where a system novice will only have four. Consequently, the browsing expert must make a choice about the nodes that are likely to be useful, and what node it considers to be the most useful. This is accomplished using heuristics that are described in section 5.3.2.

There are times, due to the way that the user moves through the maps, that it may not be possible to expand the neighborhood of a node and still keep it linked to the originating node. For example, if the user takes a breadth-first approach to viewing the nodes, the neighborhood around the first node will be filled quickly. In this case, a connector is used. The node to be expanded is replaced by a connector, and its neighborhood is drawn somewhere else on the map where there is sufficient open space. It is drawn with a connector on its input node. If the user selects the connector, the map is moved to the location of the neighborhood. If the connector on the input link to the new neighborhood is selected the map is moved back to the neighborhood of the originating node.

As mentioned previously the user can elect to go off in a direction of his own choosing. To do so, the user selects from the choices available in the content menus of the windows that display a document or a concept. In the case of a document, the user can look at the reference list, citation list, nearest neighbor list, or journal issue list. If one of these is selected, a node representing it will be placed on the maps. This is shown in figure 5.4. The reference list for the selected paper is:

1. Friedman, J.H., Bently, J.L., Finkel, R.A. *An algorithm for finding best matches in logarithmic time*. Stanford CS Rep. 75-482.

2. Blum, M., Floyd, R.W., Pratt, V., Rivest R.L., Tarjan, R.E. *Time bounds for selection*. Stanford CS Rep. 73-349.

3. Finkel, R.A., & Bently, J.L "Quad Trees: a data structure for retrieval on composite key." *Acta Informatica* 4, 1(1974), 1-9.

4. Knuth, D.E. *The Art of Computer Programming, Vol I: Fundamental Algorithms*. Addison-Wesley, Reading MA, 1969.

5. Knuth, D.E. *The Art of Computer Programming, Vol III: Sorting and Searching* Addison-Wesley, Reading MA, 1973.

6. McCreight, E. Computer Science 144A midterm examination, spring quarter, 1973, Stanford University.

7. Rivest, R.L. *Analysis of Associative Retrieval Algorithms*. Stanford CS Rep. 74-415.

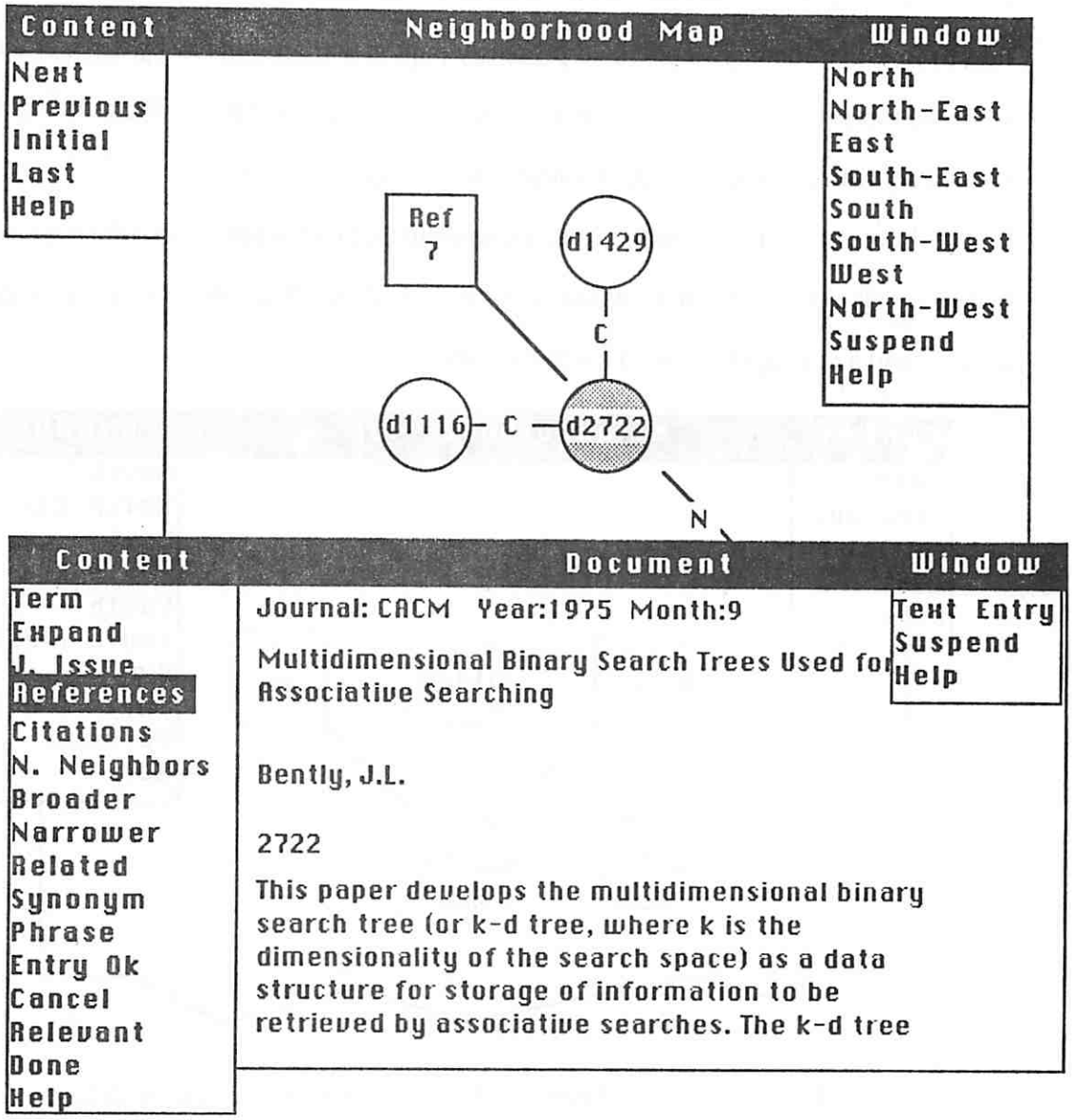

Figure 5.4: User chooses the `References` selection.

This list would be displayed in a document-list window with a banner labeling it as `Reference List: Document 2722`. None of these papers are available in the current collection, so the user cannot go anywhere. If any of the papers turned out to have an interesting title, which the user selected for viewing, the document list node would be expanded and the document node drawn around it. He then could choose the `j.issue` option to find that the issue contained primarily papers that are related by the fact that they were the top entries in the 1975 Forsythe student paper competition. The only citations in the current collection are already displayed on the map.

In the case of a concept, he can choose the `Select` option from the content menu and then select the concept he wishes to examine. A window with the new concept then appears and a concept node is added to the map.

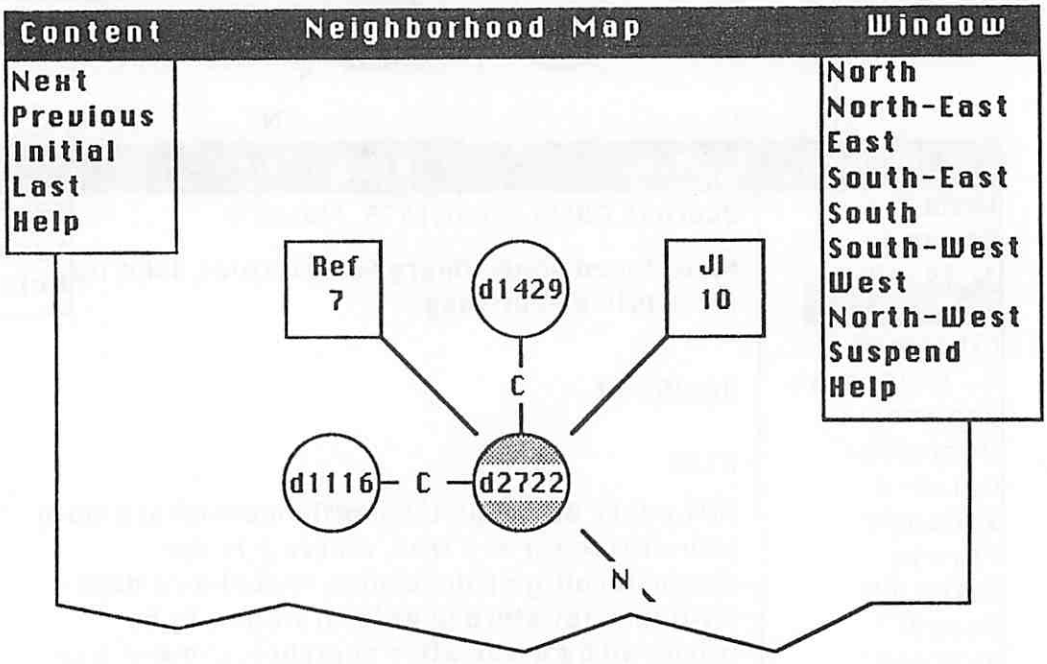

Figure 5.5: Neighborhood Map showing the addition of Reference and Journal Issue nodes.

5.3.2 Browsing Heuristics

The browsing heuristics make use of the evidence provided by the links and the request model to recommend nodes that are strongly related to the node currently in view. Those that are related by multiple pieces of evidence are selected as the most promising to examine. None of the heuristics infer recommendations by looking more than one link away. The heuristics are divided into two kinds, concept heuristics and document heuristics.

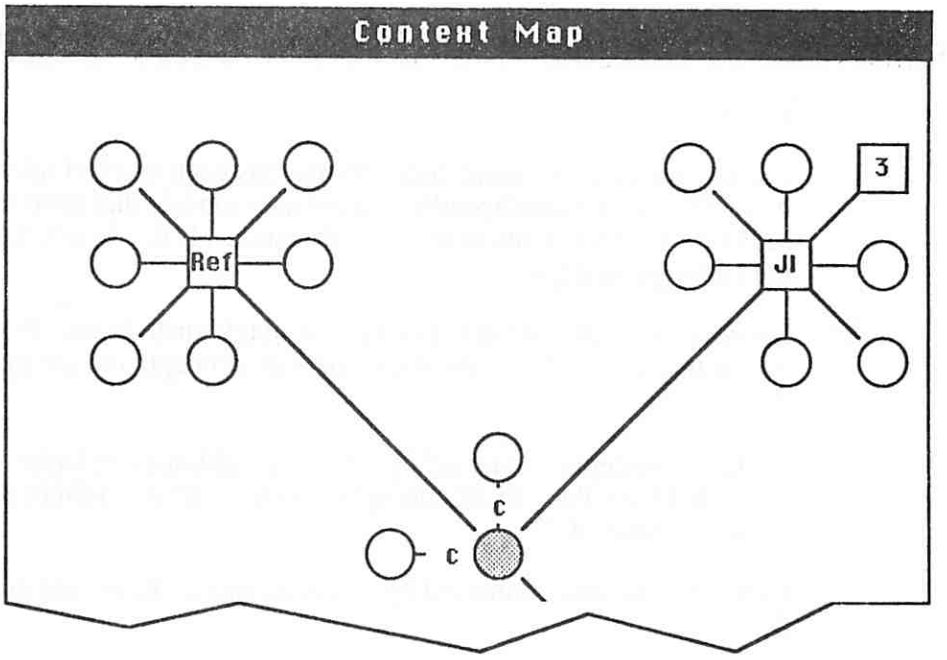

Figure 5.6: Context Map showing configuration if the Reference and the Journal Issue Nodes are expanded (Content and Window menus not shown, but are the same as a neighborhood map).

5.3.2.1 Concept Heuristics

The concept heuristics contain one underlying assumption; since documents are the fundamental units of information in the system, the user should be directed towards documents rather than other concepts. Consequently, when a concept occurs in 4 to 7

documents (depending on the user model) that have not been seen, the recommended nodes will be those documents.

In determining what concepts to show if the previous condition does not hold, the most important links between concepts are the *nearest neighbor* and the *synonym* links. In the domain knowledge there should be relatively few synonym links, since these represent a very strict view of synonymy. Similarly, for any single word concept there should be few nearest neighbors, since only at most the top five most similar ones are saved. The next most important are the *related-to* and *phrase* links. The following are the heuristics of choosing concepts.

1. For any term (single-word concept) that has been marked relevant and occurs in 4 to 7 documents (depending on the user model) that have not been viewed select those documents as the neighborhood. If this is not the case, perform the subsequent steps.
2. Retrieve concepts connected by nearest-neighbor-to links. These are concepts on the nearest neighbor list of the *current* concept, and are given a weight of 3.
3. Retrieve concepts connected by nearest-neighbor-from links. These are concepts that have the current concept on *their* nearest neighbor list, and also are given a weight of 3.
4. Retrieve concepts connected by *synonym* links. These are given a weight of 2.
5. Retrieve concepts connected by *related-to* links. These are given a weight of 1.
6. Retrieve concepts connected by *narrower* links. These are given a weight of 1.
7. Retrieve concepts connected by *phrase* links. These are given a weight of 1.
8. If there are any tied scores, order by document frequency giving higher preference to those in fewer documents.

Once the list has been determined, the BE will take the top 4 to 7 as determined by the user model and send them to the interface manager for display. The highest ranked will marked as the recommended one.

While a user is viewing a concept, he may decide to look at the documents in which it occurs. The BE will act differently depending on the frequency of the concept in the collection and the user specific values determined by the UMB. The selection of the documents to display as nodes is determined by the frequency of the concept in the document and by the date of the document; the newest is given preference. If the number of documents is less than the number of nodes specified by the user model, all of the documents are displayed as nodes connected to the concept (figure 5.7b) and the links are marked by the concept frequency. If the number of documents is more than the number specified by the UMB or if there is no room around the node, then some of the documents are displayed as individual nodes and the remaining are represented by a document list node (box) that is marked with the number of documents it represents (figure 5.7c). If the user decides to examine the documents represented by the document list node, then that node is extended and as many document nodes as possible are shown with the remaining documents again being represented by a box node (figure 5.7d). If the concept occurs in more than 13 documents and the user is a system expert, the system asks if he really wants to look at that many documents. The value of 13 is chosen because of the organization of the map. Figure 5.8 shows the number of nodes that can be displayed around a document list node and one extension of it; there are 13 nodes surrounding the two document list nodes. If any more documents are on the list a third document list node would be required. This extra node would be shown far to the right, in this case, of the original node of interest, and would lead the user away from the immediate neighborhoods of the original node. If the user does want to look at that many documents, the document list nodes are expanded as before. If the user is a system novice, he simply is not allowed to look at that many documents, and a message informing him of that fact will be displayed in the system messages window.

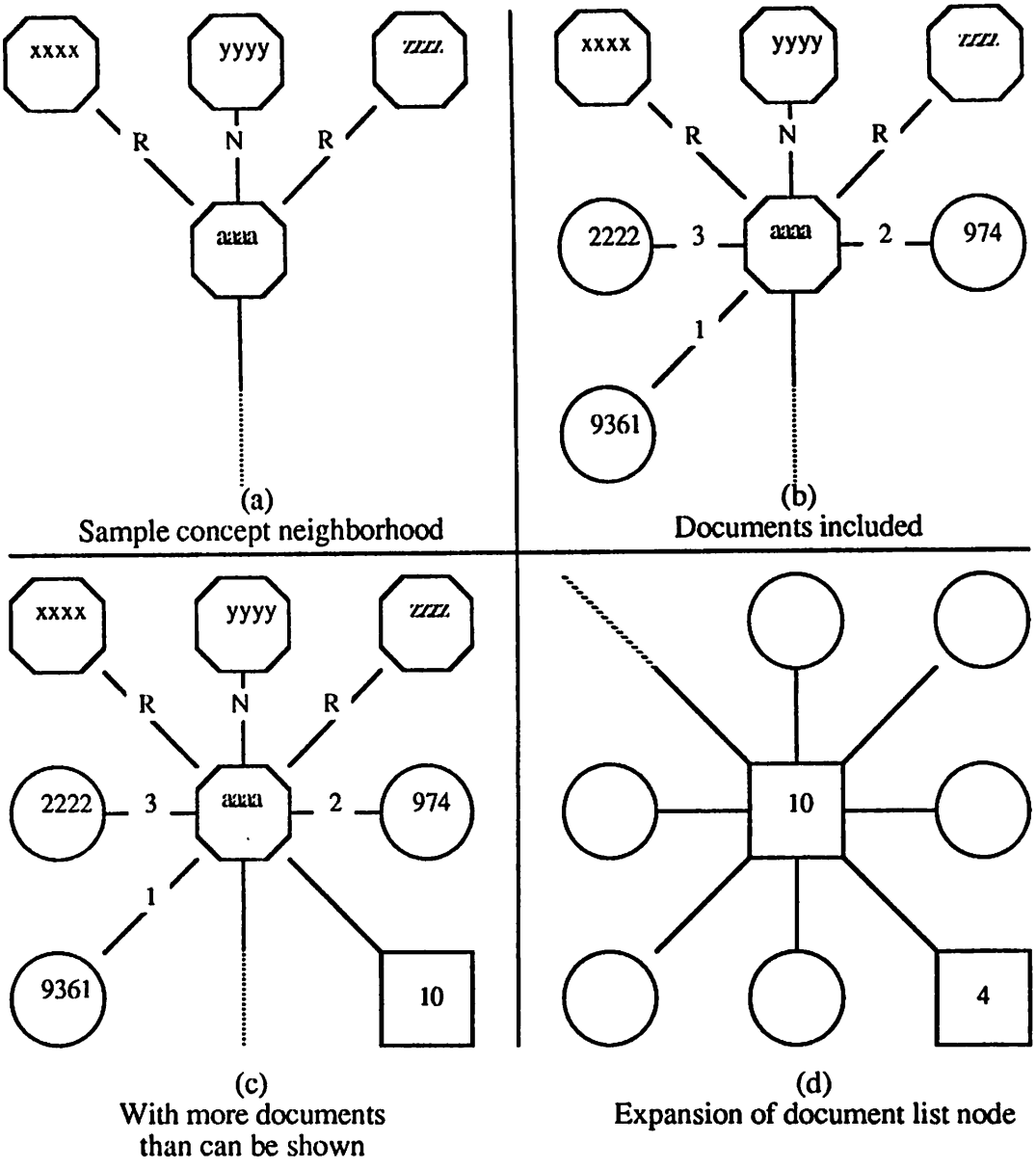

Figure 5.7: Example term neighborhoods (node markings are: R = Related, N = Nearest Neighbor, <number> = occurrences of a concept in a document).

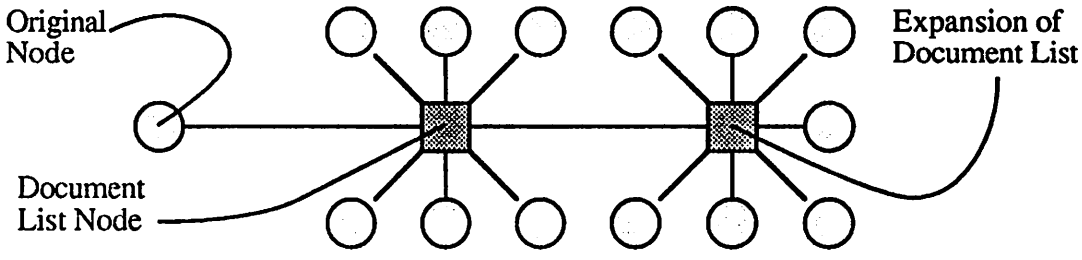

Figure 5.8: Expansion of a document list.

5.3.2.2 Document Heuristics

The document heuristics are similar to the concept ones. The most important links are the nearest neighbor links (to and from, both given a weight of 3). After this is the citation link (given a weight of 1), and then the reference link (given a weight of 1). However, since a document can potentially have many references and be cited many times, an evaluation of the documents connected by the citation links must be made. A document that is cited many times is given a higher value than one cited few times. This is based on studies of citation patterns [Salton 83] indicating that citation of a document is a relatively rare occurrence. Consequently, a document that is referenced many times is significant. The quantification of this significance for the browsing heuristics is:

1. Cited 2 to 3 times, an additional weight of 1
2. Cited 4 to 5 times, an additional weight of 2
3. Cited 6 to 9 times, an additional weight of 3
4. Cited 10+ times, an additional weight of 4.

The small numbers are due to the particular test collection being used to develop the system. It has information only about documents in the collection. A document could be cited many times, but since the citations are from other than the CACM, they are not available to the system. In a production system they would be tuned to more accurately reflect the frequency of citation in the document collection. If there are any tied scores, they are resolved by ranking on the actual number of citations.

An evaluation of a document by the size of the reference list is not so straightforward. A document with a large reference list may be good if the user is a novice in the domain of interest, since it might be assumed that the article is a survey of some field. However, a document that has only a few references may also be valuable, if it cites the current document of interest. This implies that the citing document may be closely related. If a document with few references is cited by the current document nothing can really be

determined about its potential value. So, if the user is a domain novice, a document with a large (10, because of the test set) reference list is given an additional weight of 2.

In both cases, nodes that have been viewed are not included in the list of recommended nodes. It is assumed that the searcher can remember what he has examined in a particular session, and if not, then he can access them by looking at the list of documents judged relevant, the lists of search results, or by retracing his path.

The user is not limited to viewing the nodes that are displayed on the neighborhood map. When he selects a node for viewing from the map, a window containing the textual information appears. He may decide, for example, to examine the reference list of a document node. A list of titles appears, from which he can choose any one. If he chooses one that is not on the map, a node will be put up representing the document chosen. In this way, the map always maintains the path that the user has taken. Should the user care to backtrack, he can scroll a map window back over a node marked with a reminder box, select it for view, and move in a new direction.

5.3.3 Browsing Model

The model in the STM by the BE is the "path" that the user has taken through the network. Each node on this path represents a node that the user has viewed in the course of the session. And with each node the recommendations made by the BE when the node was first visited are saved. The path is connected, even though the user may have entered and left browsing several times, so while the entry and exit points may not be connected in the concept/document database, they are in the path model. This allows a user to retrace his steps through all the nodes that he has seen while browsing in an entire session. The IM provides four commands for retracing:

Last – moves the user to the last node viewed; the end of the path.

Next – moves the user to the next node on the path, if not at the end.

Previous – moves the user to the previous node on the path, if not at the beginning.

First – moves the user to the first node viewed; the beginning of the path.

New nodes are added to the path only when the user examines a new node. While this sounds obvious, it means that the path does not record *every* move that the user makes. For example, a path at some point while browsing consists of the nodes A through E, and the user has retraced his steps back to node C. From this point, the user moves to a new node, F. The path would consist of the nodes A through F and not A, B, C, D, E, D, C, F. If the user desires to trace back the exact path, he has the maps as references.

When the session is finished (not suspended), the path is saved as part of the session history, except that the recommendations are not retained. If the user comes back to the particular session and there have been no changes to the concept/document database the recommendations will be similar. The documents already seen will not be considered as candidates for recommendation during the new session. If the session is suspended, all the context information including the path and the maps is retained, so the user would see exactly the same scenes as he did previously.

5.4 Browsing Implementation

The implementation aspects of browsing that have not been discussed previously in relation to the interface manager or the implementation of the system architecture lie in the construction and maintenance of the browsing maps, and the structure of the browsing model.

