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].

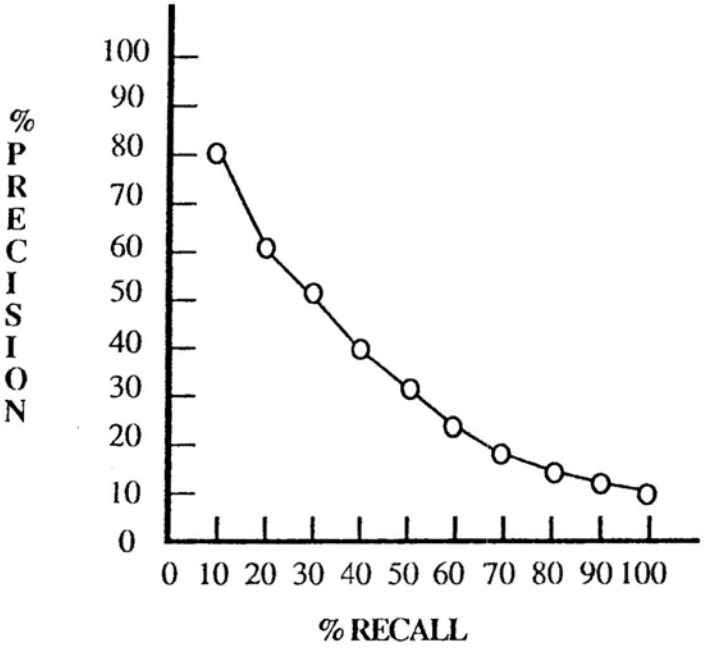

Figure 2.3: Typical precision/recall graph.

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.

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.

