Int. J. Man–Machine Studies (1989) **30**, 639–668

Support for browsing in an intelligent text retrieval system

R. H. THOMPSON

Hughes Aircraft Company, Ground System Group, PO Box 3310, Bldg. 636/HA226 Fullerton, CA 92634, USA

W. B. CROFT

Department of Computer and Information Science, University of Massachusetts, Amherst, MA 01003, USA

(Received 3 February 1988 and in revised form 4 June 1988)

Browsing is potentially an extremely important technique for retrieving text documents from large knowledge bases. The advantages of this technique are that users get immediate feedback from the structure of the knowledge base and exert complete control over the outcome of the search. The primary disadvantages are that it is easy to get lost in a complex network of nodes representing documents and concepts, and there is no guarantee that a browsing search will be as effective as a more conventional search. In this paper, we show how a browsing capability can be integrated into an intelligent text retrieval system. The disadvantages mentioned above are avoided by providing facilities for controlling the browsing and for using the information derived during browsing in more formal search strategies. The architecture of the text retrieval system is described and the browsing techniques are illustrated using an example session.

1. Introduction

Text retrieval systems came into existence in the 1960's, and most of the commercially available systems still reflect the design principles of those early systems. Their main emphasis was to provide access to as many documents as possible as efficiently as possible. However, the efficiency considerations were from a machine utilization and not a user perspective. Users had to be satisfied with rudimentary interfaces and limited search capabilities.

Since that time, much of the research in information retrieval has gone into forming better descriptions of documents and developing more effective ways of organizing and searching them. Two of the major results have been the development of automatic indexing techniques and search techniques based on probabilistic models (Van Rijsbergen, 1979; Salton & McGill, 1983). Not as much effort has been directed at helping the user with the *query formulation* problem. This is the problem of expressing the information need in the best possible way. 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 the documents that the user wants. 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.

0020-7373/89/060639 + 30$03.00/0

Another way of looking at this is that search is a form of inference (Van Rijsbergen, 1986). The system examines the stored documents to see if the information in the query can be inferred from a document. If so, then the document is selected for presentation to the user. Consequently, if the query is not well specified then the inference process cannot be accurate.

There are two major obstacles that the user has in expressing his information need in the best possible way. The first is the form of the query required by the system. Most systems require that the user cast his need into a formal notation using Boolean logic. Systems that use this kind of query notation have been shown to be less effective than statistical systems (Salton, Fox & Wu, 1983). Also, 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, 1983). Furthermore, the Boolean connectives AND and OR do not have a sufficiently precise meaning, which can be confusing to 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 are both required. Many systems allow the user to specify adjacency information in the query, but these additional operators can serve as a source of confusion and add further difficulty to query formulation. Some other kinds of systems allow the user to enter his query as free text, which is then processed to generate an internal form for system use. Although this is easier for the user, it suffers from the loss of information regarding relationships between words.

The second obstacle to the user is the inability to precisely express the content of the query. This has two causes. The first cause is that the user may have a definite idea of what he wants, but cannot express it in the appropriate terms because of lack of familiarity with the domain or simply forgetting the appropriate terms. A searcher that does not have a good understanding of the area in which he is working, cannot select the necessary words to define a query precisely. 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 these documents in the collection are automatically indexed, then that the 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 documents may not be retrieved.

The second cause is that the user will combine words in ways that do not make sense for the particular domain, and results from the user not having a definite idea of what he wants but only a vague notion or hunch. This is not an unusual experience. An example might be someone who wants to know about the topic of "colour." Is the user interested in the physics of coloured light, the psychological effects of colour, the physiological aspects of colour vision, or colour 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, searching the domain knowledge only represents a partial solution to the problem. The user needs to examine the content of example documents for evaluation. As he does this, he can rule out subject areas that do not apply, look more deeply into those that he feels

are relevant, or run across information that he had no idea even existed but is useful to him. This kind of search has the possibility of great variability with the final characterization of the information need bearing little resemblance to the original. In both cases the initial search will probably fail to satisfy the user's desire for information.

Since the success of finding documents to meet a user's information need depends on the quality of the expression of that need, an information retrieval system should provide ways to help a user formulate a query of the highest possible quality. The traditional way of helping the user refine his query is through a feedback process. The user makes an initial formulation and submits it to the system. The system responds with a list of documents that the user examines. From these documents, the user adds information that looks interesting and removes information that caused non-relevant documents to be retrieved. The user then submits the refined query. In some systems the user simply has to select what is and is not interesting and the system will adjust the query accordingly. This feedback process can be time consuming, since the user may have to examine large numbers of documents. This is particularly true in Boolean systems as they do not give the user ranked output. The user must examine all the documents to make sure that nothing relevant is missed.