Recall that the organizational principle behind the browsing maps is a square grid upon which all the nodes and other symbols are drawn. These symbols are stored in a hash table that is keyed by the grid coordinates. The reason for this is two-fold. First, it allows the interface manager to immediately determine if there is a symbol on the screen in any one of the cells by hashing on the coordinates. If there is a hit, then the cell is already taken. Second, there is no way ahead of time that the IM can know how much the user will

browse and, therefore, it has no idea how much space to allocate for the grid storage. Since hash tables in Common Lisp are automatically extended when they become full, the system need not limit the user's ability to browse because of space considerations, and can allocate a small table initially.

The records that are stored in the map table have one of the following structures:

```
(Defstruct (Node
             (:Conc-Name N-)
             (:Predicate Node?)
  ;;  Icon Types: Doc, Concept, Doc-List,
  ;;  Journal-Issue
  Icon
  ;;  Values:(A)uthor, (C)oncept,
  ;;                   (D)ocument
  ;;                   (JI) - Journal Issue,
  ;;                   (R)eferences, (Ci)tations
  Value
  ;;  a pointer to the content of the node or the
  ;;  ids of a node.
  Content
  ;;  Where the node was developed from
  Predecessor
  ;;  A list of links emanating from this node
  (Successors (Make-Array '(8)))
  ;;  Has the node been visited, relevant,recommended
  Status
  ;;  Text that goes on the icon
  N-Label ; label on the neighborhood map
  C-Label ; label on the context map
  ;;  Where th node is on the maps
  Coordinates
  ;;  Whether the node has been marked as interesting
  ;;  This means a reminder box is around it
  (Marked nil))
```

```
(Defstruct (Connector
                (:Conc-Name Connect-)
                (:Predicate Connector?)
   Coordinates
   Source
   Destination)
```

The map table can be saved if the session is suspended, and when resumed the map can be reconstructed. When the map table is saved, the pointer to the content is replaced with the identifier of the object pointed to. For example, with a document, the document number is put in; with a document list, a list of documents is put in; with a concept, the text of the concept is put in.

Since the information that determines the contents of the browsing map is stored in the map table, the structure of the browsing model is simple. It consists of a list of coordinates of the cells that the user has been to. Any information required can be obtained by using these coordinates to key into the hash table.

5.5 Summary

Browsing is potentially an important technique for retrieving documents from large knowledge bases. Its advantages are that a user receives immediate feedback from the structure of the knowledge base and exerts complete control over the outcome of the search. The primary disadvantage is that it is easy to get lost in a complex network of nodes representing documents and concepts. Furthermore, there is no guarantee that browsing will be as effective as a more conventional search. These disadvantages can be avoided by providing facilities for controlling the browsing and for using the information derived during browsing in conventional search techniques.

CHAPTER 6

EXAMPLE SCENARIOS

6.1 Introduction

In this chapter, four example scenarios are presented to demonstrate the operation of I^3R, and to show how the system as a whole operates. Before these scenarios are presented, the difficulty of evaluating highly interactive systems like I^3R is discussed. It should be noted that I^3R is a prototype system, consequently parts of the interface are not as well developed as those of a production system would be. Furthermore, the screens shown in this chapter were composed using a drawing program to facilitate easy inclusion into the text, and are taken from screen dumps of the system while operating. The only differences between the screens seen in the text and those seen by the user are in the typefaces and in the sizes of the windows and symbols.

6.2 Evaluation

Evaluation of a highly interactive system such as I^3R presents a number of challenges. The first is a matter of retrieval effectiveness; how will the addition of the facilities provided by I^3R help increase the performance of the system. One way to answer this question is to notice that, since the system relies on retrieval techniques that have a sound theoretical and empirical basis, it will not perform any worse than those techniques. Furthermore, since Boolean query retrieval systems do not perform as well as statistically based techniques, I^3R will perform better than a Boolean system. This, however, is inadequate since it only tells us the minimum performance to be expected, does not consider the amount of effort that the user must expend to use the system, and provides no real way to compare the effectiveness of this system with others.

Major difficulties arise when considering how to evaluate a system like I^3R that has a variety of different functions that all contribute to the operation of the system. One difficulty is getting genuine needs and another is getting genuine users to interact with the system. Use of the standard test collections only provides a partial foundation for testing. In the test queries, there is no indication of the user's interest in recall or precision. One possibility for ascertaining this is to count the number of relevant documents; many relevant documents implies that the query is recall oriented, and few implies that the query is precision oriented. This, however, is a simplistic categorization. It ignores the possibility that a query may be recall oriented, but there is little information on the topic in the database, so there will not be not many relevant documents. Or, the opposite case, in which the user is looking for a specific piece of information that may be contained in many documents. One possible solution to this problem is to develop an new set of queries for one of the collections, such as the CACM collection, since it is not too large, and have the query authors indicate their interest in either an exhaustive or a specific search, as well as supplying other pertinent information This would include additional domain knowledge. A possible way to get this kind of information need description would be to extend the dialogue analysis technique used by Belkin et al. past the presearch interview stage to include retrieval with the user present,with the intermediary, helping him evaluate the results of the search.

The problem of getting genuine users is also very difficult. One possibility is to form pseudo-users, much as Oddy did in evaluating THOMAS [Oddy 1974]. The aforementioned dialogues could be analyzed to determine how users would react to specific situations arising in the course of a session. This analysis would produce rules that describe this behavior. The advantage of this approach is that the users remain constant, making the test relatively repeatable. This approach has a number of problems. First, it is not possible to determine all behaviors of the users using different facilities. Second, it is

not possible to determine the reaction of the user to new facilities that were not present when the sessions were being developed.

The alternative is to test using real users; this too has many difficulties. It is impossible to give one user the same need to use on a system as novice for testing various combinations of facilities, since the user will have learned something about the problem the first time he uses the system. This will affect how he uses the system to deal with the need in subsequent sessions. This learning process, however, can be used as training for the user. In subsequent sessions, the user can take different system experience stereotypes, progressing from novice to expert, so a test subject can participate in more than a single experiment. This points out, however, the need for a great number of test subjects for experiments to test all the possible combinations of facilities that a system like I^3R has to offer. These combinations include not only the use or non-use of a particular expert, like the domain knowledge expert for example, but the use of different heuristics in implementing the expert. This situation is exacerbated by the requirement to have a statistically large enough sample to reduce the occurrence of any biases. Although this kind of testing requires a significant investment of resources (ie. perhaps thousands of test subjects), it is in the opinion of experts [Spark Jones 88], the only legitimate way to proceed. This kind of testing is far beyond the current resources available for the I^3R research.

6.3 Scenarios

With the difficulties of evaluating a highly interactive system in mind, this section of the chapter presents example scenarios to indicate how the system works. These scenarios are meant to be illustrative of the flexibility of the architecture, and show how the it assists the user to find the information he desires. The query used in scenarios one, two, and three is taken from the 52 test queries that are a part of the CACM test collection. This

means that there is an established set of relevance judgements for the query. The query used in scenario four was developed by the author, who having a background in computer systems was qualified to make relevance judgements.

6.3.1 Scenario One

The first scenario demonstrates the basic operation of the system. The user is a domain and a system novice. In this scenario, much of the internal operation of the system will be shown. Each section will summarize what happens during the cycle. There are a number of occasions where the user will be thinking about what his response should be. In this case the system simply "spins." This points out a difference in the implementation of the experts from traditional rule based systems. Typically, when the system has nothing to do, (ie. no rules to fire) that signals the end of processing. In I^3R it means that the system has nothing to do for the moment, but that does not means that more information of interest to an expert is not forthcoming from the user. The system ends when it reaches the end state.

6.3.1.1 Cycle 1

The first part of every system cycle is the operation of the control expert to determine the state of the system. The initial state of the system is $Start. The CE uses rule 2 that recognizes that it is in `$Start` and changes it to $CRS (conduct retrieval session). Since this is not a leaf state, one that has an associated priority list of experts, the CE continues to operate. Since a state has been changed, the rules that are associated with state change are active. The rule selected changes the CE state form $CRS to $CU (characterize user). This is a leaf state, so a priority list for the experts is established, which consists of one expert, the user model builder, UMB. Figure 6.1 shows schematically the portion of the plan that the scheduler has traversed.

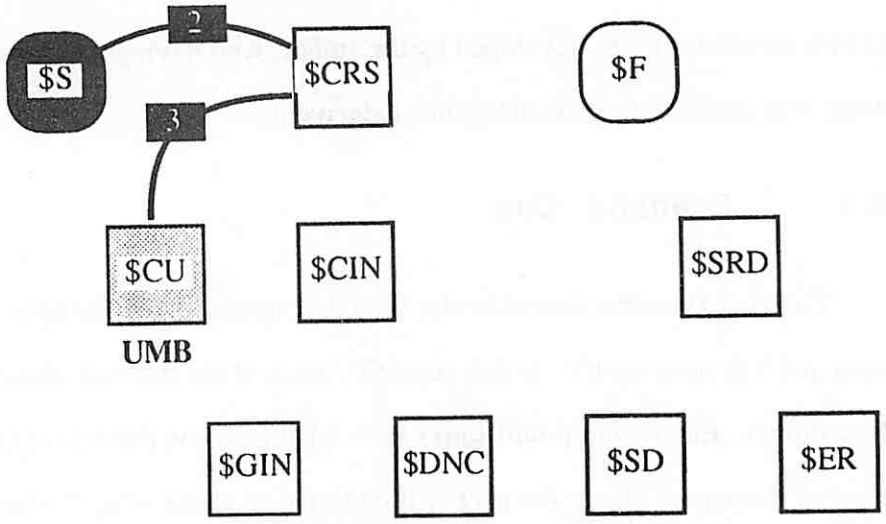

Figure 6.1: First portion accomplished of the CE plan. The transitions taken are shown in bold lines and marked with the rule that they correspond to. A state or goal that has been "satisfied" is filled with black; partially satisfied goals are filled with varying shades of gray.

The UMB recognizes that there is no user model, so it sends a message,

```
(IPM :Message-No 1 :Msg-Id UMB-User-Name :Choices 1
:Value-Type Name :Values Nil),
```

to the interface manager, IM, telling it to ask for the user's name. Figure, 6.2, shows the initial state of the interface with the user being prompted for his name.

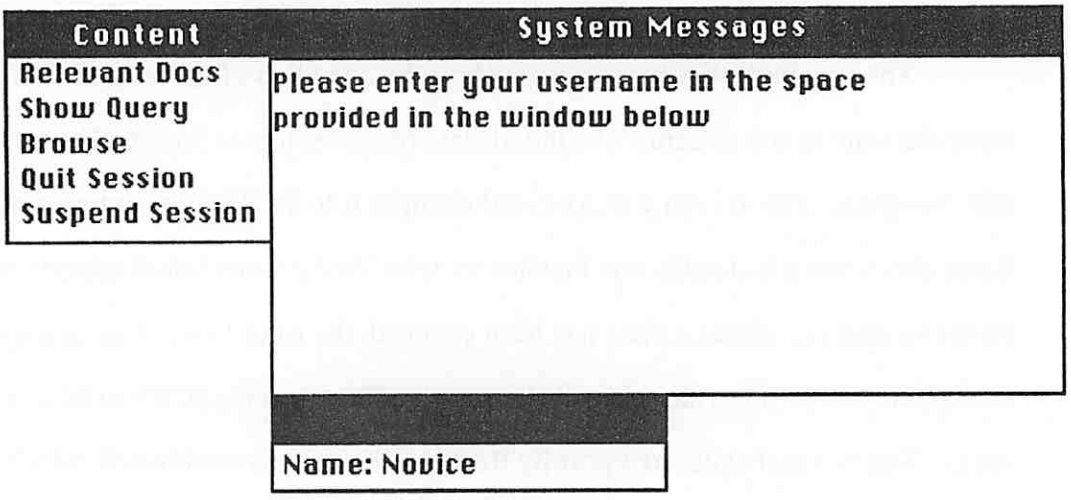

Figure 6.2: Initial state of the interface.

6.3.1.2 Cycle 2

The user enters his name and keys `return`. From this action, the IM returns the message with the `:Values` field filled in with the user's name, `Novice`. The CE runs through another cycle looking for either a change of state or a change of the condition of the STM. Since neither of these has happened, at least nothing the CE is interested in, it remains in the same state, so the priority list remains unchanged.

As part of determining which rule to fire, the UMB searches the system for a file containing the user's model, which has the name `Novice.model`. Since there is no such file, it builds an empty model, which means that there are no previous session records and no user specific domain knowledge.

The UMB begins the process of asking the user questions to determine the appropriate stereotypes, by sending a message with the first group of questions. The message is

```
(IPM :Message-No 2 :Msg-Id UMB-System-Experience
:Choices Any :Value-Type Choice-List :Values (UMB-
Seldom UMB-Word-Proc UMB-Own-PComp UMB-Never-IR,
UMB-Used-IR UMB-Freq-IR))
```

Each of the values in the list following the keyword `:Values`, refers to a question that is stored in a table of questions that the IM accesses. The display derived from this message is shown in figure 6.3. In this figure, only the first five questions are shown; the rest are viewed by selecting `Scroll Down` from the window menu.

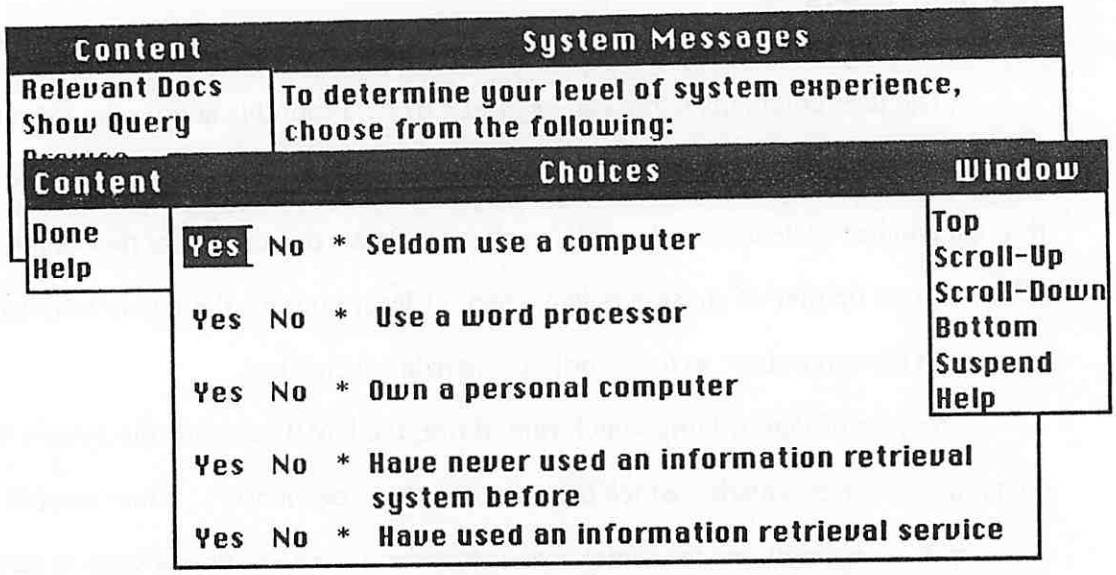

Figure 6.3: System prompting the user to answer questions that will determine the appropriate stereotypes. These choices determine system expertise.

6.3.1.3 Cycle 7

A number of cycles have passed while the user determines the choices he will select, and indicates that he is done with the question. Since the determination of the user stereotypes is not complete the CE will stay in the $CU state. Upon receiving the message back with the user's selection placed in the `:Values` field (in this case he selects only the first choice, (`UMB-Seldom` **`Yes`**), indicating that he seldom uses a computer) the UMB determines that he is a system novice, and posts this on the STM place `*User-Model*`, which then has the following list bound to it, `((System-Type Novice))`.

6.3.1.4 Cycle 8

Upon recognizing the establishment of the system-type stereotype, the UMB sends a message to the IM with choices for the user that will indicate his domain knowledge expertise. This message is:

```
(IPM :Message-No 3 :Msg-Id UMB-Domain-Experience
:Choices Any :Value-Type Choice-List :Values (UMB-
```

```
Know-Little UMB-Read-NewsMag UMB-Read-SciMag UMB-
Read-Text UMB-Read-Jrnl UMB-Write-Jrnl UMB-Write-
Text)))
```

The display derived from this message is shown in figure 6.4

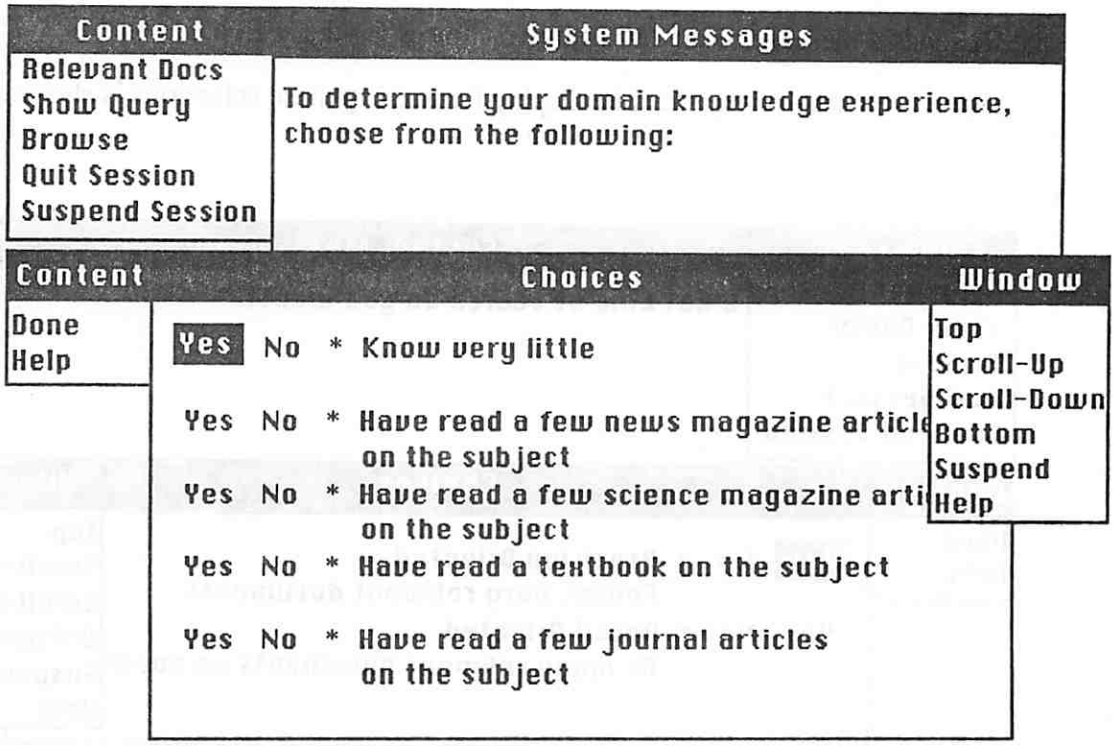

Figure 6.4: These choices determine domain knowledge expertise.

6.3.1.5 Cycle 14

The user responds by selecting `Know very little`. The symbol for this message along with the response, `(UMB-Know-Little Yes)` is placed in the `:Values` field of the message and returned. From this response the User Model Builder determines that the user is a domain novice, and adds `(Domain-Type Novice)` to the list bound to the STM place `*User-Model*` resulting in the list `((Domain-Type Novice) (System-Type Novice))`.

6.3.1.6 Cycle 15

In a similar manner as the system and domain stereotypes were determined, the UMB sends a message to the IM with two choices for the search orientation, `UMB-Precision` and `UMB-Recall`. The display for making these selections is shown in figure 6.5.

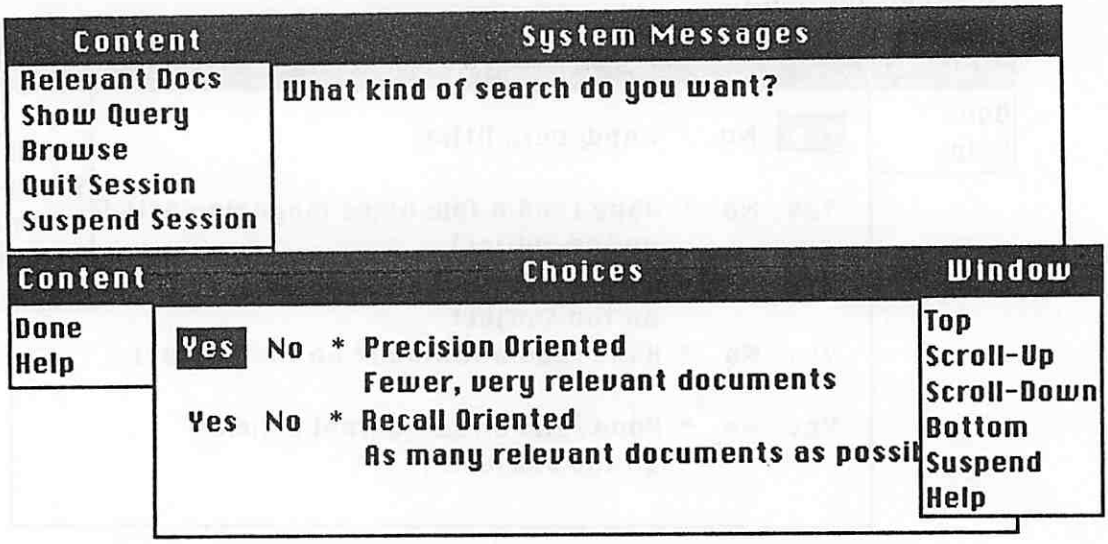

Figure 6.5: These choices determine search orientation.

The user selects the **`Precision`** choice which is then transmitted back to the experts.

6.3.1.7 Cycle 24

The UMB receives the message and posts the final stereotype on the user model, so `*User-Model*` now has the following list bound to it: `((Search-Type Precision) (Domain-Type Novice) (System-Type Novice))`.

6.3.1.8 Cycle 25

The UMB now evaluates the stereotypes that it has determined apply to the user and determines other parameters that also apply. These are the expectations of the number of relevant documents, and the number of searches that it will take to find them, that the con-

trol expert uses to determine the course of the rest of the session, which in this case are 2 searches and 5 relevant documents. The stereotypes also limit the choices that the user has from which to choose in many of the interface windows, and affect the number of nodes shown on the browsing maps, which are four for a system novice and seven for a system expert. They also affect the number of documents that will be judged relevant before the system initiates another search and whether or not the system asks the user before initiating one. For a domain novice the number of documents is 3 and for a domain expert it is 5. The STM place `*User-Model*` now has the following list bound to it: `((Searches 2) (Relevant 5) (Candidates 4) (Search-Type Precision) (Domain-Type Novice) (System-Type Novice))`. The UMB also marks the user model as finished.

6.3.1.9 Cycle 26

Since the user model is now complete, the focus should shift from characterizing the user to characterizing the information need. This shift is now done by the CE. It recognizes that the UMB has marked the user model as being finished. This causes a rule to fire that takes the system from $CU back to $CRS, and marks the state $CU as finished or satisfied. The next rule recognizes this and takes the system from $CRS to $CIN, characterize information need. This state has two substates, neither of which at present is satisfied, so the next CE rule that fires takes the system to $GIN, get initial need. This state is a leaf state, and its priority list consists of one expert, the request model builder (RMB). This is shown in figure 6.6.