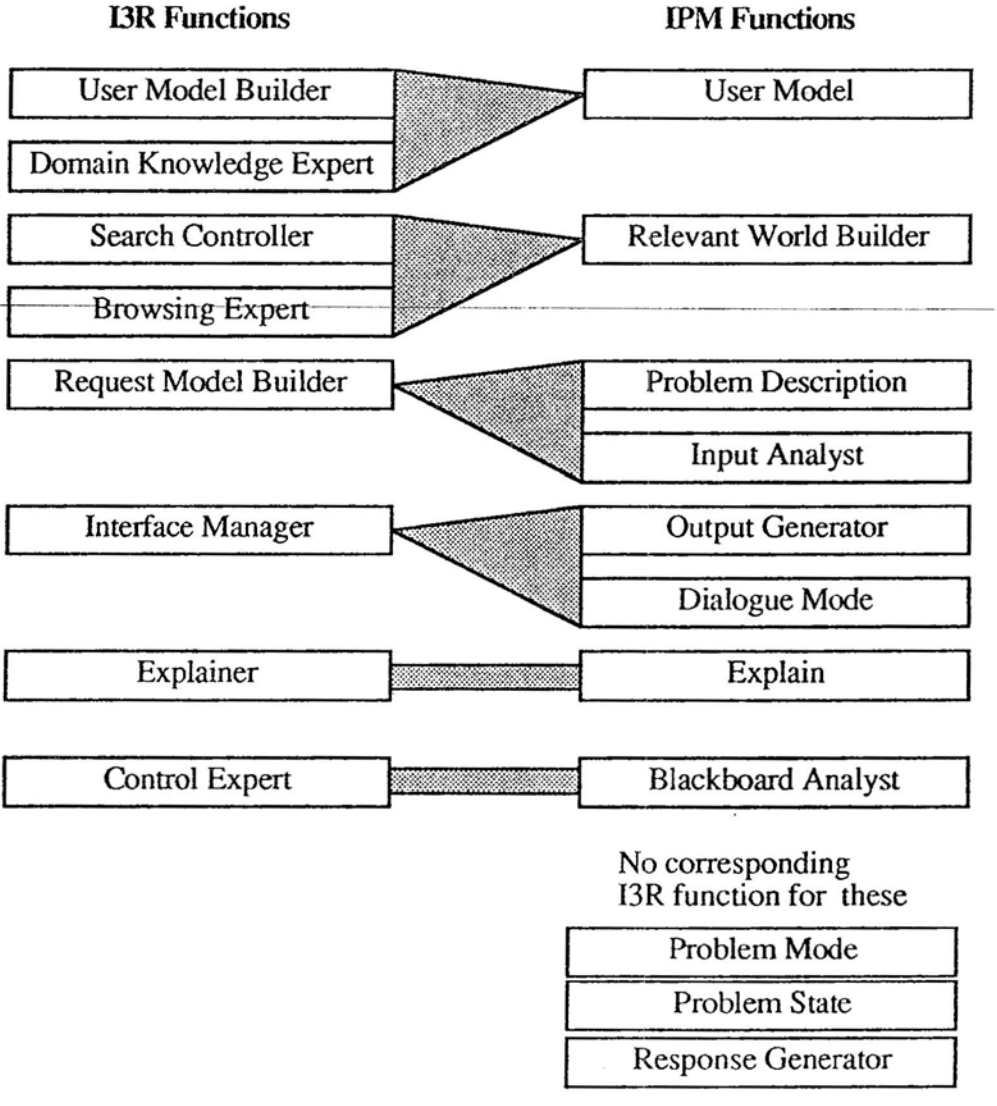

Figure 3.1: I^3R / IPM functional correspondence.

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.

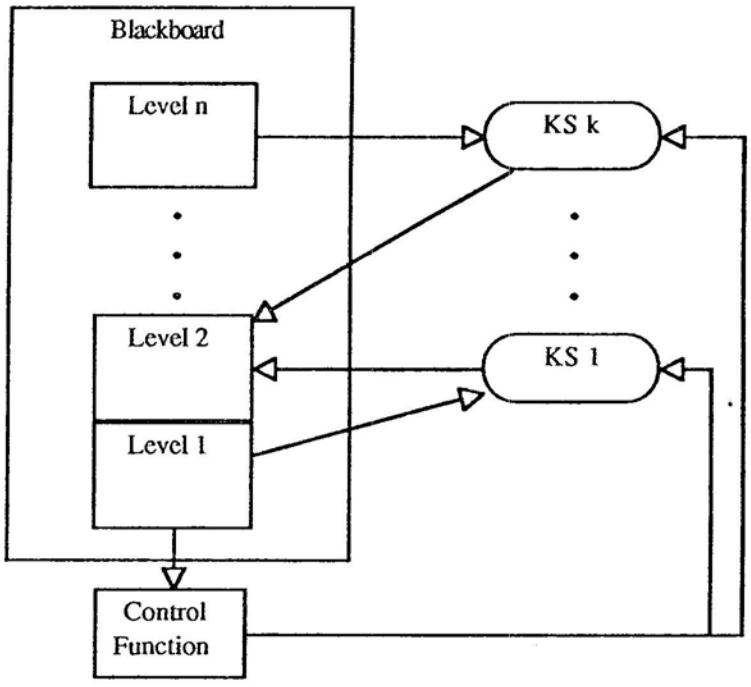

Figure 4.1: Hearsay II high level architecture.

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."

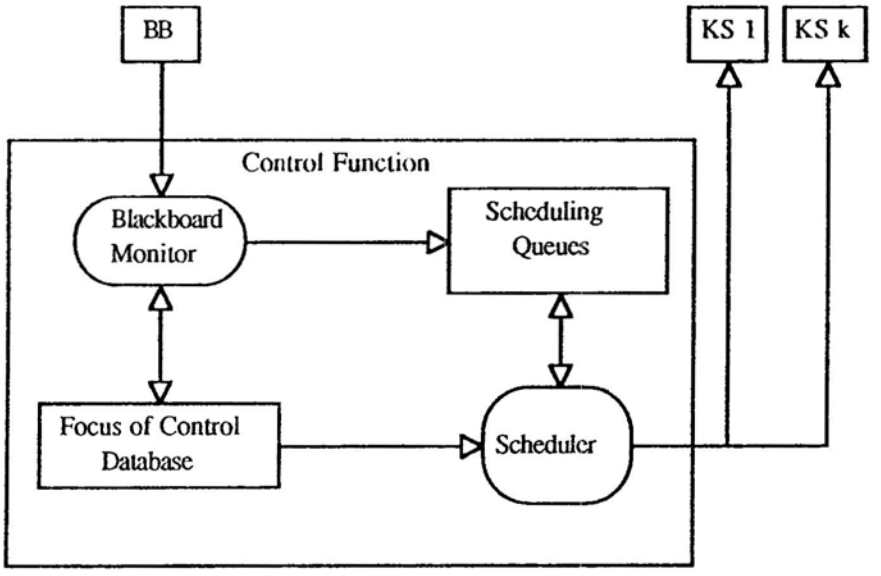

Figure 4.2: Hearsay II control function.

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.