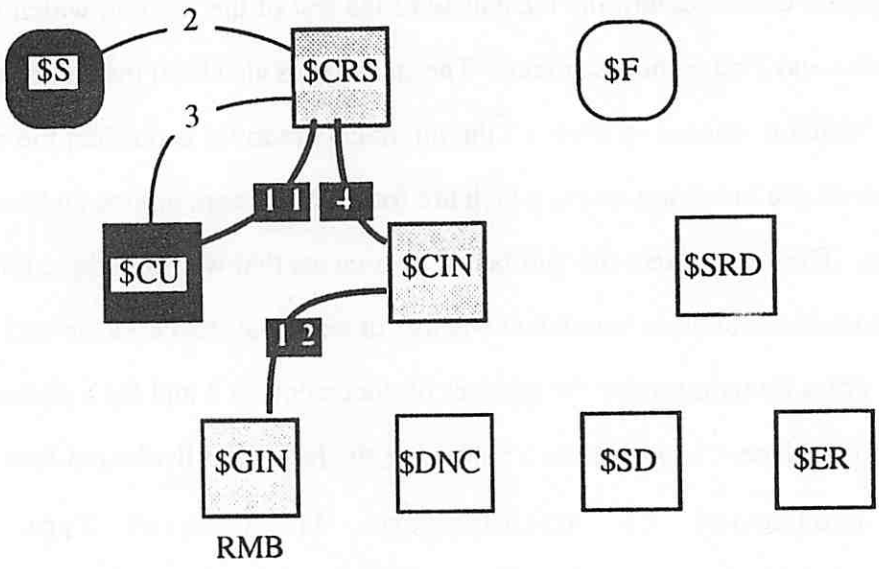

Figure 6.6: New portion of CE Plan accomplished. CE rules 4, 11, and 12 fired to move the system to the new state.

The RMB now responds to the fact that a new session record was posted without an initial need. This happens because when the user model built and installed the new session record, it had on its list of interested experts for this action the RMB, so it sent a message to the RMB. This message was not cleared during any of the previous cycles because the RMB was not active. The response is to send a message to the IM asking it to ask the user how he would like to enter his query. The actual message sent is:

```
(IPM :Message-No 5 :Msg-Id RMB-Get-Query-Entry-Form
:Choices 1 :Value-Type Choice-List :Values (RMB-
Text RMB-Document RMB-Simple-Boolean))
```

Three choices are available, and these are presented in figure 6.7. Had the user been a system expert a fourth choice, a complex Boolean query, would have been made available. In this case, the user decides to enter his query as text. One thing to note in the previous message is the value for `:Choices`. In previous messages, this field had the value `Any`, indicating that the user could choose all of the choices; in the recent message, this value is `1` indicating that the user can only choose one.

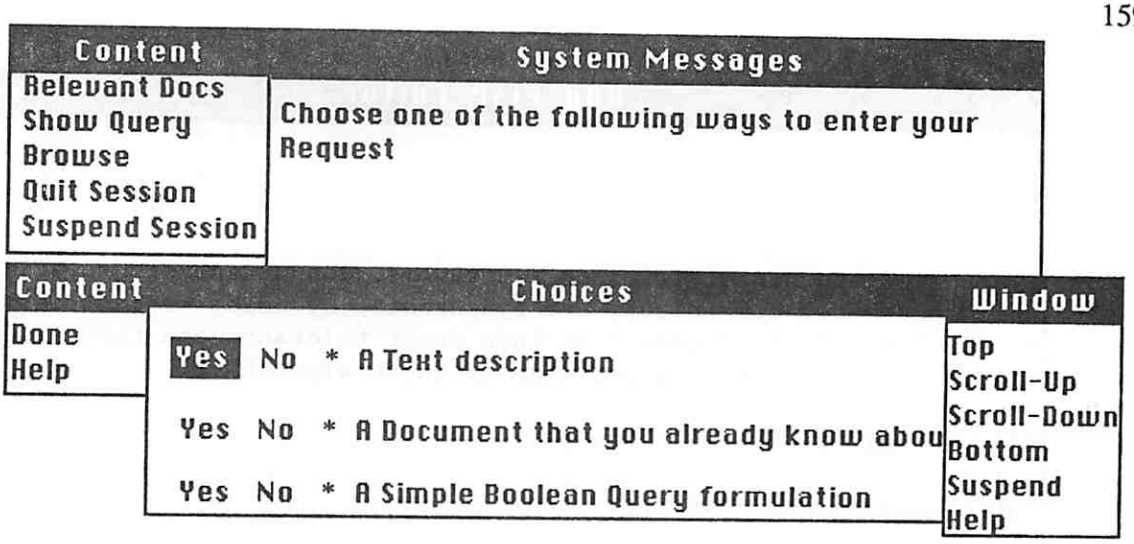

Figure 6.7: System asks the user for the kind of input form to initially specify his query.

6.3.1.10 Cycle 27

The RMB responds to the user's selection of input form by sending the IM a message to display that form. The IM interprets the message by generating a file with the name `NOVICE.QUERY` and copying the appropriate form into it from the file `TEXT.FORM`. The IM then transfers control to the VAX Lisp Editor with the query file as input. An editor window appears with the file consisting of an empty form, and the user types in his query. This is shown in figure 6.8. When the user is done with entering his query, he saves it back into the file and exits the editor, returning control to the interface manager. The IM sends back the message that the RMB sent to it with the `:Values` field filled with the query.

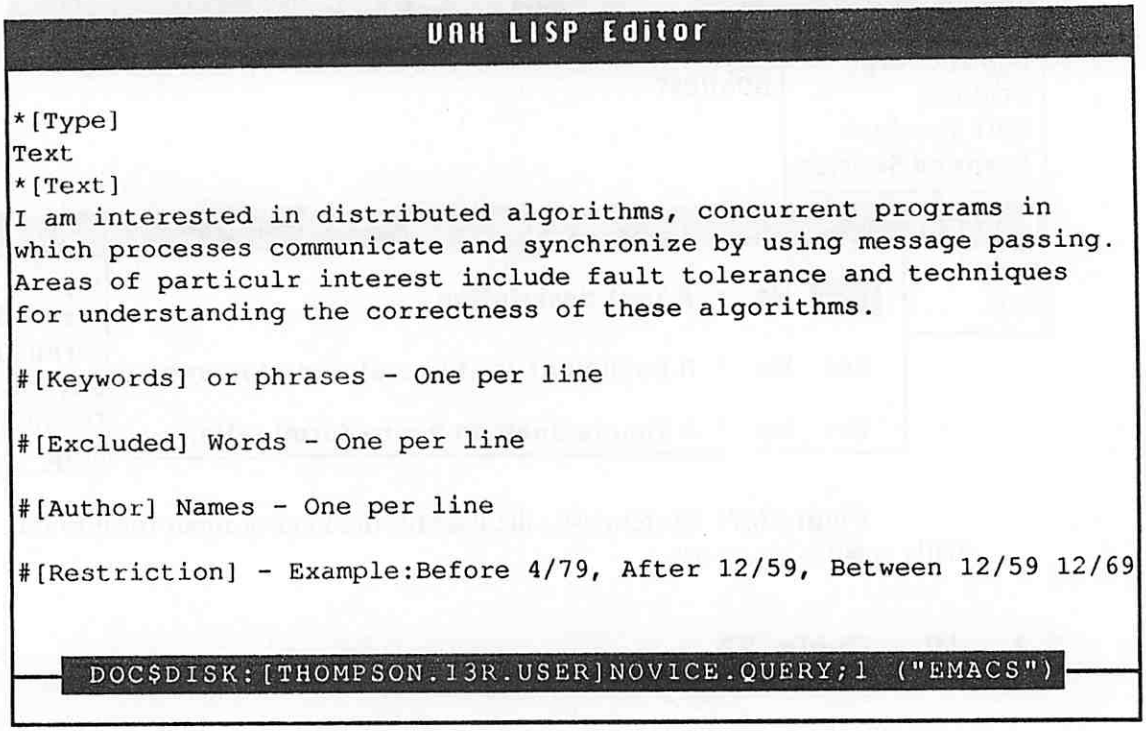

Figure 6.8: The user has entered his query in a free text format.

6.3.1.11 Cycle 28

At this point, the RMB processes the query by indexing it (note this query is used as the example for indexing in chapter two), and putting on the STM place `*Term-Weights*` under the property `Representations`; any phrases that have been extracted from the query are processed and put on the property `Tuples` in the form of a list of pairs or triples of term numbers. The RMB then signals that the initial request model has been formed.

6.3.1.12 Cycle 29

The indication that the RMB gives that the initial request model has obtained from the user causes the CE to mark the state $GIN complete and move the system back to the $CIN state. Since $CIN has another state, $DNC (develop need context), that has not been completed, it goes to that state. Figure 6.9 shows the state of the CE's plan.

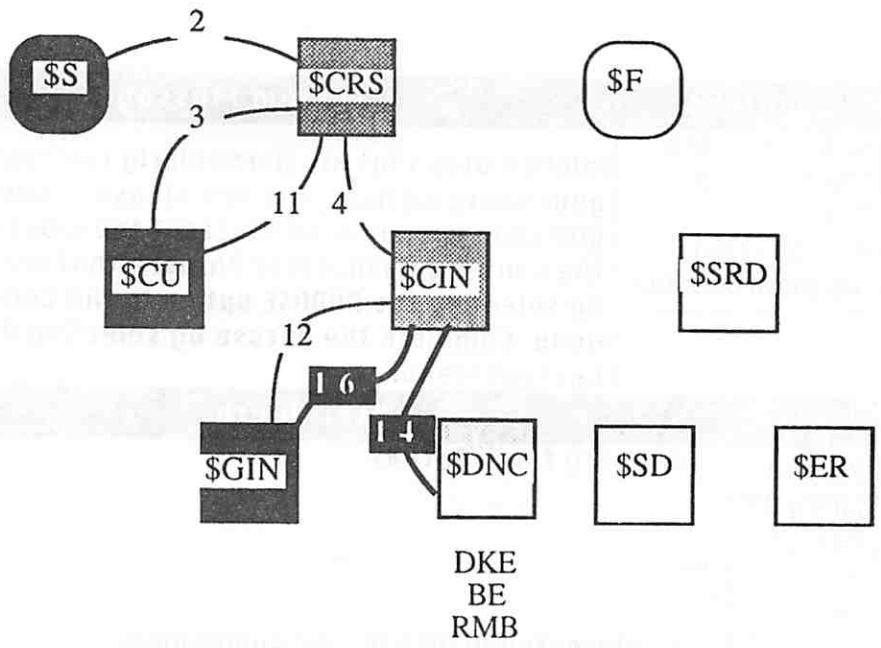

Figure 6.9: The CE's plan after CE operation in cycle 29.

This also marks the first time during the operation of the system that there is more than one expert on the priority list. The priority list has both the domain knowledge expert (DKE) and the RMB on it. The DKE will look for more concepts that might be related to those already in the request model and the RMB will take any concepts approved by the user and place them in the request model.

The first activity on the part of the DKE is to have the user pick out important phrases from the initial request. It posts a message on the *IM-TO* place that tells the IM to redisplay the query so that the user can highlight important words or phrases. This activity is show in figure 6.10.

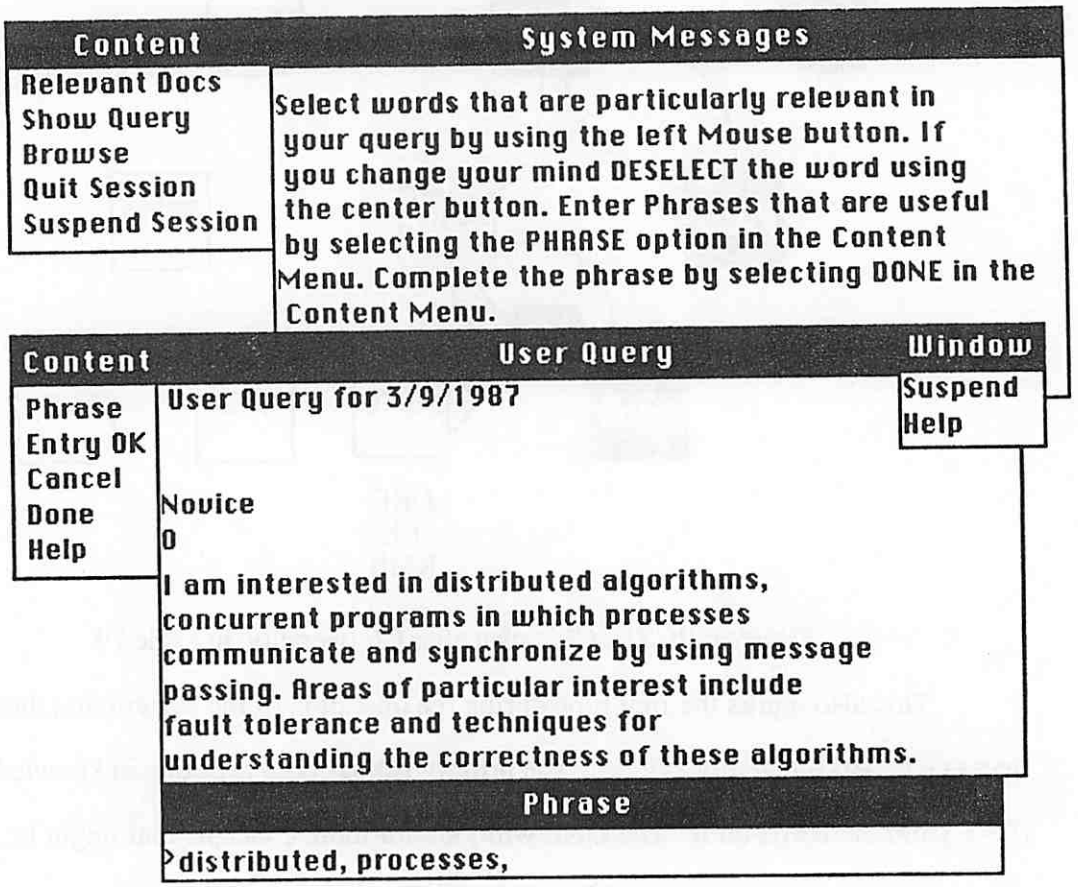

Figure 6.10: User selecting phrases and important words.

This is one place where the stereotypes have limited the user's options. If the user were a system expert, he would have more relationships to choose from. In this session, the user highlights the phrases "fault tolerance," and "message passing." He also decides that "concurrent processes" and "distributed processes" should be considered phrases by selecting their component words.

The RMB does not post a rule on the agenda for execution this cycle.

6.3.1.13 Cycle 140

A number of cycles have passed while the user has been selecting phrases from the query display. These phrases are stored in the record that represents the query in the interface manager. They are stored until the user selects Done from the content menu and then

they are put in the `:values` field of the message record and returned. The evaluation portion of the message is:

```
(((0 nil) (correctness (Phrase fault tolerance 1.0)
synchronize (Phrase message passing 1.0) (Phrase
concurrent processes 1.0) (Phrase distributed pro-
cesses 1.0)))
```

This format is the same as that used when the user evaluates the result of a search. The first item, `(0 nil)`, is the evaluation of the query, and is a result of using the structures for the evaluation of a document to get domain knowledge from the query. This query is considered a document with the number zero. The next part of the list consists of important words and domain knowledge connections, in this case only phrases. The value of 1.0 that is associated with the phrases is a remnant of the initial implementation of the domain code, when consideration was given to associate strengths with the connections in a manner similar to RUBRIC [Tong 83]. They are not used in the current system.

The DKE will take the content of the message, extract the phrases and put them in the user's domain knowledge. In doing this, it checks to see if any of the words are already in the request model, so it can avoid stemming the words and making access to the LTM to get information such as the term number.

The RMB will take the list and convert the phrases into tuples. It does this by retrieving the term numbers from the domain knowledge models.

6.3.1.14 Cycle 141

At this point in the session the DKE will be examining the global domain knowledge to find concepts that are related to the concepts in the request model that have not been checked already. Since this is the initial phase of domain knowledge search, none of them have been, but as they are used, they are marked as having been checked. The DKE looks for synonyms that are related, and finds none.

6.3.1.15 Cycle 142

The DKE looks for phrases that contain words that are in the request model. It finds six and posts a message to the IM with those six for evaluation by the user. They are shown in figure 6.11. The RMB has no activity at this time.

6.3.1.16 Cycle 209

A large number of cycles pass as the user evaluates the phrases presented to him by the interface manager. The user selects `programing techniques` as relevant to his need.

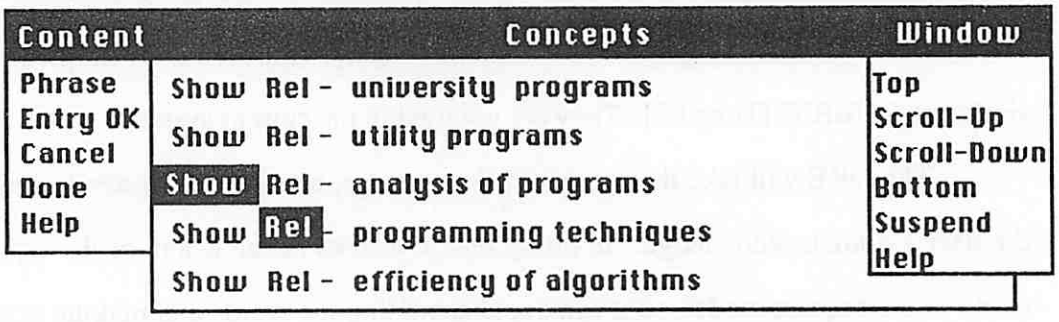

Figure 6.11: Concepts presented for user evaluation.

The interface manager in response to the user selecting `Show` displays the information about the concept `analysis of programs`, which is shown in figure 6.12.

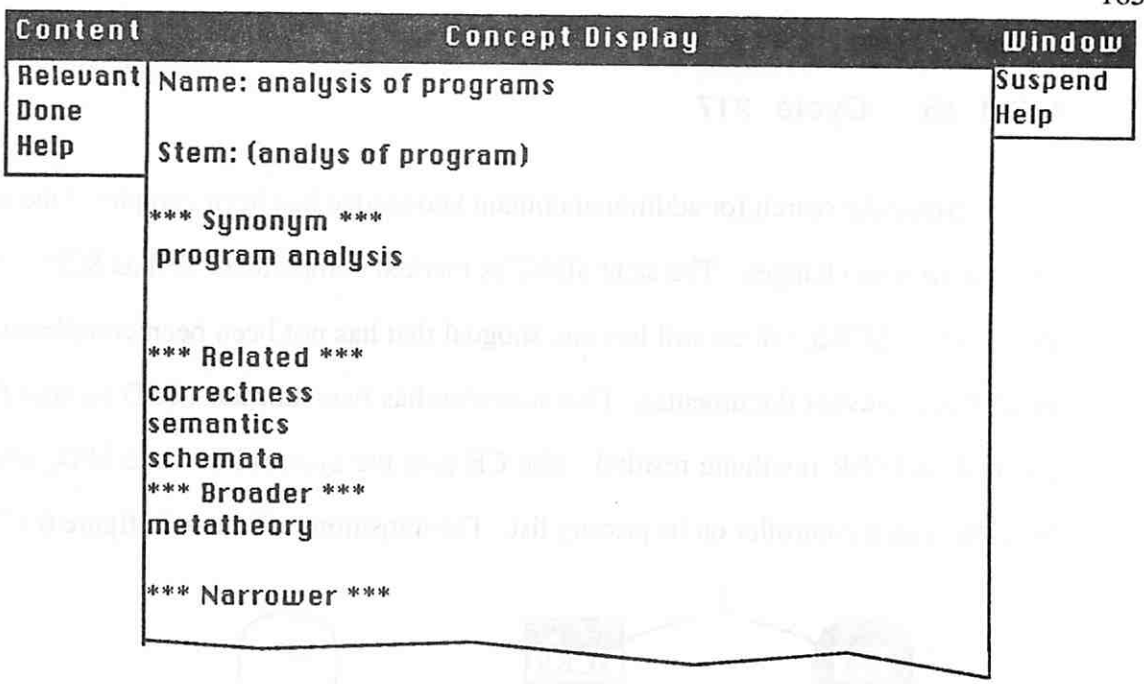

Figure 6.12: Information about `analysis of programs`.

The user decides that `analysis of programs` and `programming techniques` are relevant to his information need. These are transmitted back to the experts when he selects `Done` from the content menu.

The DKE takes the phrases that have been generated by the user and puts them into the user's domain knowledge model.

The RMB takes the phrases that have been selected by the user and puts the component words into the request model and also converts them to tuples, adding them to the tuple list.

6.3.1.17 Cycles 210 – 216

In these next few cycles the DKE looks for domain knowledge using the phrase, synonym, related, broader, and narrower links. In this case, no more additional concepts are found. Since no new information from the user has been obtained, the RMB performs no actions.

6.3.1.18 Cycle 217

Since the search for additional domain knowledge has been completed the focus of the system now changes. The state $DNC is marked complete as well as $CIN. The CE goes back to $CRS, which still has one subgoal that has not been been completed, $SRD (search for relevant documents). This state also has two substates, $SD (search for documents) and $ER (evaluate results). The CE puts the system into state $SD, which has only the search controller on its priority list. The transitions are show in figure 6.13

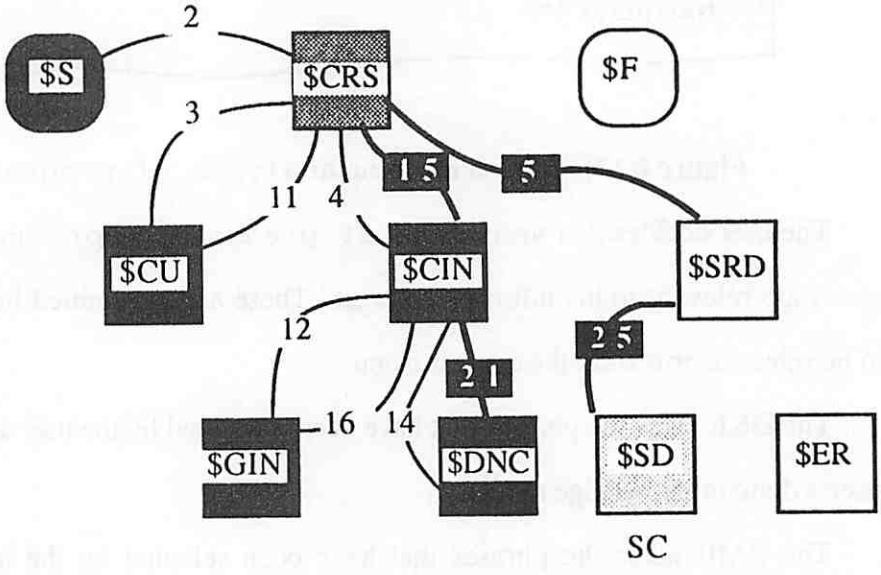

Figure 6.13: CE moves from $DNC to $SD using control expert rules 21, 15, 5 and 25, making the search controller active.

The SC now invokes a search. From the information that the user is precision oriented, the SC chooses to use an initial probabilistic search, which means the term weights are computed using the inverse document frequency weight. In all of the searches, a correction factor is added to increase the score of documents that have terms that are dependent, which are derived from the phrases that the user generates.

The SC generates a file with the search parameters that consist of the kind of search, the number of relevant documents found so far, the term numbers and their collec-

tion frequency, the tuples generated from the phrases, and any documents that have been seen by the user. In the case of this search, the number of relevant documents is zero, and no documents have been seen as yet. When the search is completed, the results are sent to the interface manager for display. The initial display is shown in figure 6.14.

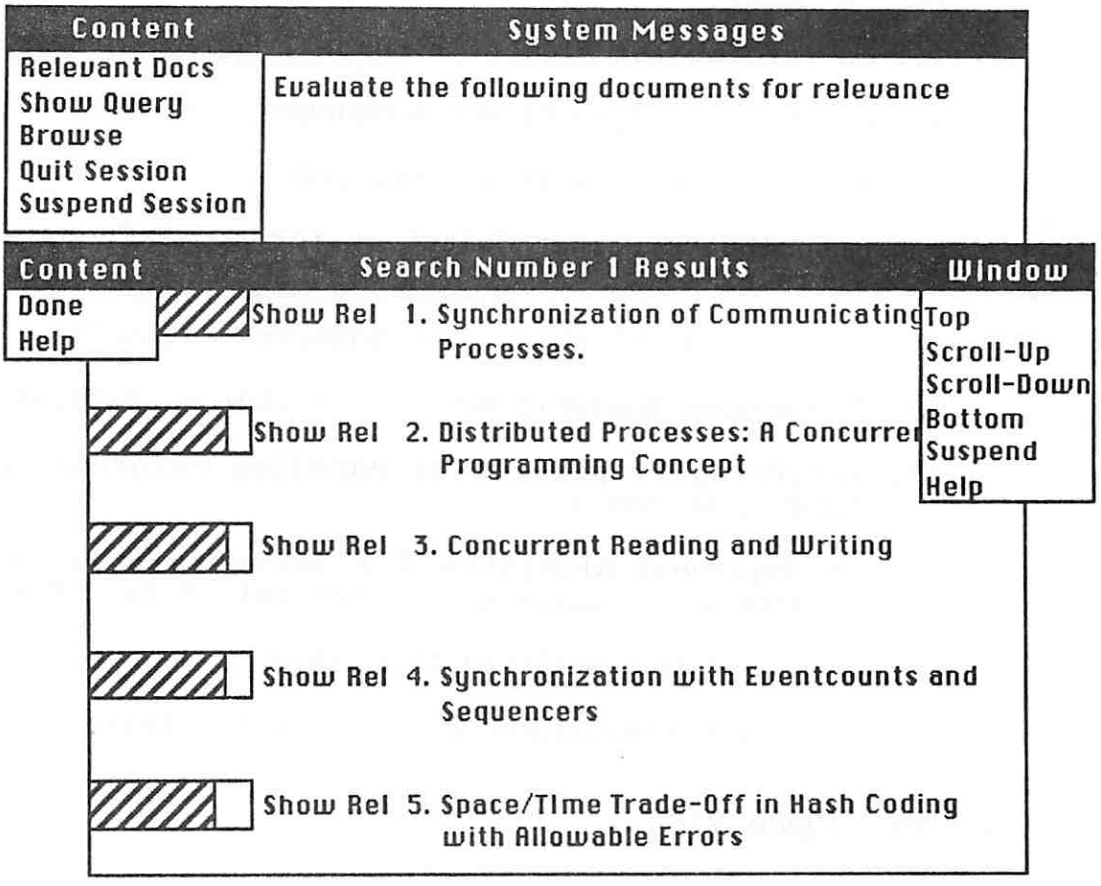

Figure 6.14: Top five documents of initial search.

The bar to the left indicates the relative relevance of the documents. The first document is the basis of the measurement. The length of the bar is simply <document score>/<first document score>. To see the other documents retrieved, the user selects one of the scroll options from the window menu on the right. The rest of the results are:

6. On Computer Enumeration of Finite Topologies

7. Proving Monitors

8. Proving the Correctness of Heuristically Optimized Code

9. On Multiprogramming, Machine Coding, and Computer Organizations

10. The Next 700 Programming Languages

11. An Information Algebra - Phase I Report-Language

12. Programming Systems and Languages 1965-1975

13. Logic and Programming Languages

14. Distributed Packet Switching for Local Computer Networks

15. Secure Communication over Insecure Channels

16. A Computer Analysis Method For Thermal Diffusion

17. Verifying Properties of Parallel Programs: An Axiomatic Approach

18. An Improved Algorithm for Decentralized Extrema-Finding in Circular Configurations of Processes

19. The Expanding World of Computers

20. Exclusive Simulation of Activity in Digital Networks

6.3.1.19 Cycle 218

With the result of the first search done, the CE marks the state $SD as satisfied and moves back to $SRD. Since $SRD has one substate still not satisfied, $ER, it moves the system to that state. The priority list for $ER is BE, RMB, DKE. The browsing expert is first on the priority list since it is given priority to interact with the user. The RMB and DKE are in a passive role; accepting and interpreting input from the user. Figure 6.15 shows the moves of the control expert.

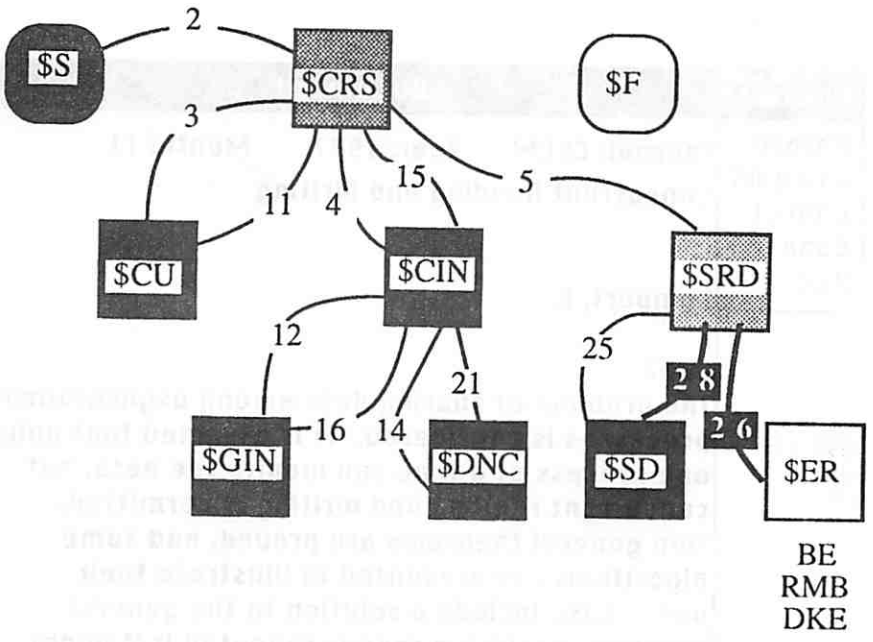

Figure 6.15: The control expert moves the system to state $ER, evaluate results, for evaluation of the search results.

The only activity by the experts during this cycle is the RMB taking the documents and putting them on the `*Doc-Evals*` STM place.

6.3.1.20 Cycles 220 – 785

The primary activity during these cycles of the system is the user evaluating the search results. The same domain knowledge entry options are available to the user that were shown when he was asked to highlight important information in the query. This information is also entered into the user's domain knowledge model and the request model. It is very important at this stage that the user select important words and phrases in the documents that he views. These words are the basis of the relevance feedback process, since these are the only words that are added to the request model. The system does not take all of the words in the relevant documents and add them to the request model. If the user fails to select additional words from the documents that he determines to be relevant, the system will redisplay the search results and prompt him to select some words. Figure 6.16 shows one of the user's judgements.

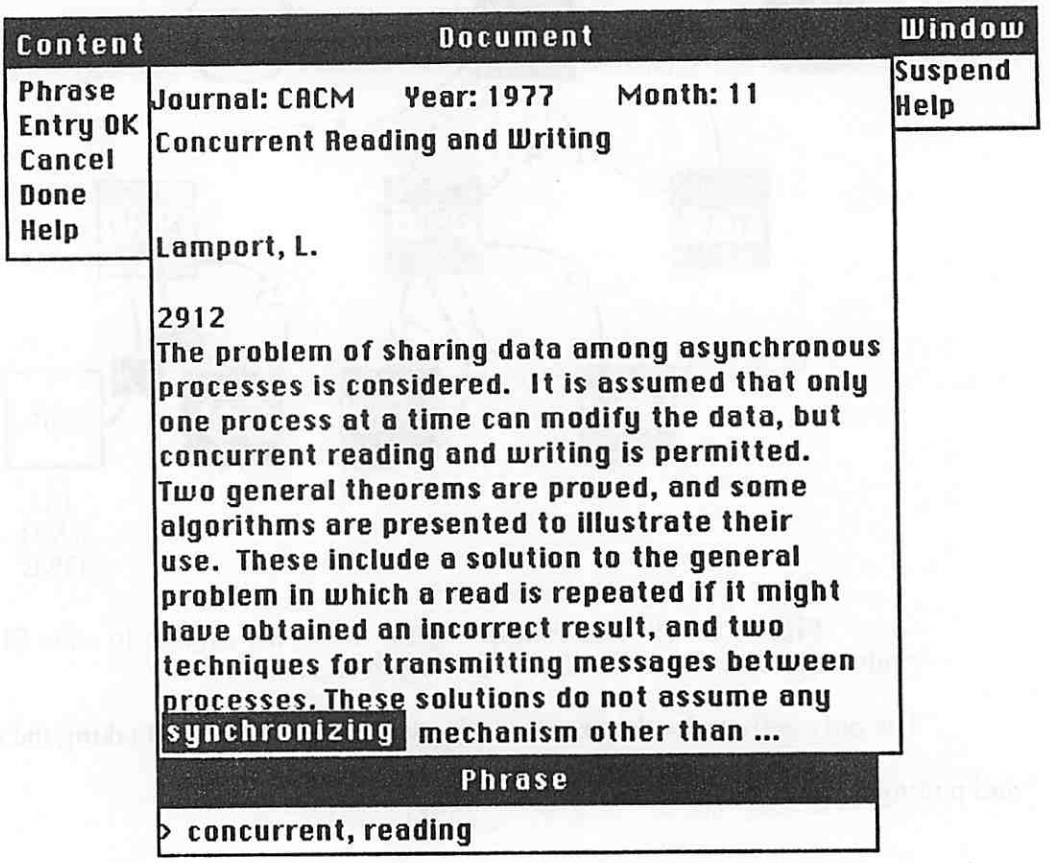

Figure 6.16: User makes relevance judgements of documents terms and phrases in the retrieved documents.

In this search, the user selects four documents as relevant, the first three shown in figure 6.16 and the 17th document on the list. The important words and phrases are deadlock, receivers, senders, synchronization, concurrent writing, concurrent reading, and asynchronous process.

6.3.1.21 Cycle 786

After the user has indicated that he is done with evaluating the search results by selecting `Done` from the content menu, the interface manager sends back the search result message with the user's evaluations.

The RMB adds the selected terms and words that compose the phrases that are new to the request model. The phrases are be converted into tuples.

The DKE puts the phrases, if they are new, into the user's domain knowledge.

6.3.1.22 Cycle 787

The user has indicated that only four of the twenty documents are relevant. This fails to meet the expectation that the user will find five relevant documents, so the CE will, using an exception transition, shift the system back to the state $SD to search for additional documents. Figure 6.17 shows the change in state.

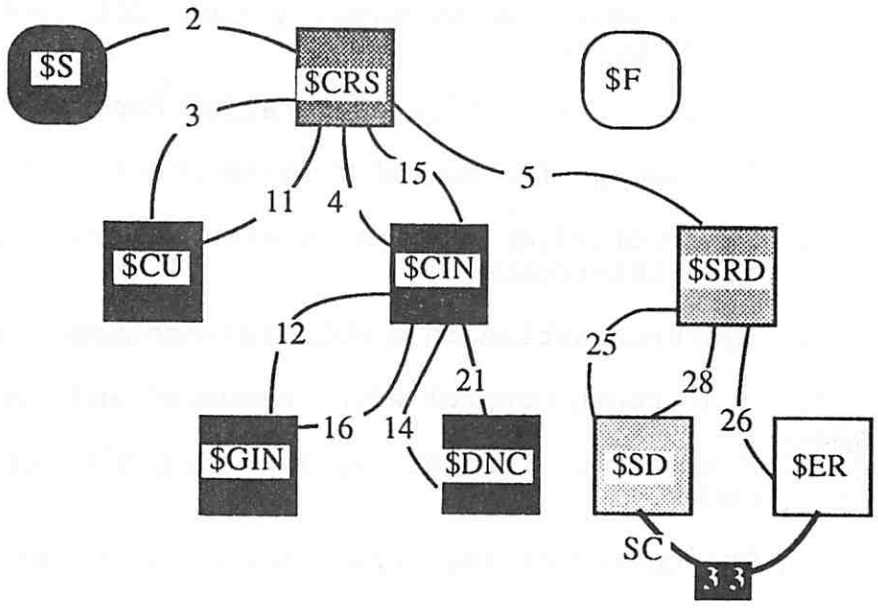

Figure 6.17: The exception transition back to $SD to enable the search controller.

The first thing that the SC does is to evaluate the results of the previous search in order to have a basis for selecting the next search. The first search had a precision of 20 percent (4 out of 20 documents).

6.3.1.23 Cycle 788

The SC now fires of the second search. Since the previous search had a precision level greater that 15 percent the SC chooses to use the same search method. However, the request model has been refined by the addition of new terms and phrases and the search

method can use the full probabilistic weight (equation 2.4 in chapter two) rather than the idf estimate (log(N) - log(n) +1). The results of this search are the following documents.

1. The Executive System Implemented as a Finite-State Automation
2. Three Criteria for Designing Computing Systems to Facilitate Debugging
3. A Large Semaphore Based Operating System
4. An Alternative to Event Queues for Synchronization in Monitors
5. Formal Verification of Parallel Programs
6. A language for Formal Problem Specification
7. Synchronizing Processors with Memory-Content-Generated Interrupts.
8. Synchronization in a Parallel-Accessed Data Base
9. Concurrent Control with "Readers" and "Writers"
10. Signature Simulation and Certain Cryptographic Codes.
11. The design of the Venus Operating System
12. A New Solution of Dijkstra's Concurrent Programming Problem
13. Productivity of Multiprogrammed Computers - Progress in Developing an Analytic Prediction Method
14. Internal and Tape Sorting Using the Replacement-Selection Technique
15. Comments on Prevention of System Deadlocks
16. A Modular Computer Sharing System
17. Monitors: An Operating System Structuring Concept
18. PUFFT-The Purdue University Fast FORTRAN Translator
19. On-the-Fly Garbage Collection: An exercise in Cooperation
20. A Note on Data Base Deadlocks

6.3.1.24 Cycle 789

Since the SC has performed its second search the CE moves the system to the $ER state; figure 6.18 shows this.

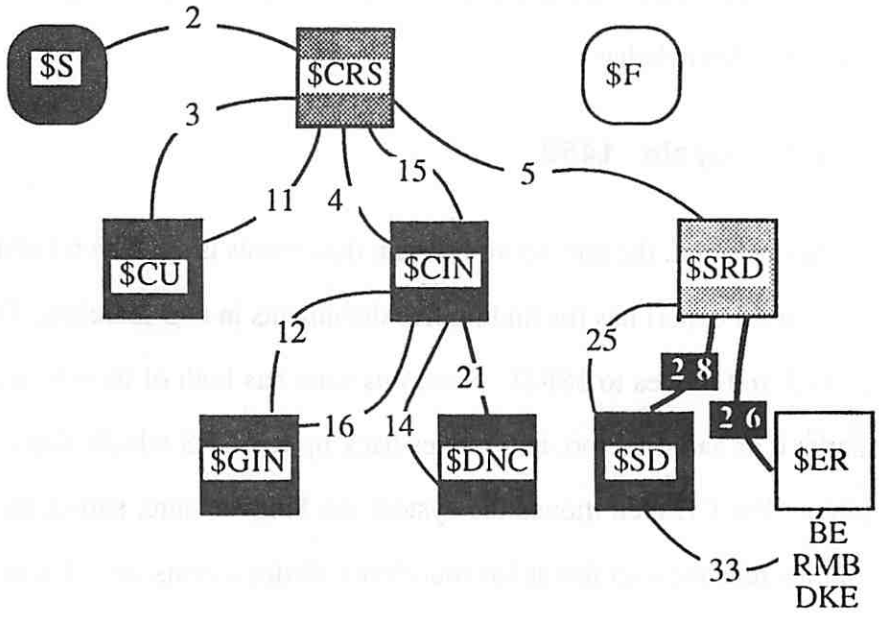

Figure 6.18: CE moving back to $ER

The RMB takes the documents retrieved and puts them into the document evaluations.

6.3.1.25 Cycles 790 – 1448

During this time the user is evaluating the search results. The user finds two more documents, 5, and 6, that he feels are relevant. From these two documents, the user selects two phrases, parallel programs and communicating parallel processes as important concepts.

6.3.1.26 Cycle 1449

As before, in cycle 786, the RMB puts the component words of the phrases and any selected terms into the request model, and the DKE puts any new phrases into the user's domain knowledge.

6.3.1.27 Cycle 1450

At this point, the number of relevant documents is six which fulfills the expectation that the control expert has for finding five documents in two searches. The CE marks $ER as satisfied and moves to $SRD. Now, this state has both of its substates satisfied so the CE marks it as satisfied too, and moves back up to $CRS which also is recognized to be complete. The CE then moves the system the $Finish state, shown in figure 6.19. The system then tells the user that it has found enough documents, and that he may continue the session at a later date. The system writes out the user model with the new session record and the session comes to an end.

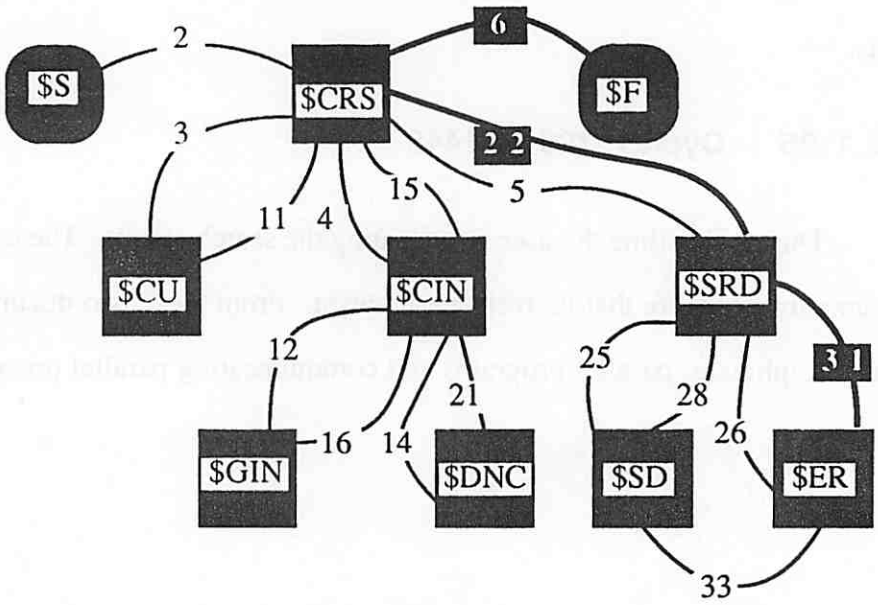

Figure 6.19: The CE moves the system to the $Finish state.

There are a number of behaviors not observed in this first scenario. If, after the two searches, the user had failed to find the expected number of relevant documents, the CE would have recognized the need to do more work in developing the query. To accomplish this, the CE would have caused the states $CIN and $DNC to be unsatisfied, and then the system would move back to the $DNC state where the DKE could begin to look for additional domain knowledge. It would use the new concepts added by the user in evaluating the search results as starting points for new spreading activations in the domain knowledge.

This kind of behavior differs from simple backtracking. Simple backtracking can be characterized in the following way. Consider the problem of the Knight's tour, where the object is for the knight chess piece to visit every square on a chessboard. The piece moves until it cannot move further. If it has not visited every square, it backs up a move to try an alternative move. If every move has been exhausted, it backs up two moves and tries alternatives and so forth until it finds a path through every square. The CE returns to a previous state, but does not retract any information other than the fact that a state has been satisfied; it does not throw out the domain knowledge collected from the user the first time it was in the $DNC state. It is a recognition that it does not have enough information.

6.3.2 Scenario Two

Scenario two is variation on scenario one; its purpose is to show the behavior of the system when another one of the exception transitions is taken. In figure 4.13 there is a transition from $SRD to $CIN, which is taken when the system has made the expected number of searches, but has not found the expected number of relevant documents. In order to show the system taking this transition, the selection of relevant documents in the search results is changed slightly. In the first search, only two documents, numbers one and three are evaluated as being relevant. The same phrases and important words are se-

lected from each document. The results of the second search are slightly different, mostly the order of the documents is changed. In search two, only document five is chosen to be relevant.

The effect of this is to lower the precision of the searches to 0.1 and 0.05 respectively for searches one and two. This causes the SC to evaluate them as marginally effective rather than as complete failures. If a search is a complete failure, the SC will try a different search technique the next time. So, for example, if the first search was a probabilistic one and a failure, the next one would be a cluster search. In this case, where two searches in a row were only marginally effective, the SC will choose a cluster search for its third search.

The CE uses rule 30 to go back to the $SRD state which recognizes that the expectation of the number of searches has been met. Once in state $SRD, it recognizes that the expectation of the number of searches has been met, but the expectation on the number of relevant documents has not been met. This causes the CE to first raise the search expectation by one, then to unsatisfy the states $DNC and $CIN, and finally to put the system into the $CIN state. Once in the $CIN state, the CE recognizes that the $DNC state is not satisfied, so it moves the system to the $DNC state, where it can again search for any potentially relevant domain knowledge. This CE activity is summarized in figure 6.20.

Once in the state where it can interact with the user, the DKE begins to look for new domain knowledge and informs the user of what it is happening by the message shown in figure 6.21.

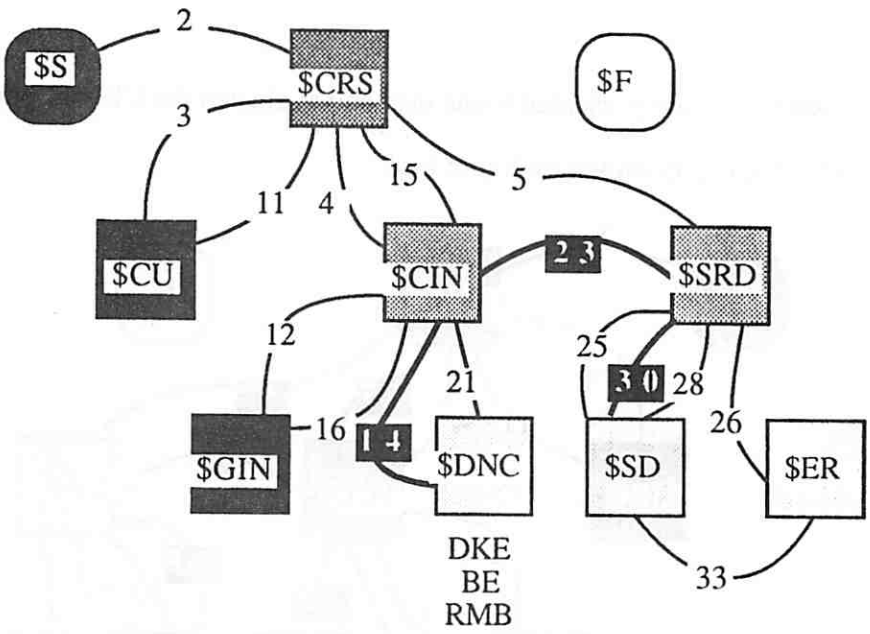

Figure 6.20: CE moving back to the state $DNC to allow the DKE to search the domain knowledge for other concepts.