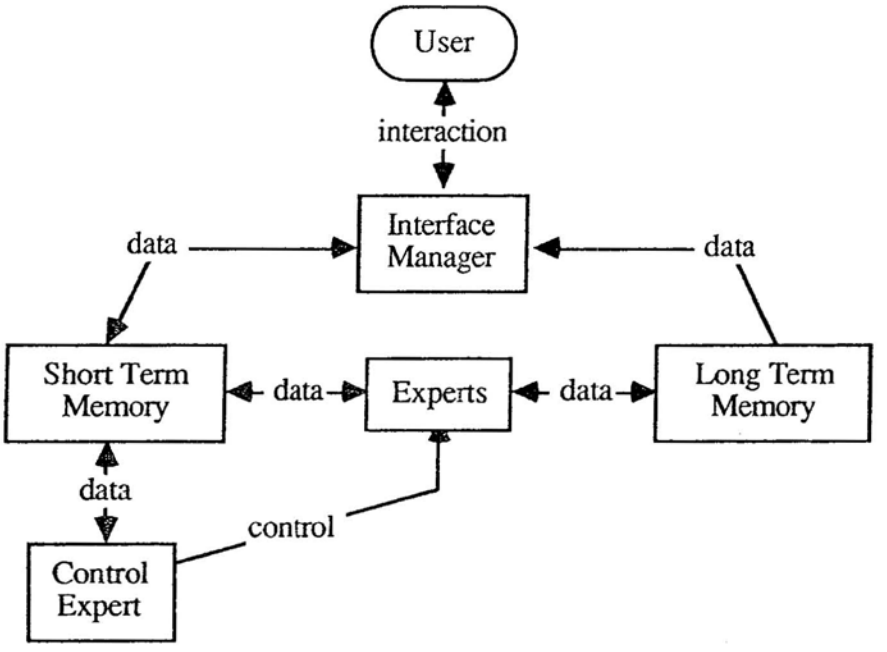

Figure 4.3: High level I^3R design.

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.

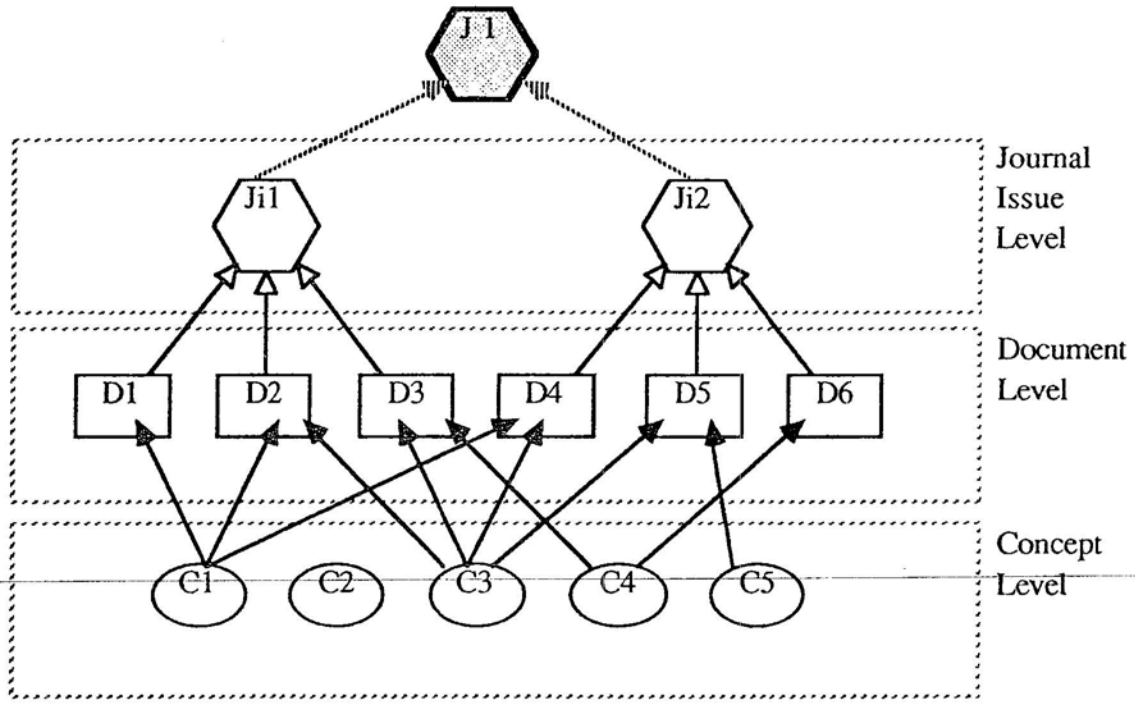

Figure 4.4: Basic structure of document collection, showing the relationships between the levels. A concept (ellipse) can be in many documents (squares) and a document can be in only one (generally) journal issue (hexagon).

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