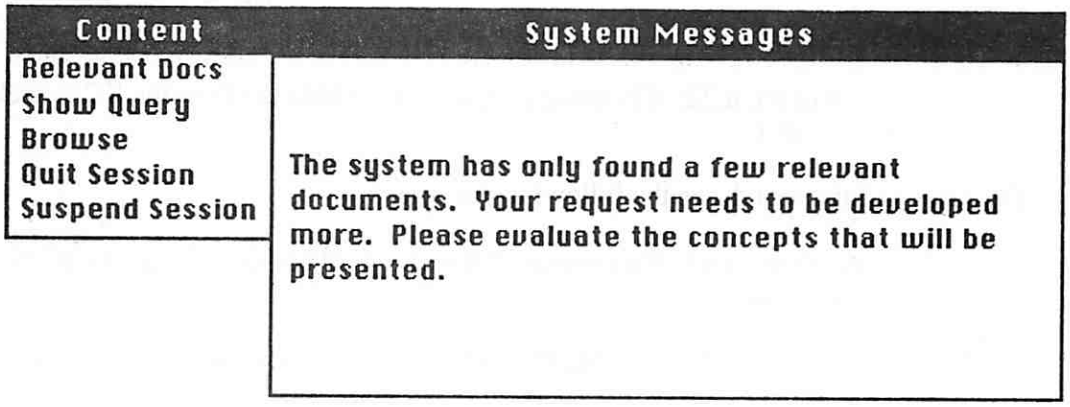

Figure 6.21: Message advising the user on the next activity.

The DKE now searches the domain knowledge using the words that the user has added to the request model from the document that have been evaluated previously. The added terms are: "deadlock," "reading," "writing," "receivers," "senders," "synchronization," and "parallel." This search produces the following new concepts: "process management," "deadlock avoidance," "input output," "parallel algorithms," "parallel processors," and "parallel rewriting systems." The only one of interest is "parallel algorithms."

Once these are presented to the user and evaluated the CE will put the system back into the $SD state, as shown by figure 6.22.

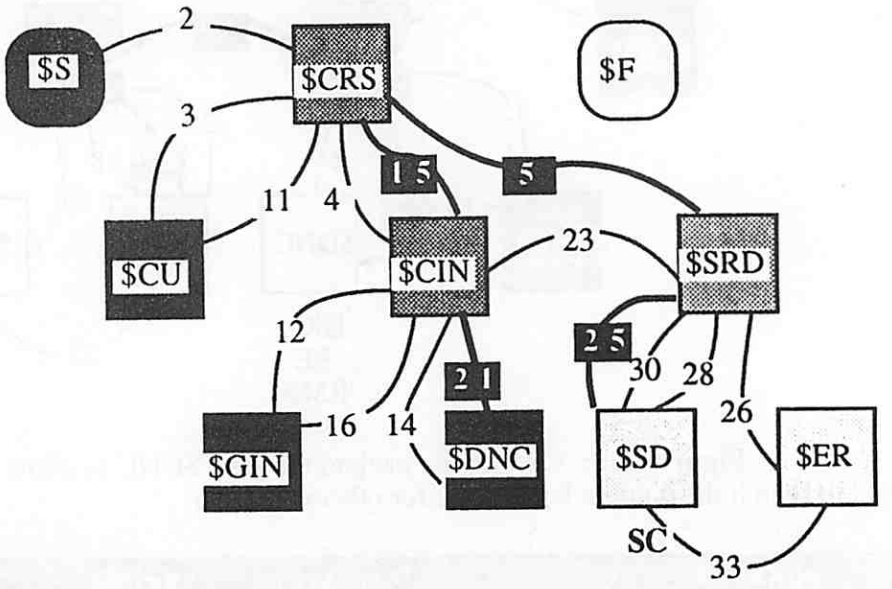

Figure 6.22: CE move system state to $SD to allow the SC to make another search.

The results of the search are the following documents:

1. A General-Purpose Display Processing and Tutorial System
2. A Model for a Multifunctional Teaching System
3. Read-Backward Polyphase Sorting
4. A comparison Between the Polyphase and Oscillating Sort Techniques
5. The ALCOR Illinois 7090/7094 Post Mortem Dump
6. Recursive Solution of a Class of Combinatorial Problems: An Example
7. Polyphase Sorting with Overlapping Rewind
8. Sorting on a Mesh-Connected Parallel Computer
9. Merging with Parallel Processors
10. Mechanization of Tedious Algebra: The Newcomb Operators of Planetary Theory

11. Mechanization of Tedious Algebra: The Coefficients of Theoretical Chemistry

12. Computing Connected Components on Parallel Computers

13. Fast Parallel Sorting Algorithms

14. A Policy driven Scheduler for a Time-Sharing System

15. A Practical Approach to Managing Resources and Avoiding Deadlocks

16. Dynamic Computation of Derivatives

17. A Simple Automatic Derivative Evaluation Program

18. Thoth, A Portable Real-Time Operating System

19. A Multiprogramming Monitor for Small Machines

20. A Language for Describing the Functions of Synchronous Systems

21. SIMULA - An Algol-based Simulation Language

None of the documents are, in the opinion of the user, particularly relevant to his query, so he decides to review the results of the previous searches. This can be done by selecting the partial search results windows, which are generated when the user signifies that he is done in the content menu. Instead of disappearing like the rest of the windows, they are redrawn as shown in figure 6.23. By placing the pointer on the single line and clicking the left mouse button, the window reappears as it was originally shown (figure 6.9).

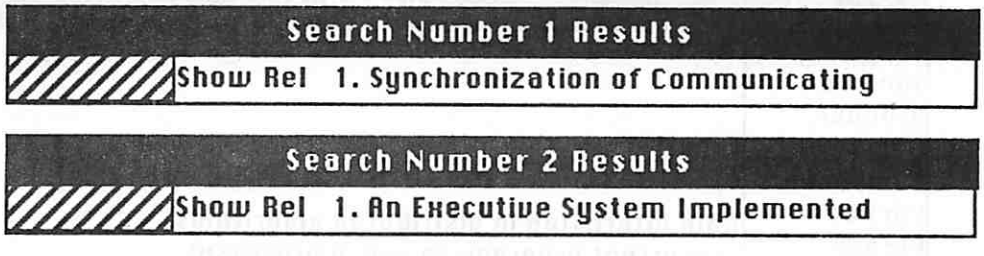

Figure 6.23: Search results windows after the user is done with the second search.

The user decides that several more of the documents from search one and search two are relevant. In so doing, the expectation on the number of relevant documents is met,

as well as the expectation on the number of searches. This will cause the CE to recognize that the session is over and will put the system into the $Finish state as before (figure 6.19).

6.3.3 Scenario Three

The third scenario demonstrates the extra choices available when the user is a system and domain expert. In order to simulate expertise the user's domain knowledge has been expanded in the area of operating systems. This scenario emphasizes the differences in the operation from the basic novice operation, so many of the details presented in the previous scenario are omitted, particularly the cycle by cycle operation of the system.

After the user has entered his name, the system, as before, asks him questions about the nature of his experience in the domain, computer and IR systems, and the type of search. In this case, he is familiar with the domain, has used an IR service before, and is interested in a recall oriented search. This sets up the control expert's expectations, which are 20 relevant documents in 2 searches.

After entering his query (the same one as in the first scenario), the system presents it to him for elaboration (figure 6.24).

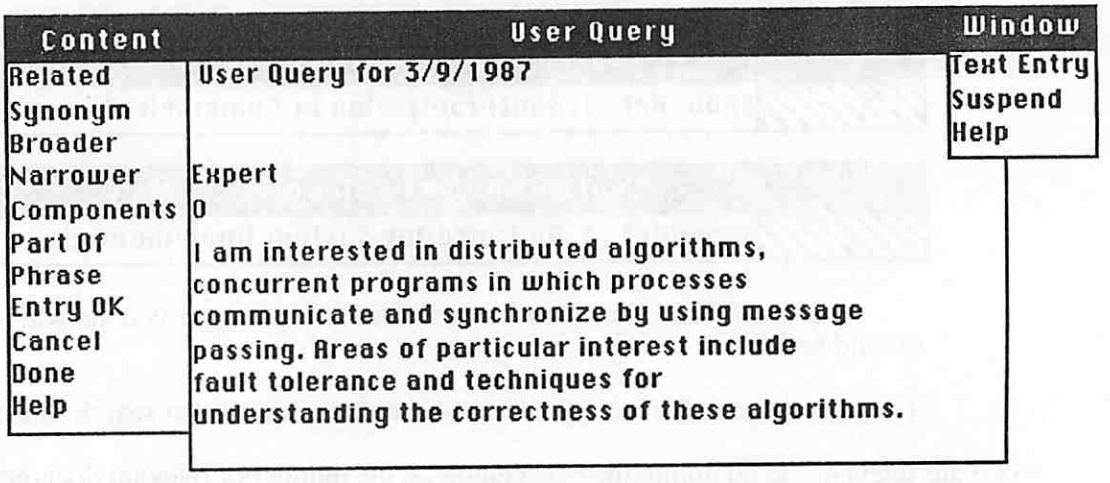

Figure 6.24: Query elaboration with more choices for the expert user.

Besides indicating *phrases*, the user can indicate *broader*, *narrower*, *synonym*, *related*, *component*, and *part-of* relationships. Furthermore, the expert user is not limited to the words in the original query. If he should think of words that might apply that are not in the original query, he can add them by selecting the `Text Entry` selection from the window menu on the right. Figure 6.25 shows the user entering two phrases, `distributed algorithms` and `parallel algorithms`, that are *related* to each other.

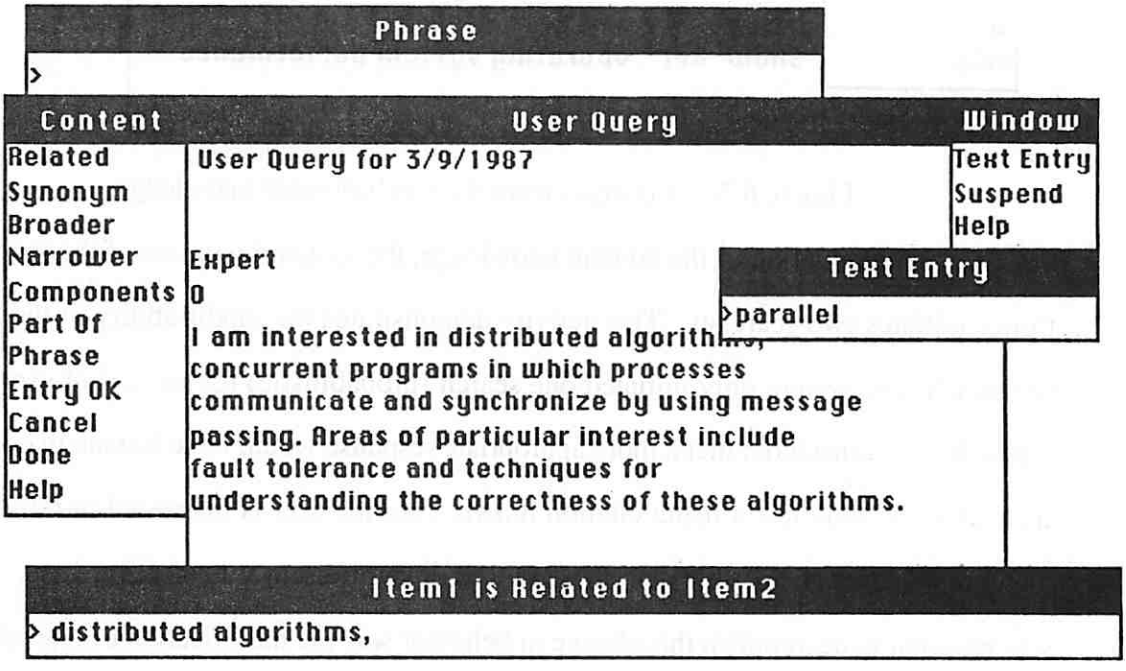

Figure 6.25: Domain knowledge entry by a domain and system expert.

As before, the system shifts to a state where the DKE presents candidate concepts to the user for approval and addition to the request model. Figure 6.26 shows the concepts that are taken from the user's domain knowledge model. The DKE searching the global DK will retrieve the same concepts as shown in figure 6.5 in the first scenario.

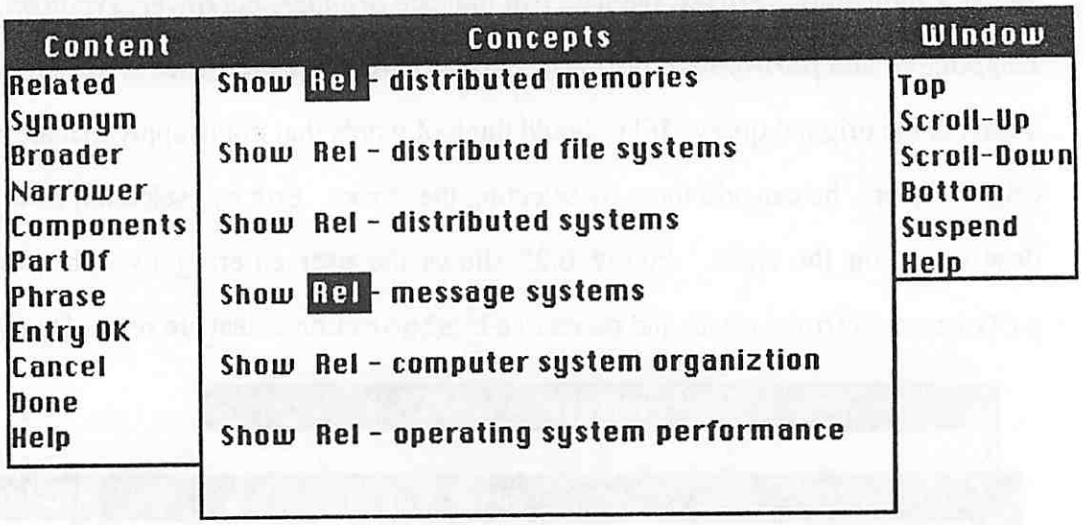

Figure 6.26: Concepts from the user's domain knowledge.

After evaluation of the domain knowledge, the system by means of the search controller initiates two searches. This activity demonstrates the modifiability of the system. Originally, the system only initiated one search (probabilistic) for this set of user stereotypes. It was considered that a more appropriate response would be to initiate two searches (probabilistic and cluster using citation links), since the user is interested in retrieving as many documents as possible, and these two retrieve different sets of documents. All that was required to accomplish this change in behavior was the modification of one rule in the search controller. The results are shown in figure 6.27.

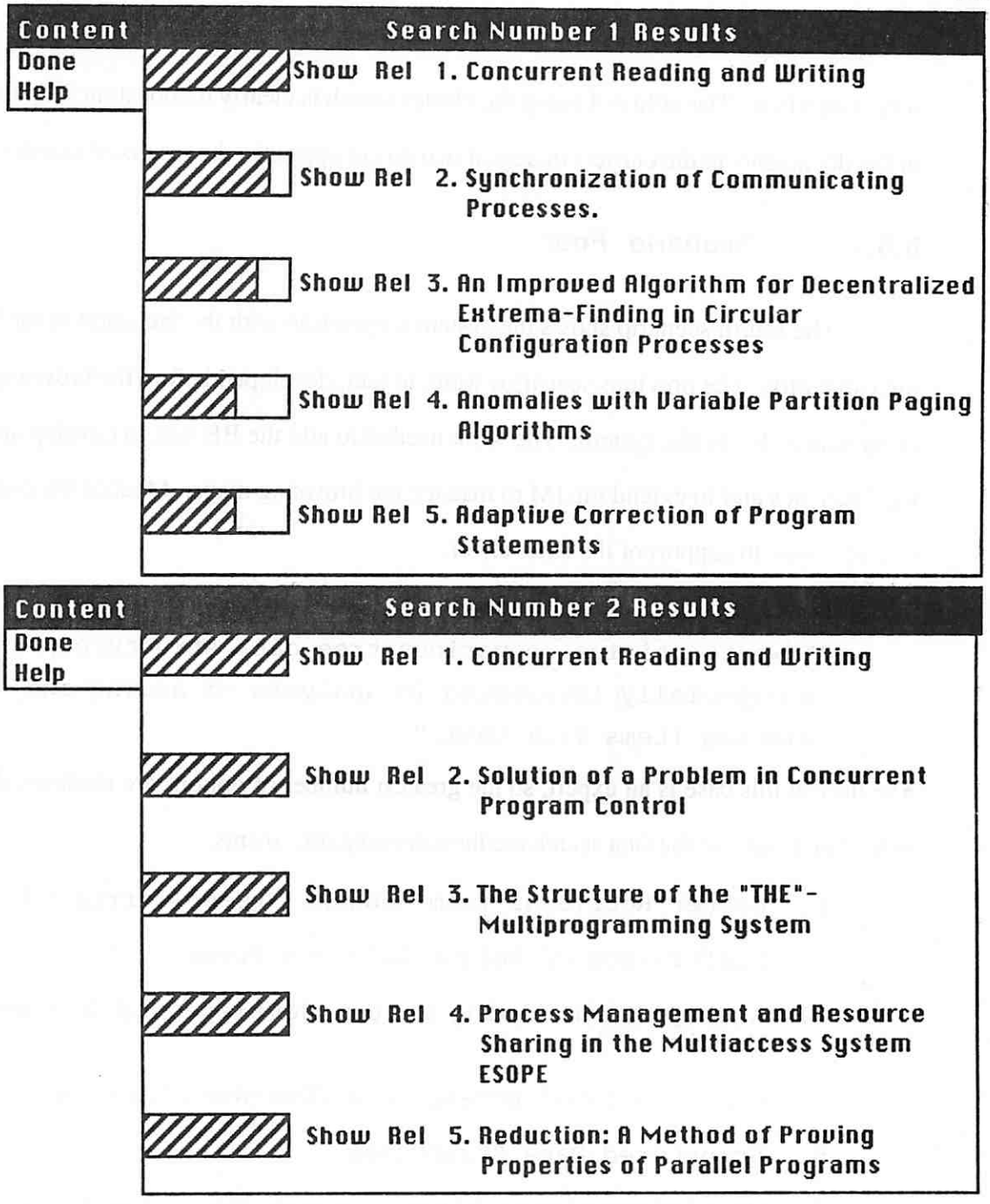

Figure 6.27: Results of the first two searches (window menus not shown).

Notice that documents 1 through 5 of search one have the same relative relevance, this is because they are members of a single cluster. Document 2 in search one is document

6 in search two. The utility of using the cluster search is clearly demonstrated, since many of the documents in the clusters of search two do not appear in the results of search one.

6.3.4 Scenario Four

The fourth scenario shows the system's operation with the inclusion of the browsing capability. The previous scenarios were, in fact, developed before the browsing capability was added to the system. The work needed to add the BE was to develop and code the heuristics and to extend the IM to manage the browsing maps. Most of the coding effort was done in support of the latter effort.

In this scenario, a different query is used.

"I want articles on various tree data structures, I am especially interested in analysis of adding and deleting items from them."

The user in this case is an expert, so the greatest number of options are made available to him. The results of the first search are the following documents.

1. Faster Retrieval from Context Trees (Corrigendum)
2. Performance of Height Balanced Trees
3. A Compiler-Building System Developed by Brooker and Morris
4. Conversational Access to a 2048-Word Machine
5. Structured Data Structures
6. Multidimensional Binary Search Trees Used for Associative Searching
7. A Comparison of Simulation Event List Algorithms
8. Cellular Arrays for the solution of Graph Problems
9. COKO III: The Cooper-Koz Chess Program
10. Fen-An Axiomatic Basis for Program Semantics
11. Abstraction Mechanisms in CLU

12. Blocks-A New Data Type in SNOBOL4
13. Accommodating Standards and Identification of Programming Languages
14. Optimizing Binary Trees Grown With a Sorting Algorithm
15. A Relational Model of Data for Large Shared Data Banks
16. File Structures Using Hashing Functions
17. Syntax-Directed Documentation for PL 360
18. Algebraic Simplification: A Guide for the Perplexed
19. The SMART Automatic Document Retrieval System - An Illustration
20. A Record and File Partitioning Model

At this point in the session, the experienced user is allowed to browse; a novice user is allowed to browse only after the system has failed to retrieve its expected number of documents in the expected number of searches. Browsing is signaled by the selection of the Browse option from the Content Menu that is associated with the System Messages window (for example, see fig 6.1). Selection of this option tells the system that the user is done with the search results. The interface manager sends the evaluation of the search to the experts for their use. This causes the RMB to update the request model by adding new terms (if any) from the documents that were marked relevant by the user. Evaluation by the SC of its search performance is delayed until the system enters the $SD or $Finish state. If he has made any connections between terms the DKE will establish them in the user's domain model.

If the user selects browsing before he has marked any of the documents of the search as relevant, the search will initially be categorized as a complete failure. But, if the user subsequently marks documents that were in the search as relevant, the search controller will adjust its evaluation of the success of the search.

The system then tells the user to select a document as the initial point from which to begin. The user may choose a document from a variety of sources. He may select a relevant document by getting to the relevant document list, or may redisplay the result of a particular search. Indication of the starting document is made by selecting Show.

Commencement of browsing is indicated by a message in the system messages window and by the appearance of the neighborhood and context maps on the screen. Figure 6.28, shows the initial configuration of the neighborhood map. In this instance, the user selected document two, "Performance of Height-Balanced Trees" (document number 2889).

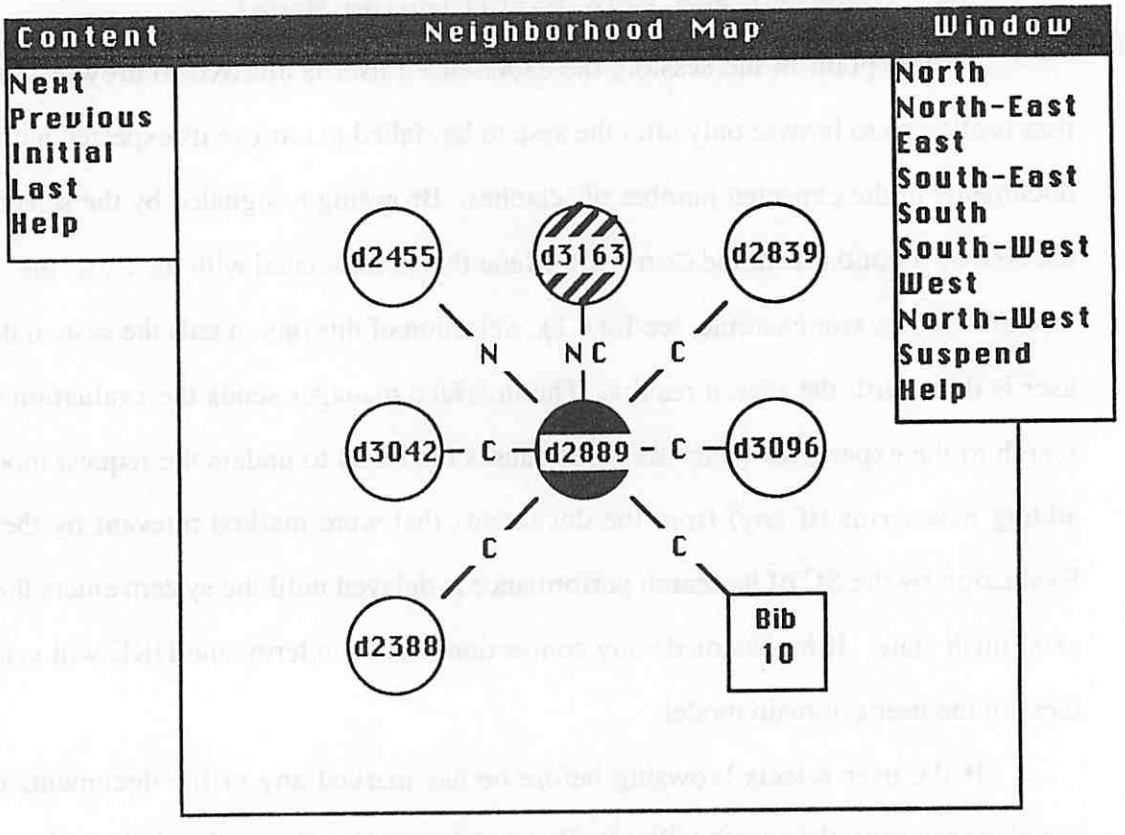

Figure 6.28: Initial display on the Neighborhood Map (the document 2889 and Context window is not shown).

In the center of the map is the document that the user selected as the starting point. The surrounding nodes are documents that the browsing expert has decided are likely to be

interesting, based on its heuristics. One node is marked as the most likely to be interesting, which in this case is document 3163, being selected by having a nearest neighbor and a citation link to the current node. The boxes represent lists of documents that cannot be shown but are related to the current document. The relationship is marked on the link and the number of items that the box represents is in the center of it. The menu to the right labeled `Window` allows the user to scroll around the context map. The content menu selections allow the user to trace the path that he has chosen. The selections mean:

`Next` – select the recommended node for viewing.

`Previous` – go to the previously viewed node .

`Initial` – go to the first node displayed.

`Last` – go to the last viewed node.

Here, the display shows that there are a number of documents that the system judges as likely to be interesting, and that there are more documents connected than can be shown.

When the user selects one of the nodes with the pointing device, the document that it represents appears on the screen (figure 6.29) and the node is shaded to indicate that it has been visited. When the user indicates that a node is relevant and says that he is done evaluating it, the evaluations are added to the request model and any domain knowledge is added to his domain knowledge model. After a number of documents are judged relevant the system will, depending on the user stereotype, seek to initiate a search. If the user is a system novice, as well as a domain novice, the search controller will automatically be given control to initiate a search. If the user is system expert, the system will indicate, via the system messages window and choice selection window, that it can make a search based on

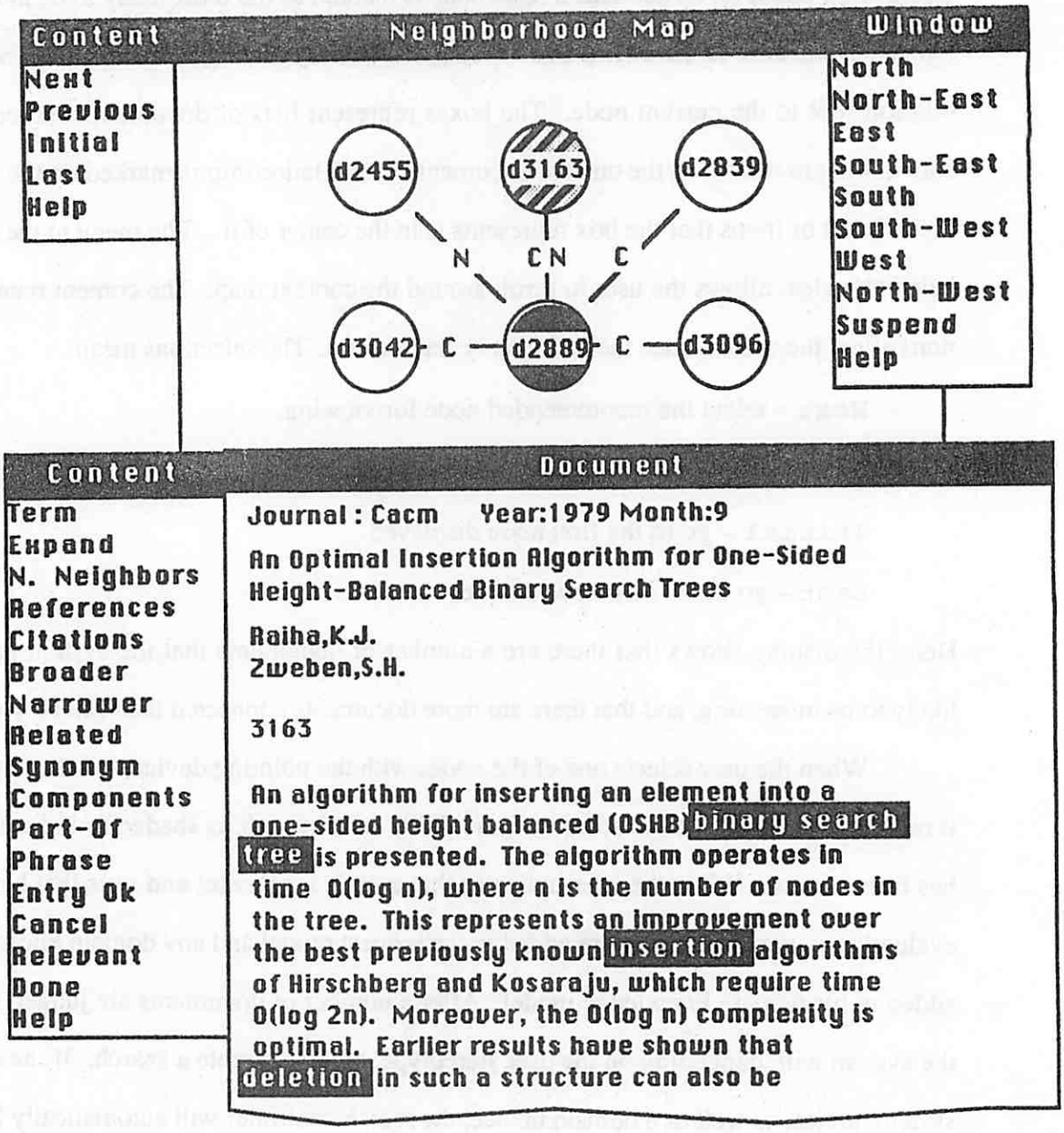

Figure 6.29: User selects a recommended node to view its contents, and selects terms that are particularly relevant or interesting. He also selects the concept "AVL," which is not shown.

the new information and requests the user's permission to do so. The number of new documents judged relevant that causes a search to be performed is determined by the stereotypes set by the UMB.

The neighborhood of a document is not determined unless the user judges it to be relevant or selects the Expand option from the content menu associated with the document text window. If either of these choices is made, the node is redrawn farther away to provide space for its neighborhood. The user selects the Expand option and the map is redrawn showing the neighborhood of the selected document (figure 6.30).

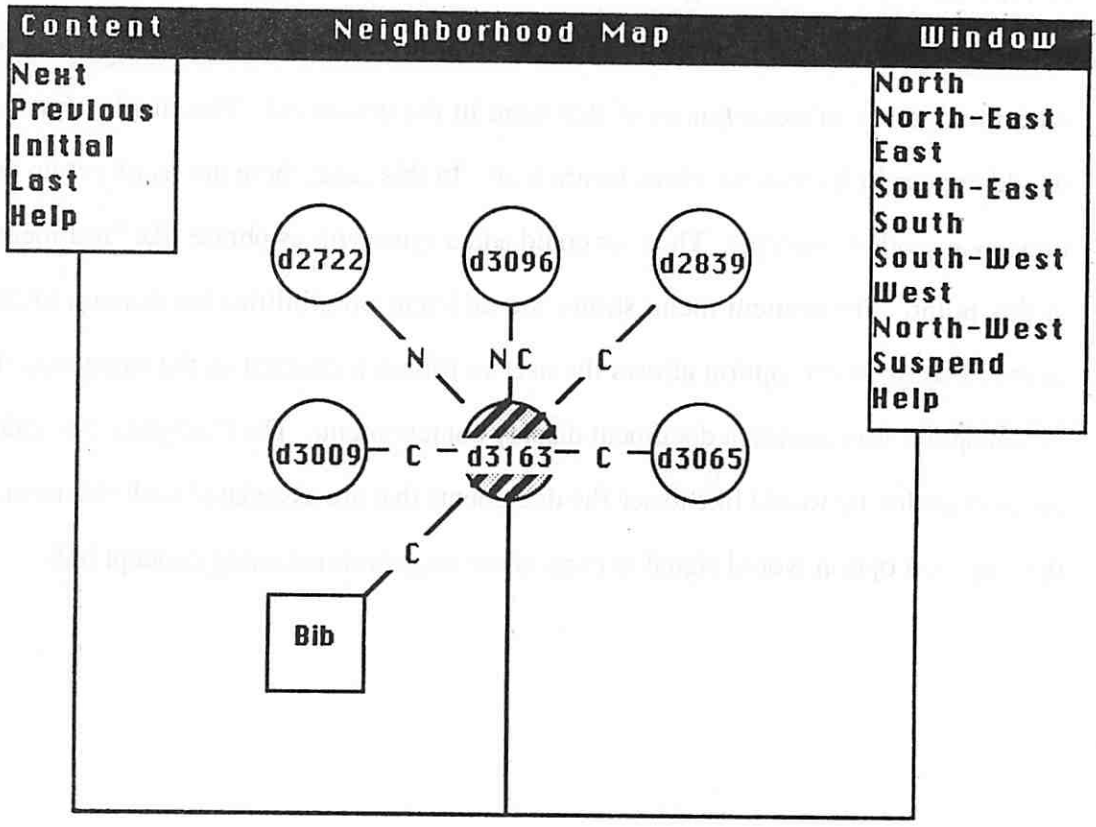

Figure 6.30: Neighborhood Map with expanded document neighborhood.

The user is not restricted to selecting documents that appear on the neighborhood map. He may go back to the current document of interest and select any list of documents associated with it. These are the reference list, the citation list, nearest neighbor list, the author list, or, the journal issue list. Figure 6.31 shows the neighborhood map after the user has examined the recommended node and has decided to examine the other node connected by the nearest neighbor link, marked "N"

The user is not restricted to just moving from document to document, but may also examine concepts. In figure 6.32, the user has decided that the term "multidimensional" is possibly a fruitful path to follow. He selects the `Term` option from the content menu and then the term of interest. This tells the system that he is selecting this as a node to examine and not that it is an interesting term. The node icon with the term's number appears on the neighborhood map and is shaded to indicate that it has been visited. The link is marked with the number of occurrences of that term in the document. The display for the term would appear in its own window, figure 6.20. In this case, there are as of yet no connections to any other concepts. The user could add a synonymous phrase like "n-dimensional" at this point. The content menu shows the different possibilities for domain knowledge entry. The `Select` option allows the user to follow a concept in the same way that the `Term` option was used in a document display content menu. The `Documents` option tells the system that he would like to see the documents that are associated with this term, where the `Expand` option would signal to expand the neighborhood using concept links.

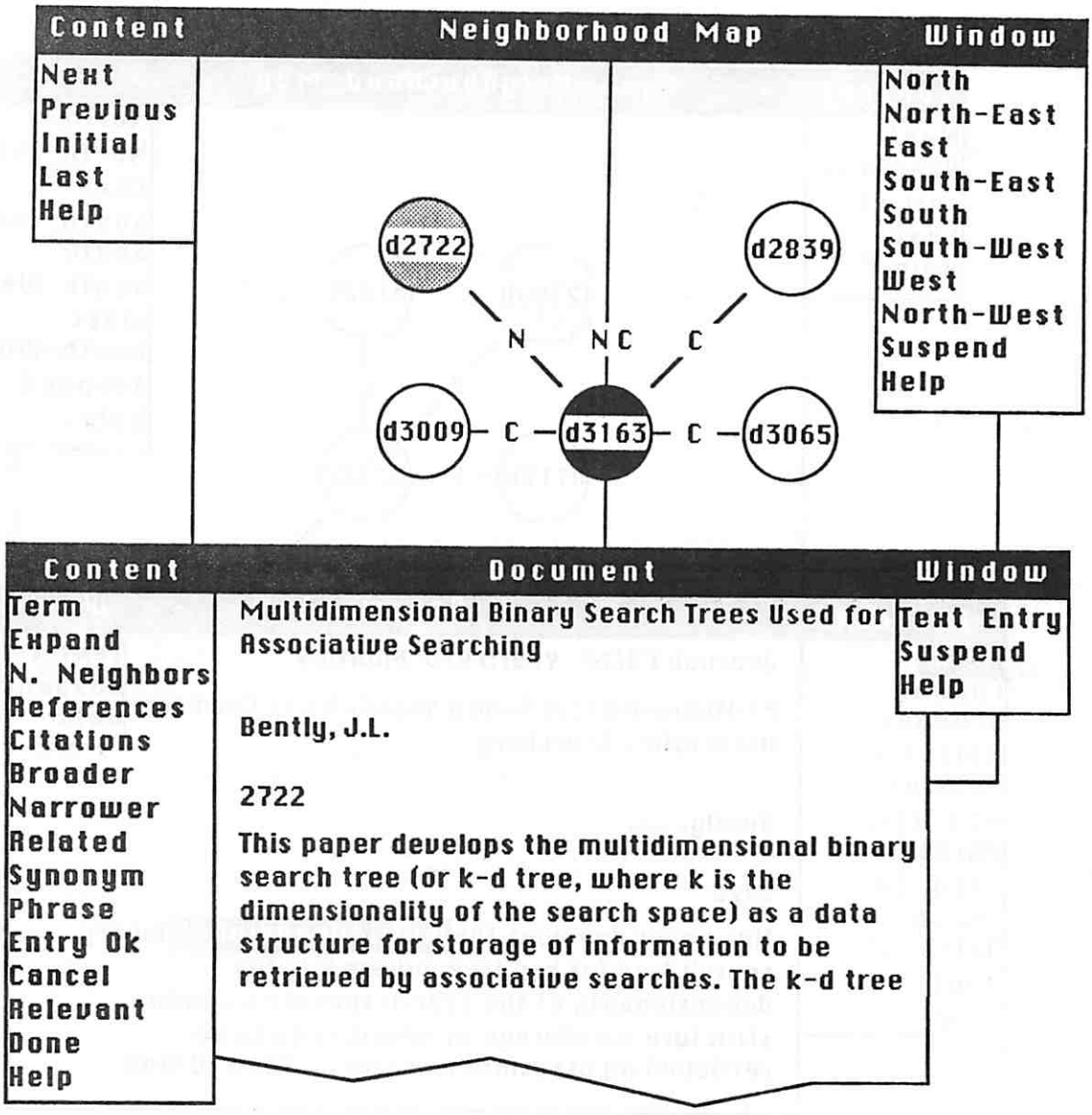

Figure 6.31: User views document 2722 (text is incomplete).

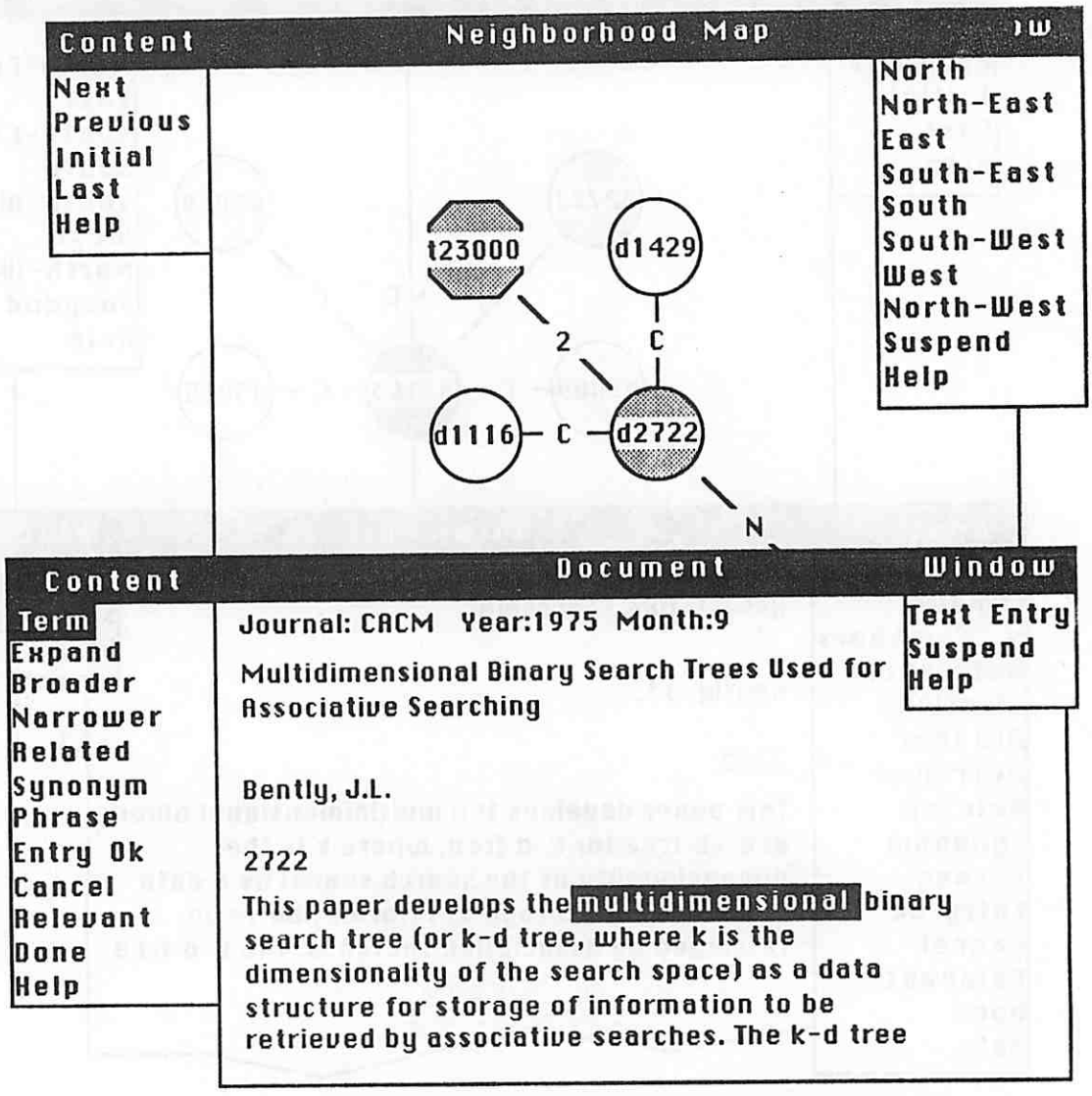

Figure 6.32: User selects a term to examine from document 2722.

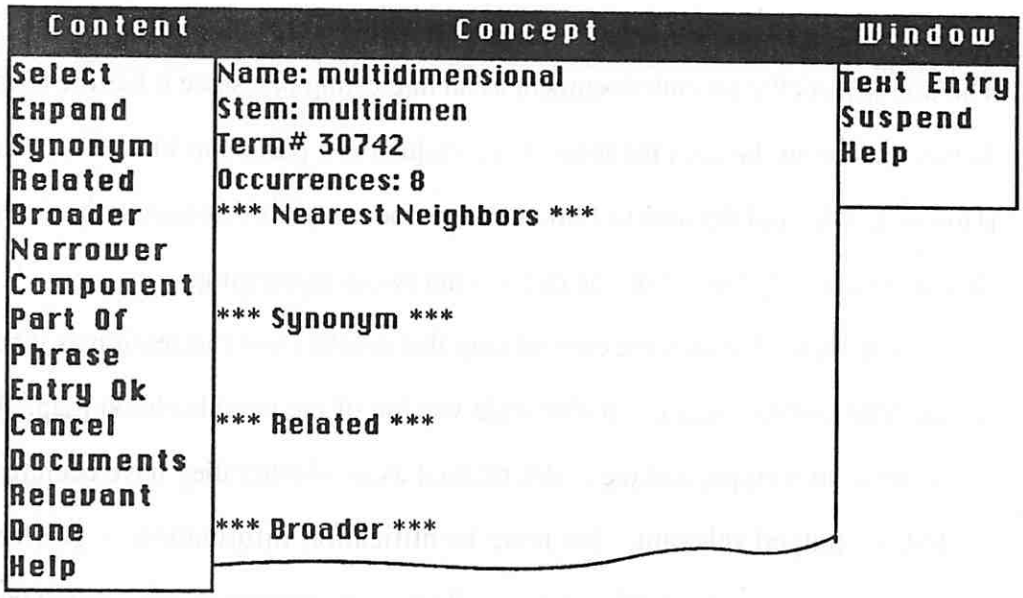

Figure 6.33: Display for the concept `multidimensional`.

Upon selecting the **`Documents`** option (figure 6.34), the browse maps are both expanded and a document list similar to the search results list appears (figure 6.21) with the titles of the documents that have the term "multidimensional" in them.

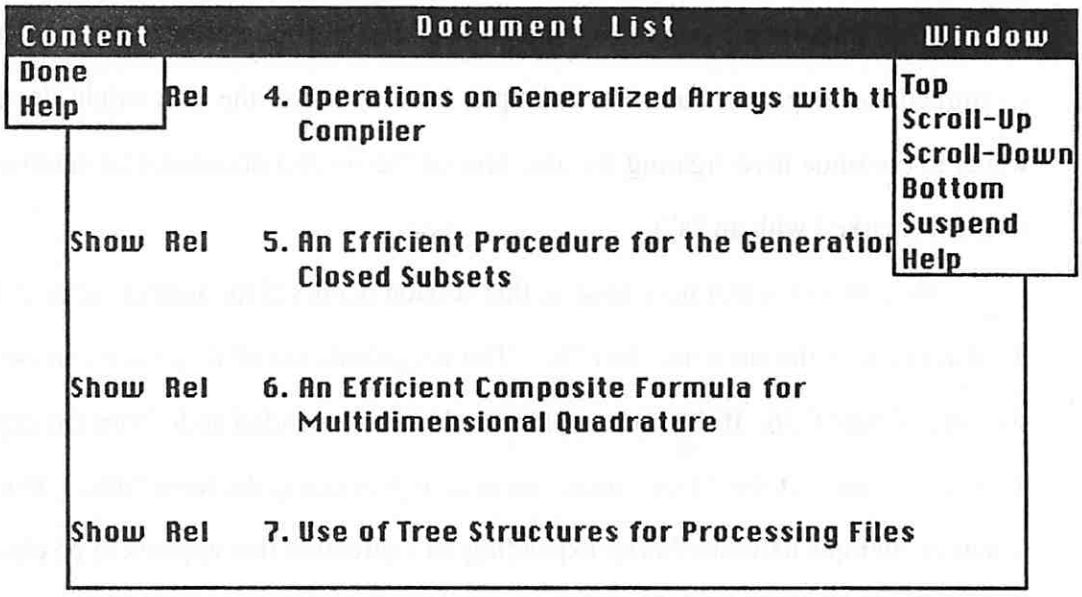

Figure 6.34: User selects **`Documents`** option.

The user selects the seventh document as an interesting one since it has the term "tree" in it. In this document, he sees the term "trie," (which is a particular kind of tree data structure [Horowitz 84]) and decides to find out what documents are connected to it. In the collection, there are only three, but one of the other two is interesting.

Figure 6.35 shows the context map that results from the session as it has developed so far. The context map is smaller scale version of the neighborhood map. The links are still marked as to type, and the nodes marked as to whether they have been recommended, visited, or judged relevant. No node identification information is given since that is available from the neighborhood map. Both maps remain in correspondence with each other, centered on the same node. When the user selects a node in the neighborhood map, the context map is updated at the same time, and vice versa. An example of a context map is figure 6.35. From this map, the user can go to any node by scrolling it into view and selecting it (placing the cursor on it and clicking with the left mouse button). This will cause the neighborhood map to be redrawn with the new node of interest in the center with its immediate neighborhood. For example, in this session the user might decide that he wants to continue investigating the area around the second document he determined to be relevant (marked with an "a").

One problem that may arise in this session occurs if the user decides to expand in the direction of the node marked "b." The neighborhood of this node can be drawn as shown in figure 6.36. If the user would take the recommended node from the expansion of the marked node, it would overwrite the node representing the term "trie." The interface manager attempts to avoid this by expanding in a direction that appears to be clear, such as the location marked "c."

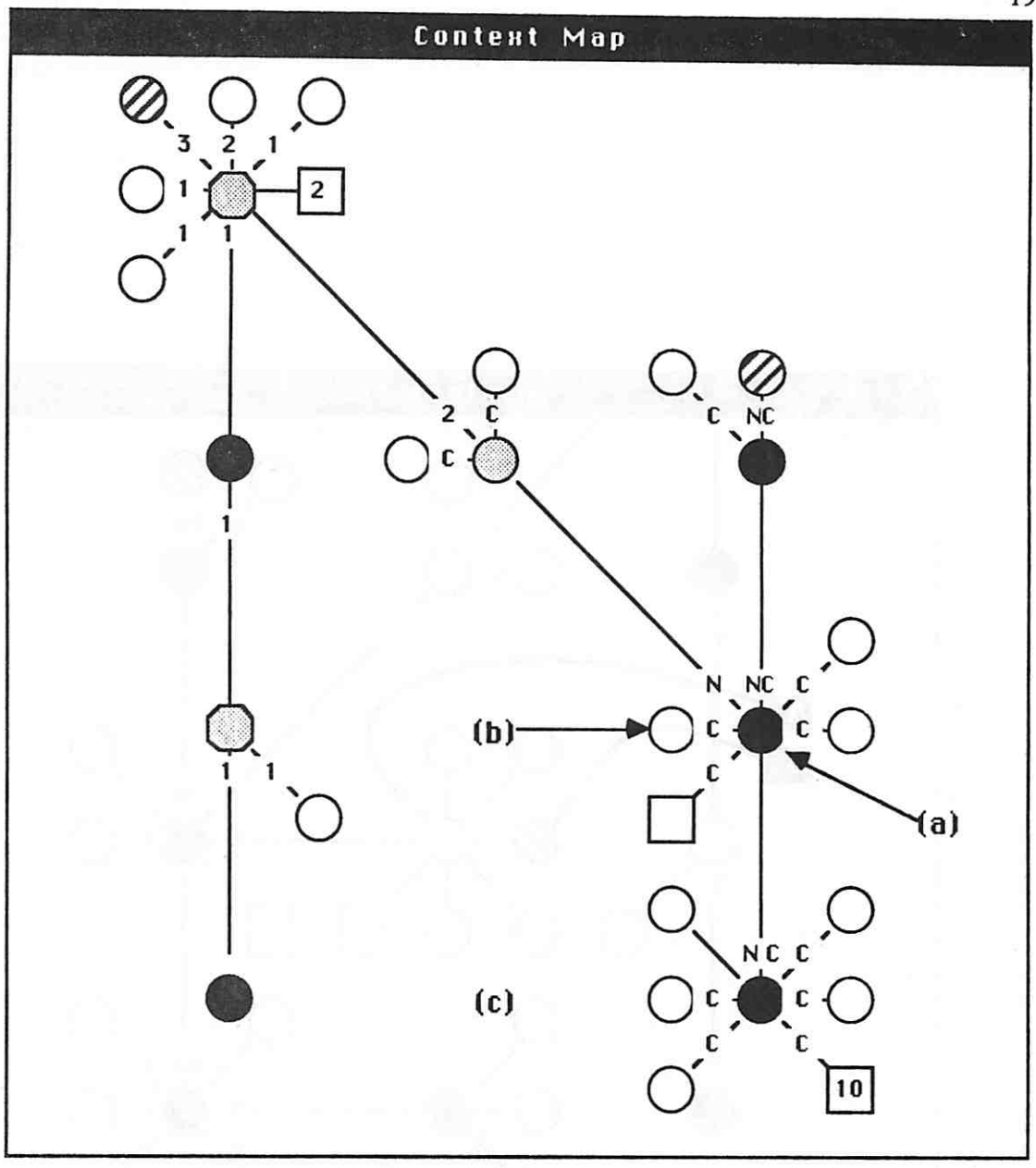

Figure 6.35: Context Map after examining document #2846 (menus not shown, but are the same as those with the neighborhood map).

If this location was taken, as in figure 6.36, the system would draw the node in another region of the map and place a connector icon on a link leading to it (figure 6.37). Selecting the connector would put the map back in the region where the node originally was drawn.

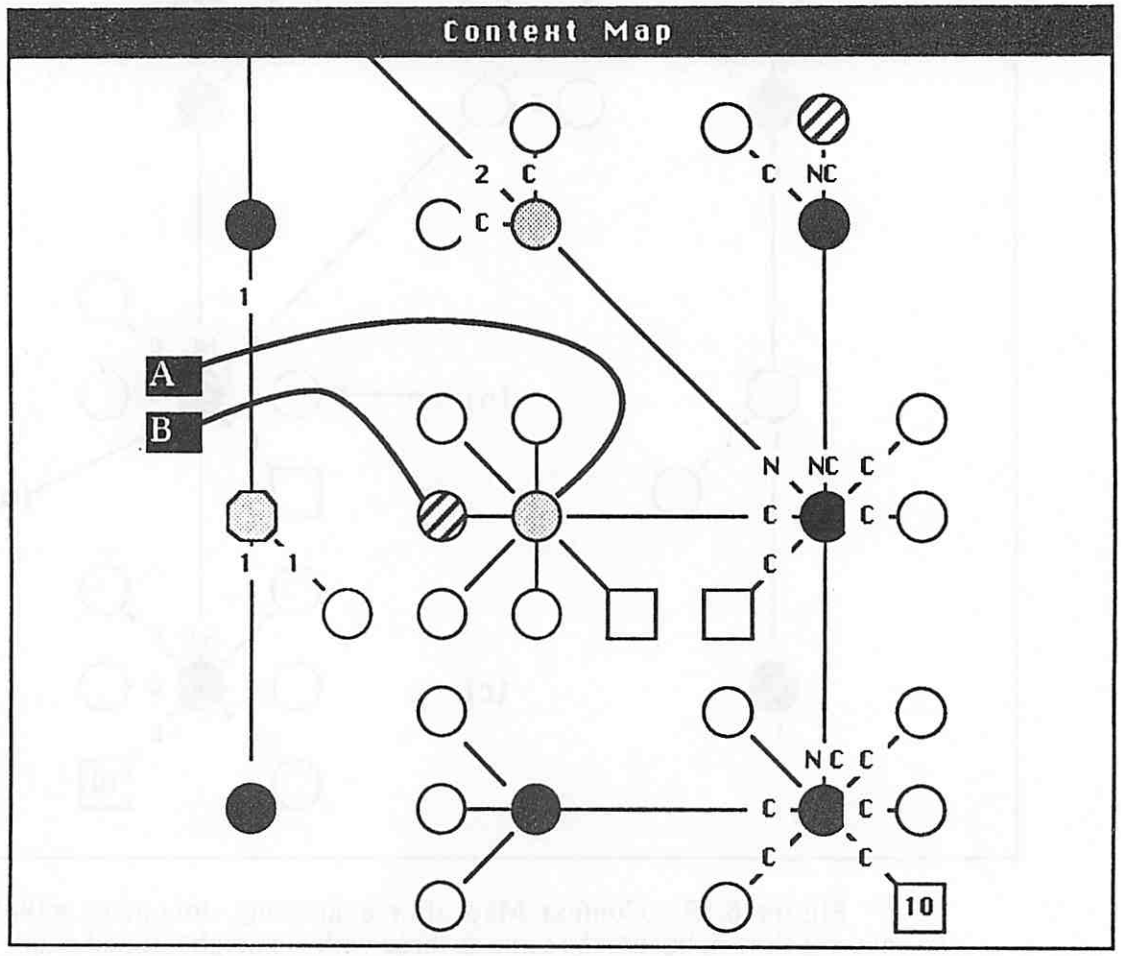

Figure 6.36: Context Map showing crowded region around node "A," and user desires to expand node "B."

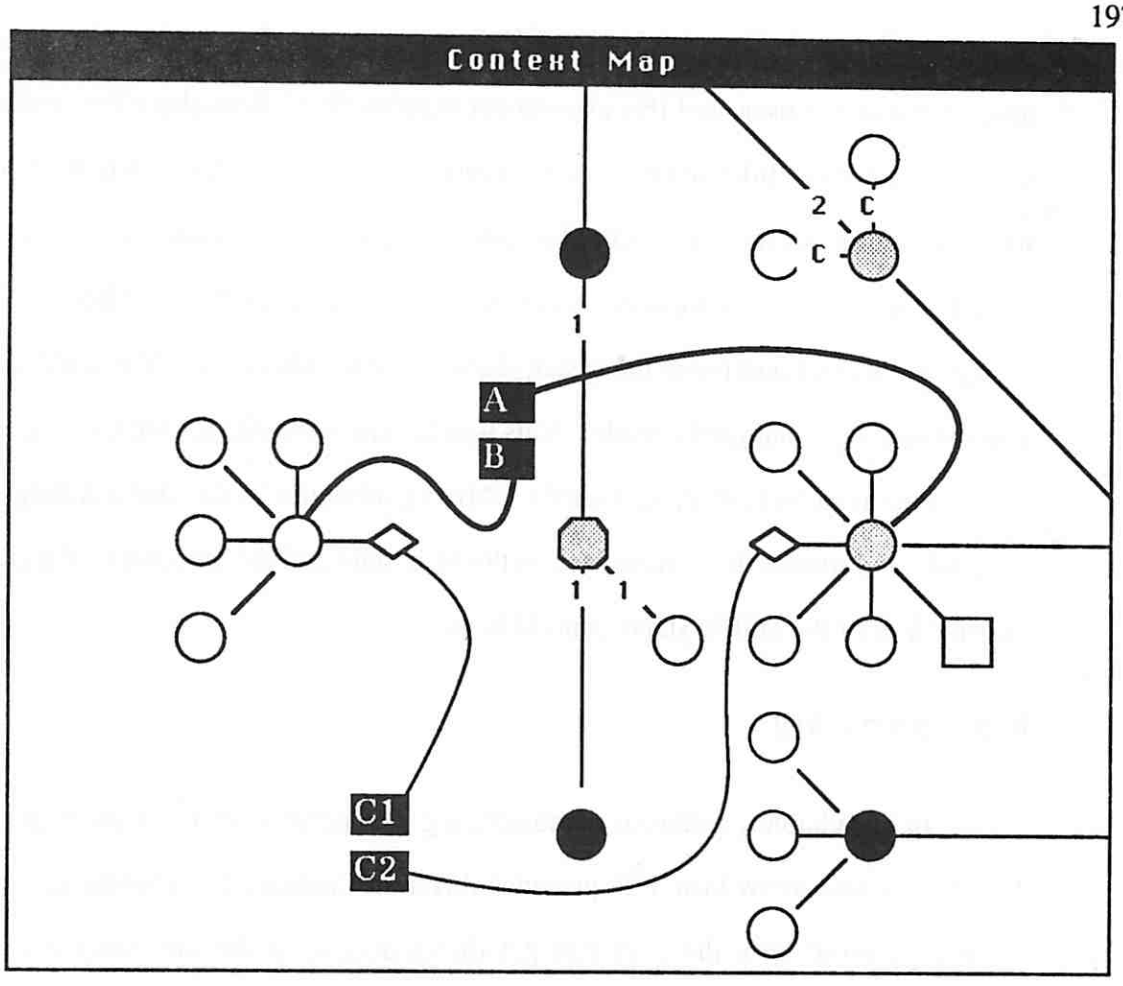

Figure 6.37: Use of connector to expand node "B." Connectors are placed on the links to facilitate the movement between nodes.

The connectors are useful primarily on the neighborhood map, where the context is limited to the immediate neighborhood. When the user selects connector C1 (figure 6.37), the node marked "A" is placed in the center of the map. When connector C2 is selected the node marked "B" is centered. In this way the user can follow the path he has selected. The connectors can be useful in the context map too. If it is very dense in one region the new expanded node can be very far away from the node it was expanded from.

6.4 Possible Behavioral Changes

There are other behaviors that could be observed by changing the control expert and by significantly enhancing the UMB. Currently, the UMB makes only a simple initial

assessment of the user, and this assessment remains fixed throughout the session. If the user model had confidence factors associated with it assessments, and if the UMB monitored the activity of the user, it could continually adjust the model to reflect how the user's activity corresponded to the initial assessment. If this fell below a threshold the CE could recognize this fact and move the system back to a state where the UMB could again question the user and change the model. This would simply require for the CE, the addition of one transition (rule) that recognizes the UMB's confidence in the model falling below the threshold and moves the system back to the $CU state and the addition of the UMB to the priority lists of the $GIN, $DNC, and $ER states.

6.5 Summary

In this chapter, scenarios demonstrating the operation of I^3R have been presented. These scenarios show how I^3R provides different facilities to different kinds of users. They also show how the user can get direct access to the information in the concept/document knowledge base by browsing. Indications have been made as to how the behavior of the system could be altered by the addition of transitions to the control expert.

CHAPTER 7

CONCLUSION

7.1 Summary

This thesis has presented an information retrieval system that embodies the concept of an intelligent interface. This intelligent interface is implemented by an architecture that has a number of innovative features that support facilities that help the end-user overcome the difficulties of previous IR systems.

The first major feature is the implementation of the major functions of the system as individual rule-based systems, called experts. These individual rule systems operate independently, posting the results of their activity on a short term memory. This promotes the clean separation of functions and allows new major functions to be added with minor impact on the other experts in the system. The implementation of the experts using rules allows the incremental development of each expert, so that minor changes can be made to the expert without grossly affecting its current operation. The idea of multiple independent cooperating experts communicating using a blackboard [Erman 80] as an architecture for I^3R was developed from a system analysis point of view [Croft 85]. A similar architecture was independently specified by Belkin [1983, 1984]. A later system, CODER, has also adopted this basic architecture, but uses two blackboards with separate knowledge sources, one for documents analysis and the other for retrieval [France 86].

The second innovative feature is the coordination of the operation of the experts by a flexible control structure based on an analysis [Brooks 83, 85, 86, Daniels 85, 86] of actual end-user/search intermediary interactions. This control structure, implemented as a transition network, moderates the activity of the experts to provide a logical dialogue with the user, so that he does not become confused by numerous independent demands for

information from the experts. The course of the dialogue is determined by the progress that the user has made in expressing his information need and retrieving documents relevant to his information need. The measurement of the progress is determined by the use of expectations of the number of relevant documents retrieved and the number of searches performed.

The operation of the system and its adaptation to a particular user is controlled by stereotypes, which are judgements about the user's domain experience, IR system experience, and interest in either an exhaustive or selective search. These stereotypes determine the expectations used by the control expert to determine the progress of a search session. They also determine what facilities are available to the user. An expert user has more facilities available to him and can take more initiative to control the course of a session than a novice user can. This idea has been subsequently used in the IOTA system [Chiaramella 87] as a means to adapt its natural language responses to different kinds of users.

Another important innovation in I^3R is the use of user supplied domain knowledge. This obviates the need for a significant investment of resources to derive a global thesaurus for system use. Instead, the effort is spread out, so that the users themselves supply the relevant domain knowledge for the particular information need. If global domain knowledge is available, it can be used. Since the user and global domain knowledge are represented in the same way, the knowledge supplied by the users can be migrated into the the globally available domain knowledge. In this way, as the system is used, its base of domain knowledge can increase and domain knowledge obtained from experts can be made available to all the users. An extension of this concept would be to provide facilities to allow users to explicitly share their domain knowledge models.

I^3R was the first system to propose and implement the use of multiple search strategies in order to take advantage of the differences in performance of techniques based on a variety of retrieval models. In previous systems the concept of adaptable search

strategies was based on the experience of human search intermediaries in manipulating the form of Boolean queries to achieve the best possible results.

Another major innovative feature of I^3R is the incorporation of browsing as a method for both query formulation and search. In fact, browsing merges these two phases of information retrieval. In the process of query formulation, browsing can help the user to find concepts that more accurately represent his information need by providing a path from the concepts that he already knows to new ones. He can then check their relevance by examining their use in documents. In the process of search, the user can examine documents to determine their relevance, and from a relevant document can follow links to other documents that are related by citations or by similar content as determined by nearest neighbor links. The user is not restricted to moving from documents to documents but can move to concepts or to journal issues. As the user browses, and makes judgements about the relevance of concepts and documents, the system records this information and uses it to further enhance its model of the user's information need.

The user is assisted during browsing by recommendations made by the Browsing Expert. These recommendations are based on the structure of the concept/document data and the information in the request model. This information provides evidence for the browsing expert so that it can determine what concept or document is most likely to be useful to the user at his current location in the concept/document information.

Another interesting feature of I^3R is the use of graphics to provide the user with visual context to aid him in the browsing process. The maps that the system constructs gives the user a context that helps him determine where he has been, so that he does not get lost while browsing in the highly interconnected network of documents and concepts.

Finally, the underlying organization of concept/document knowledge in I^3R allows it to support multiple search strategies, browsing, and the use of different domain knowledge models. This organization is significantly more sophisticated than previous ones,

since it fuses what are normally separate sources of knowledge into a single knowledge base.

7.2 Future Directions

7.2.1 Evaluation of the system

The primary work to be performed on I^3R in the future is the evaluation of its performance. This encompasses not only its effectiveness in retrieving relevant documents, but also its usability with regard to the extra facilities. In other words, how well do the extra facilities help the user to get the results he desires. The difficulties of evaluating a highly interactive system have been discussed in sections 2.2.3 and 6.2.

7.2.2 Extension of the experts

The prototype implementation used basic implementations of many of the experts to demonstrate the functionality of the overall system. All of the experts can be extended in a number of ways.

The most important expert to develop further is the user model builder. By developing the user model to a greater extent, the basis is formed for developing more sophisticated patterns of interaction with the user. At present, the user model is constructed solely on the basis of the user's responses to questions posed by the UMB. A more sophisticated UMB would monitor the activity of the user, as well as asking him questions, to determine the stereotypes that apply. Modelling the user based on his activity requires that the UMB have models of different patterns of system usage. These models can be developed only by studying the patterns of usage by many different users over a long period. Initial work on the functions and structure of a sophisticated UMB has been the

subject of study by Dainiels [1985, 1986, 1987]. The stereotypes of the users could take on the structure of those proposed by Rich [1979].

The request model builder can be extended by feeding back to it from the results of the search the terms that were used to make the decision about what documents to retrieve. This information can be shown to the user, so that he may understand what contributed to the selection of documents that were shown to him. Also, in relation to the request model, the user should be given a way to look at and alter the request directly in the form of a request model display. Whether this display is alterable and the kinds of information shown on the display would be controlled by the user stereotypes. For example, an expert user would definitely be allowed to alter the model and could be shown the probabilistic weights of the terms. A novice, perhaps, would only be shown the relative importance of the terms by their ordering based on their weight. In addition to this the user should have the ability to invoke a search manually.

It is also important to make the control expert more flexible in its operation. This is also based on the extension of the user model builder. With a better user model, the system can act more intelligently about the course of the retrieval session.

7.2.3 Addition of an Explainer

One of the major experts defined in the requirements specified in chapter three was an explainer. The need for this expert as well as other considerations lead to the selection of rules as the basis for the implementation of the experts as rule-based systems, so that their reasoning for their actions could be made available to the user.. Furthermore, this expert is particularly important in assisting novice users of the system. Because of these considerations, an intelligent explainer is an important extension to the current capabilities of I^{3}R.

To accomplish this, the previously mentioned extensions to the user model builder must be made. The explainer needs for its basis a good model of the user representing not only who the user is, but how he interacts with the system. For example, a user may usually express his information need as a full text statement. If he then chooses to use a complex Boolean formulation, the user model builder would notice this, making an assertion that the user is using an unfamiliar method to enter his query. In response to this assertion, the explainer might be more likely to offer assistance These requirements involve some significant research issues.

An initial pass at implementing an explainer would be to provide to the user with help facility like that available on the VAX/VMS operating system and then record the kinds of help that the user requests. The explainer would be integrated into the system in much the way that the interface manager is. It would run as a separate expert, not under the coordination of the control expert.

7.2.4 Natural Language Techniques

I^3R represents a beginning in the integration of AI techniques into IR systems. There are a number of ways that I^3R can be extended. The first way is the incorporation of natural language processing techniques in a number of different areas. Since, in the document retrieval domain, the primary means of expressing knowledge is natural language, natural language processing techniques are prime candidates for inclusion into I^3R. This has already begun in the context of improving the performance of the search techniques. This work is being pursued as an independent system called ADRENAL [Croft 87]. The focus of this work is to define and demonstrate the use of NL techniques that can be applied effectively in systems that contain large numbers of documents. The basic idea is to apply the techniques to compare the query to sets of documents that have been retrieved using traditional methods. This limits the comparison to those documents that have a like-

lihood of being relevant to the information need. These techniques could be added to I^3R by extending the search controller or by adding a new expert.

Another place where NL techniques could be added to I^3R is in the analysis of the context of citations. While it has been shown that citations are a valuable source of information for finding relevant documents, more information could be determined about what a citation means from its context. For example, some citations point to articles that have a differing point of view than the author's, or they can point to work that contains similar ideas. This kind of information would be valuable for the operation of the browsing expert, since it would allow the BE to make a more informed recommendation to the user. These kinds of citation link types have been partially defined by Trigg [1986]. This kind of information could be incorporated into I^3R by adding link types to the citations links, extending the heuristics of the browsing expert, and adding an expert to do the analysis of the documents.

7.2.5 Representation

Another area in which I^3R can be extended is in the information kept about different kinds of journals and particular issues. For example, in the publications of the ACM, Computing Surveys is primarily directed at providing tutorial kinds of articles, whereas the Journal of the ACM is focussed at highly theoretical articles requiring a relatively high degree of mathematical sophistication. Having this kind of information as evidence would allow the system to provide documents that are better matched to the user's level of domain expertise.

7.2.6 Interface Considerations

Finally, an area of major concern in the further development of I^3R is the enhancement of the graphical presentation of information. One of the ways that this can be ac-

complished is the use of color to convey other kinds of information such as relevance. For example, the most relevant documents could be displayed in red and decreasing relevance shown by scaling the colors to blue to represent the least relevant. Another use of color is highlighting different groups of links.

Another possibility is to use three dimensional representation to show more of the neighborhood of a node in the concept/document knowledge. Specifically, extending the current organization of the neighborhood and context maps to three dimensions would allow the map to show 18 more nodes for a total of 25. Use of three dimensions to represent the neighborhoods would be available to system experts, since very few people are trained to visualize in three dimensions.

On a purely implementational note, the entire interface could be rewritten using the currently developing X-standard [Scheifler 86]. This would allow I^3R to operate on a much wider variety of architectures.

BIBLIOGRAPHY

[Barbi 84]
BARBI, E. ET. AL. "A conceptual approach to document retrieval." *Proc. of the 2nd ACM-SIGOA Conf. on Office Info. Sys.* 219-226, 1984.

[Bates 79a]
BATES, M.J. "Information search tactics." *J. of the Amer. Soc. for Info. Sci*, **30:** 206-214, 1979.

[Bates 79b]
BATES, M.J. "Idea tactics." *J. of the Amer. Soc. for Info. Sci*, **30:** 280-289, 1979.

[Bates 86]
BATES, M.J. "Terminological assistance for the online subject searcher." *Proc. of the 2nd Conf. on Computer Interfaces & Intermediaries for Info. Ret.* Defense Technical Information Center, Alexandria VA, 1986.

[Belkin 83]
BELKIN, N.J., SEEGER, T., & WERSIG, G. "Distributed expert problem treatment as a model for information system analysis and design." *J. of Info. Science*, **5**: 153-167, 1983

[Belkin 84]
BELKIN, N.J., HENNINGS, R.D., SEEGER, T.,".Simulation of a distributed expert-based information provision mechanism."*.Information Technology*, **3**:122-141, 1984.

[Belkin 86]
BELKIN, N.J. & KWASNIK, B.H. "Using structural representations of anomalous states of knowledge for choosing document retrieval strategies." *Proc. of the ACM SIGIR Int'l Conf. on Research and Development in Info. Retrieval.* 11-22, Pisa, Italy 1986.

[Belkin 87a]
BELKIN. N.J. & CROFT, W.B. "Retrieval techniques." in *Annual Review of Info. Science and Technology*, American Society for Info. Science, **22:** 109-145, 1987.

[Belkin 87b]
BELKIN, N.J. ET. AL. "Distributed expert-based information systems: an interdisciplinary approach." *Information Processing and Management*, **23:** 395-409, 1987.

[Borgman 87]
BORGMAN, C.L. "Information system functionality: a user-driven perspective." Position paper presented at *Workshop on Distributed Expert-Based Info. Systems*, 1987.

[Brajnik 85]
BRAJNIK, G. GUIDA, G. & TASSO, C. "An expert interface for effective man-machine interaction." *Cooperative Interactive Systems*, L.Bolc (Ed.). Springer-Verlag: New York, 1985.

[Brajnik 87]
BRAJNIK, G. GUIDA, G. & TASSO, C. "User modeling in intelligent information retrieval." *Info. Proc. & Management*, 23: 305-320, 1987.

[Brooks 83]
BROOKS, H.M. & BELKIN, N.J. "Using discourse analysis for the design of information retrieval interaction mechanisms." *Proc. of the Sixth Annual International ACM SIGIR Conf.: Research and Development in Info. Retrieval*, 31-47, 1987.

[Brooks 85]
BROOKS, H.M., DANIELS, P.J., & BELKIN, N.J. "Problem descriptions and user models: Developing an intelligent interface for document retrieval systems." *Proc.of Informatics 8*, Oxford, April 1985, London Aslib, 1985.

[Brooks 86]
BROOKS, H.M. "Developing and representing problem descriptions." *IRIFS 6: Intelligent Info. Systems for the Info. Society*, Frascati, Italy 1985., Amstersdam, North Holland, 1986.

[Brooks 87a]
BROOKS, H.M. "The functions of an information system: the Monstrat model." Position paper presented at *Workshop on Distributed Expert-Based Info. Systems*, 1987.

[Brooks 87b]
BROOKS, H.M. "Experts systems and intelligent information retrieval." *Info. Proc. & Management*, 23: 367-382, 1987.

[Burket 79]
BURKET, T.G. EMRATH, P. & KUCK, D.J. "The use of vocabulary files for online information retrieval." *Info. Proc. & Management*, **15:** 181-189, 1979.

[Caplinger 86]
CAPLINGER, M. "Graphical Database Browsing." *Proc. of Third ACM-SIGOIS Conference on Office Inf. Sys.*, October 1986, Providence Rhode Island, 113-121.

[Carberry 87]
CARBERRY, S. "Workshop report: First international workshop on user-modeling." *The AI Magazine*, Fall: 71- 74, 1987.

[Charniak 85]
CHARNIAK, E. & MCDERMOTT, D. *Introduction to Artificial Intelligence*, Reading, Massachusetts, Addison-Wesley, 1985.

[Chiaramella 87]
CHIARAMELLA & Y. DEFUDE, B. "A prototype of an intelligent system for information retrieval: IOTA." *Info. Proc. & Management*, **23:** 285-304, 1987.

[Cleverdon 66]
CLEVERDON,C.W., MILLS.,J., KEEN, E.M. "Factors determining the performance of indexing systems." *Aslib Cranfield Project, Vol. 1—Design*, Cranfield, England, 1966.

[Cohen 87]
COHEN, P. R. & KJELDSEN, R. "Information retrieval by constrained spreading activation in semantic networks." *Info. Proc & Management* **23**: 255-268, 1987.

[Conklin 87]
CONKLIN, J. "Hypertext: an introduction and survey." *IEEE Computer* **20** (Sept): 17-41, 1987.

[Cooper 1971]
COOPER, W.S., "A definition of relevance for information retrieval." *Information Storage and Retrieval*, 7:19-37, 1971.

[Croft 79]
CROFT, W.B. & HARPER, D.J. "Using probabilistic models of document retrieval without relevance information." *J. of Documentation*, **35**: 285-289, 1979.

[Croft 83]
CROFT, W.B., WOLF, R., THOMPSON, R.H. "A network organization used for document retrieval." *Proc. of Sixth Annual Int'l ACM SIGIR Conf. on Research and Development in I.R.*, June 6–8, 1983, Bethesda, Maryland, SIGIR Forum **17**:4, 178-188, 1983.

[Croft 84]
CROFT, W. B. & THOMPSON, R.H. "The use of adaptive mechanisms for selection of search strategies in document retrieval systems." *Research and Development in Info. Retrieval: Proc of the 3rd joint BCS and ACM symposium,* July 2-6, Cambridge, England, Cambridge University Press, 95-110,1984.

[Croft 85a]
CROFT, W.B. & PARENTY, T.J. "A comparison of a network structure and a database system used for document retrieval." *Info. Sys.* **10**: 377-390, 1985.

[Croft 85b]
CROFT, W.B.& THOMPSON, R.H., *An expert assistant for document retrieval.* Amherst, MA: University of Massachusets, Dept of Comp. and Info. Science, Tech Report 85-05, 1985.

[Croft 86a]
CROFT, W.B. "Boolean queries and dependencies in probabilistic retrieval models." *J. of the Amer. Soc. for Info. Sci.* **37**: 71-77, 1986.

[Croft 86b]
CROFT, W.B. "User-specified domain knowledge for document retrieval." *Proc. of the ACM SIGIR Int'l Conf. on Research and Development in Info. Retrieval.* 201-206, Pisa, Italy 1986.

[Croft 86c]
CROFT, W.B. & THOMPSON, R.H. "I^3R: a new approach to the design of document retrieval systems." *J. of the Amer. Soc for Info. Sci.* (to appear), 1986.

[Croft 87a]
CROFT, W.B. & LEWIS, D.D. "An Approach to natural language processing for document retrieval." *Proc. of the Tenth Annual International ACM SIGIR Conf.: Research and Development in Info. Retrieval*, 26-32, 1987.

[Croft 87b]
CROFT, W.B. "Approaches to intelligent information retrieval." *Info. Proc. & Management*, **23**: 249-254, 1987.

[Daniels 85]
DANIELS. P.J., BROOKS, H.M., & BELKIN, N.J. "Using problem structures for driving human-computer dialogues." in *RIAO 85. Actes of the Conf.: Rechere d'Informations assistee par Ordinateur*,203-221, Grenoble, France, I.M.A.G., 1985.

[Daniels 86]
DANIELS, P.J. "The user modelling function of an intelligent interface for document retrieval systems." *IRIFS 6: Intelligent Info. Systems for the Info. Society*, Frascati, Italy 1985., Amstersdam, North Holland, 1986.

[Daniels 87]
DANIELS, P.J. *Developing the user modelling function of an intelligent interface for document retrieval systems*. Ph.D. Thesis, City University, London, 1987.

[Defude 85]
DEFUDE, B. "Different levels of expertise for an expert system in information retrieval." *Proc. of the 8th Annual Intl' ACM SIGIR Conf. on Research and Development in Info. Retrieval,* June 1985, Montreal, Canada, 147-153, 1985.

[Dijkstra 68]
DIJKSTRA, E.W. "GOTO statement considered harmful." *Com. of the ACM*, **11**: 147–148, 1968.

[Dillon 83]
DILLON M. & GRAY, A.S. "FASIT: A fully automatic syntactical based indexing system." *J. of the Amer. Soc. for Info. Sci.* **34**: 99-108, 1983.

[Dowty 81]
DOWTY, D.R., WALL, R.E., PETERS, S. *Introduction to Montague Semantics*, Dordrecht: Reidel, 1981.

[Englebart 68]
ENGLEBART, D.C. & ENGLISH, W.K. "A research center for augmenting human intellect." *Proc. of the Fall Joint Computer Conf.*, 395- 410. AFIPS Press, 1968.

[Englebart 72]
ENGLEBART, D. C. WATSON, R. C. & NORTON, J. C. "The augmented knowledge workshop." *Proc of the Nat. Computer Conf.* 9-21, 1972.

[Erman 80]
ERMAN, L.D., ET. AL. "The hearsay-II speech understanding system: integrating knowledge to resolve uncertainty." *ACM Computing Surveys*, **12**: 213–253, 1980.

[Forgy 82]
FORGY, C.L., "Rete: a fast algorithm for the many pattern/many object pattern match problem." *Artificial Intelligence*, **19**: 17-37, 1982.

[Fox 85]
FOX, E.A. "Composite document extended retrieval: an overview." *Proc. of the 8th Annual Intl' ACM SIGIR Conf. on Research and Development in Info. Retrieval*, June 1985, Montreal, Canada, 42-53, 1985.

[Fox 87a]
FOX, E.A. "Building the CODER system: working toward a comprehensive testbed for AI in IR." Position paper presented at *Workshop on Distributed Expert-Based Info. Systems*, 1987.

[Fox 87b]
FOX, E.A. "Development of the CODER system: a testbed for artificial intelligence methods in information retreival." *Info. Proc. & Management*, **23**: 341-367, 1987.

[France 86]
FRANCE, R.K., *An artificial intelligence environment for information retrieval*, MS Thesis, Computer Science Department, Virginia Polytechnic Institute & State College, Blacksburg, VA, 1986.

[Frei 83]
FREI, H.P & JAUSLIN, J.-F. "Graphical presentation of information and services: a user-oriented interface." *Info. Technology: Research and Development*, **2**: 23-42, 1983.

[Godin 85]
GODIN, R. SAUNDERS, E. & GECSEI, J. "Lattice model of browseable data spaces." Publication #550, Department D'Informatique et de Rescherche Operationnelle, Universite de Montreal, Quebec, Canada, 1985.

[Greenberg 88]
GREENBERG, D.P. "Coons award lecture." *Communications of the ACM*, **31**:122-129, 151, 1988.

[Hayes-Roth 83]
HAYES-ROTH, F. WATERMAN, D.A. & LENAT, D.B, (Eds.) *Building Expert Systems*. Reading, Massachusetts, Addison-Wesley, 1983.

[Hayes-Roth 85]
HAYES-ROTH, B. "A blackboard architecture for control." *Artificial Intelligence*, **6**: 251-321, 1985.

[Horowitz 83]
HOROWITZ, E., SAHNI,S. *Fundamentals of Data Structures in PASCAL*. Computer Science Press: Rockville, Maryland, 1983.

[Ingwersen 84]
INGWERSEN, P. "Search procedures in the library–analysed from the cognitive point of view." *J. of Documentation*, **38**: 165-191, 1982.

[Kay 77]
KAY, A. GOLDBERG, A. "Personal dynamic media." *IEEE Computer*, 31-41, 1977.

[Kjeldsen 87]
KJELDSEN, R. & COHEN, P. J. "The evolution and performance of the GRANT system." *IEEE Expert*, 2: 73-79, 1987

[Korfhage 82]
KORFHAGE, R.R. & CHAVARRARIA-GARZA, H., "Retrieval Improvement by the Interaction of Queries and User Profiles." *Proc. of the COMPSAC '82, Sixth International Conf. on Computer Software and Applications*, 470-475, 1982.

[Kolodner 83]
KOLODNER, J. L. "Indexing and retrieval strategies for natural language fact retrieval." *ACM Trans. on Office Info. Sys.* **8**: 434-464, 1983.

[Krawczak 85]
KRAWCZAK, D.A. ET.AL, "A knowledge-based system to aid in searches of the environmental pollution literature." *Proc. IEEE 1985 Second Conf. on Artificial Intelligence Applications*, 552-557, 1985.

[Mansur 80]
MANSUR, O. "On selection and combining of relevance indicators." *Info. Proc. & Management*, **16**: 139-153, 1980.

[Marcus 81a]
MARCUS, R.S. & REINJES, J.F. "A translating computer interface for end-user operation of heterogeneous retrieval system. I. Design." *J. of the American Soc. for Info. Sci.* **32**: 287-303, 1981

[Marcus 81b]
MARCUS, R.S. & REINJES, J.F. "A translating computer interface for end-user operation of heterogeneous retrieval system. II. Evaluations." *J. of the American Soc. for Info. Sci.* **32**: 304-317, 1981.

[Marcus 83]
MARCUS, R. S. "An experimental comparison of the effectiveness of computers and humans as search intermediaries." *Journal of the Amer. Soc. for Info. Sci.* **34**: 381-404, 1983.

[Mason 86]
MASON, M.V. "Adaptive command prompting in an on-line documentation system." *Int. J. of Man-Machine Studies*, 25: 33-51, 1986

[McCracken 84]
MCCRACKEN, D.L. & AKSCYN, R.M. "Experience with ZOG human-computer interface system." *Int. J. of Man-Machine Studies*, **21**: 293-310, 1984

[McCormick 87]
MCCORMICK, B.H. et al. (ed) "Visualization in Scientific Computing." *Computer Graphics* **21**:1-19, 1987.

[McCune 83]
MCCUNE, B.P. ET. AL. "RUBRIC: a system for rule-based information retrieval." Tech. Rep #1018-1, Advanced Decision Systems, Palo Alto, CA, 1983.

[Motro 86]
MOTRO, A. "BAROQUE: an exploratory interface to relational databases." *ACM Trans. on Office Info. Sys.* **4**: 164-181, 1986.

[Murtagh 82]
MURTAGH, F. "A very fast, exact nearest neighbor algorithm for use in information retrieval." *Info. Technology* **1**: 275-284, 1982.

[Nii 86a]
NII, H. P. "Blackboard Systems: the blackboard model of problem solving and the evolution of blackboard architectures." *The AI Magazine*, Summer: 82-105, 1986.

[Nii 86b]
NII, H. P. "Blackboard systems: blackboard application systems, blackboard systems from a knowledge engineering perspective." *The AI Magazine*, August: 82-106, 1986.

[Ofori 82]
OFORI-DWUMFOO, G.O., *Document retrieval based on a cognitive model of dialogue*, Ph.D. Thesis, University of Aston in Birminghan, 1982.

[Oddy 74]
ODDY, R.N. *Reference Retrieval Based on User Induced Dynamic Clustering*, Ph.D. Thesis, University of Newcastle upon Tyne, 1974

[Oddy 77]
ODDY, R.N. "Information retrieval through man-machine dialogue." *J.of Documentation*, **33**: 1-14, 1977.

[Palay 80]
PALAY, A.J. & FOX, M.S. "Browsing through databases." in *Info. Retrieval Research, Proc of the First Annual Symposium on Research and Development in Info. Retrieval*, Butterworths: London, 310-324, 1980.

[Patel 85]
PATEL-SCHNEIDER, P.F. BRACHMAN, R. J. LEVESQUE, H.J. "ARGON: representation meets information retrieval." *IEEE 1984 First Conf. on Artificial Intelligence Applications*, 280-286, 1985.

[Pollitt 84]
POLLITT, A.S. "End User Touch Searching for Cancer Therapy Literature – a Rule Based Approach" *Proc. of Sixth Annual Int'l ACM SIGIR Conf. on Research and Development in I.R.*, June 6–8, 1983, Bethesda, Maryland, SIGIR Forum **17**:4, 136-145, 1983

[Porter 80]
PORTER, M.F. "An algorithm for suffix stripping." *Program*, **14**: 130-137, 1980.

[Quillian 68]
QUILLIAN, M.R. "Semantic Memory." in *Semantic Information Processing* (M.Minsky Ed.), Cambridge, Mass.: MIT Press, 216-270, 1968.

[Rau 87]
RAU. L.F. "Knowledge organization and access in a conceptual information system." *Info. Proc. & Management*, **23**: 269-284, 1987.

[Rich 79]
RICH, E. "Building and exploiting user models." Ph. D. Thesis, Carnegie-Mellon University, Technical Report CMU-CS-79-119, 1979.

[Rissland 84]
RISSLAND, E.L. "Ingredients of intelligent user interfaces." *Int. J. of Man-Machine Studies*, **21**: 377–388, 1984.

[Rouse 82]
ROUSE, W. B. ROUSE, S. H. & MOREHEAD, D. R. "Human information seeking: online searching of bibliographic citation networks." *Info. Proc. & Management*, **3**: 141-149, 1982.

[Salton 83]
SALTON, G. & MCGILL, M. *Introduction to Modern Information Retrieval.* New York: McGraw-Hill, 1983.

[Schank 75]
SCHANK, R.C. *Conceptual Information Processing.* New York: North Holland, 1975

[Scheifler 86]
SCHEIFLER, R.,W., GETTYS, J. "The X Window System." *ACM Transactions on Graphics*, 1986.

[Shoval 85]
SHOVAL P. "Principles, procedures and rules in an expert system for information retrieval." *Info. Proc. & Management*, **21**: 475-487, 1985.

[Simmons 87]
SIMMONS, R.F. "A text knowledge base from the AI handbook." *Info. Proc. & Management*, **23**: 321-340, 1987.

[Smeaton 81]
SMEATON, A.F. & VAN RIJSBERGEN C.J. "The nearest neighbor problem in information retrieval." *Proc. of the 4th ACM SIGIR Conf.* SIGIR Forum **16**: 83-87, 1981.

[Smeaton 86]
SMEATON, A.F. "Incorporating syntactic information into a document retrieval strategy: an investigation." *Proceedings of the ACM SIGIR Int. Conf. on Research and Development in Info. Retrieval*, 103-113, Pisa Italy, 1986.

[Sparck Jones 77]
SPARCK JONES, K. & BATES, R.G., *Research on automatic indexing 1974-1976.* British Library Research and Development Report 5464, Computer Laboratory, University of Cambridge, 1977.

[Sparck Jones 80]
SPARCK JONES, K. & WEBSTER, C.A. *Research on Relevance Weighting.* British Library Research and Development Report 5553, Computer Laboratory, University of Cambridge, 1980.

[Sparck Jones 83]
SPARCK JONES, K. "Intelligent Retrieval." in *Intelligent Info. Retrieval: Informatics 7* (Proceedings of Informatics 7) (ed Jones), London: Aslib, 1983.

[Sparck Jones 84]
SPARCK JONES, K. & TAIT, J.I. "Automatic search term variant generation." J. *of Documentation,* **40**: 50-66, 1984.

[Sparck Jones 87]
SPARCK JONES, K. "Architecture problems in the construction of expert systems for document retrieval." presented in Workshop on Distributed Expert-Based Information Retrieval, 1987.

[Sparck Jones 88]
SPARCK JONES, K. "A look back and a look forward." *Proceedings of the 11th Int'l. ACM SIGIR Conf. on Research and Development in Info. Retrieval,* 13-29, Grenoble France, 1988.

[Stallman 86]
STALLMAN, R. *GNU Emacs Manual,* Fifth Edition, Emacs Version 18, Cambridge, Massachusetts, Free Software Foundation, 1986.

[Tong 86]
TONG, R.M. ET. AL. "RUBRIC III: an object-oriented expert system for information retrieval.", *Second Symposium on Expert Systems in Government,* IEEE, 1986.

[Trigg 83]
TRIGG R.H. *A network-based approach to text handling for the online scientific community.* Ph.D. Thesis, Maryland Artificial Intelligence Group, University of Maryland, 1983.

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

[Van Rijsbergen 86]
VAN RIJSBERGEN, C. J. "A non-classical logic for information retrieval." *The Computer Journal,* **29**: 481-485, 1986.

[Vigil 83]
VIGIL, P. J. "The psychology of online searching." *J. of the Amer. Soc. for Info. Sci.* **34**: 281-287, 1983.

[Waltz 78]
WALTZ, D.L. "An English language question answering system for a large relational database." *Com. of the ACM*, **21**: 526-539, 1978.

[Wilensky 83]
WILENSKY, R. *Planning and understanding, a computational approach to human reasoning*. Reading, Massachusetts, Addison-Wesley, 1984.

[Weyer 82]
WEYER, S.A., "The design of a dynamic book for information search." *Int'l. J. of Man Machine Studies*, **17**: 87-107, 1982.

[Winston 84]
WINSTON, P.H. *Artificial Intelligence*, Second Edition. Reading, Massachusetts, Addison-Wesley, 1984.

[Woolf 84]
WOOLF, B.P. *Context dependent planning in a machine tutor*, Ph.D. Thesis, University of Massachusetts, COINS Tech. Report 84-21, 1984.

[Yip 79]
YIP M.-K. *An expert system for document retrieval*, M.S. Thesis, Massachusetts Institute of Technology, 1979.

[Zloof 77]
ZLOOF, M.M. "Query-by-example." *IBM System J.* **16**: 4, 1977.