Another way of doing this is to allow the user to browse through the information in the system's database. In this paper, we show that *browsing* is an effective way to formulate a description of an information need and can be coupled with statistical search methods to produce effective information retrieval. The rest of the paper is organized as follows. In section two, browsing is characterized and systems that have used browsing are described. Section three gives an overview of the Intelligent Interface for Information Retrieval (I^3R) system (Croft, 1986). We give a detailed picture of some parts of this system in order to show the context in which the browsing techniques operate. Section four describes how I^3R supports browsing, and in section five a sample session is presented.

2. Browsing

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

Bates (1986) points out the advantage of browsing by showing how it takes advantage of two cognitive capabilities. The first one is the greater ability to recognize what is wanted over being able to describe it. This is epitomized in the familiar saying, "I do not know what art is, but I know what I like." The second capability is being able to skim or perceive at a glance. This allows a searcher to evaluate rapidly a large amount of material, determining what is useful in it.

Browsing makes use of these abilities by showing the user examples of information that match his current model, as expressed to the interface, for evaluation. A system that allows a user to browse and to do so quickly will provide information that the user wants and may not have been able to ask for, and quite possibly all the information that the user needs. In this way, browsing addresses the second obstacle to precise query formulation mentioned earlier.

In order to determine what item will be displayed next for evaluation, the searcher utilizes heuristics. Some examples of browsing heuristics that a searcher might use in a document retrieval system are:

(1) If the current document is interesting, what else has been written by its authors?
(2) If the current document is interesting, are any of its references interesting?
(3) If the current document is interesting, are any of the documents that reference it interesting?
(4) If the current document is interesting, are any of the documents in the same journal issue, conference proceedings, etc. interesting?
(5) If the current document is interesting, are there any documents that are very similar to it in the database?
(6) If the current term is interesting, does it have any synonyms?
(7) If the current term is interesting, what documents is it in?

All of the heuristics depend on the kind of links maintained by the system. For example, if references are not maintained, then heuristics 2 and 3 cannot be used. The richer the set of links, the more ways that the user can move through the database. But by having a rich set of links, the system is then responsible for helping the user understand what the links mean, how they might be used, and where he is in the network formed by them. Without this kind of help, browsing can take on the aspect of the user finding his way through a maze, where he can become hopelessly lost and frustrated.

Browsing can also be seen as a form of constrained spreading activation (Cohen & Kjeldsen, 1987). The heuristics, in this case, constrain the choice of paths that the user selects from. Each path is evidence that one document or concept is related to another document or concept. For example, if two documents share a number of very common terms, these documents are likely to be related only on a very general level. But if one of the documents cites the other then the likelihood of them being related is greater.

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

Browsing has been used in a variety of systems. The kinds of systems include hypertext and text retrieval systems, database systems (Motro, 1985), and object

oriented programming systems (Goldberg & Robson, 1983). In the hypertext and text retrieval systems browsing has been used for both query formulation assistance and finding document information directly. Some prototype systems developed to tackle the query formulation problem are CANSEARCH (Pollitt, 1984), CALIBAN (Frei & Jauslin 1983), and CoalSORT (Monarch & Carbonell, 1987). These systems take an existing classification system and automate it to give a user online access in order to select terms for a query. The structure of the knowledge is organized as frames with various kinds of relationships, depending on the field to which the system was applied, connecting them. The main difference in the systems is the kind of assistance provided to the user. In both CALIBAN and CoalSORT the user is required to manually construct a query. However, in these systems the user is presented with a multi-window interface, so he only has to copy terms from one window to another to include them in the developing query. Once the user is satisfied with his query, he can have the system perform a formal search. CANSEARCH is oriented directly at producing a query. Consequently, it leads the user to a greater extent than the others to specify certain types of information. As the user makes evaluations and selections, the system includes them in the developing query.

There have been a number of text oriented systems that use browsing as a method of search. An early system is ZOG (McCracken & Akscyn, 1984), which is designed to be a general purpose human–computer interface. The fundamental mode of operation is menu selection with the basic unit of information being a screen-full of text called a frame. The information in a ZOG system is handbuilt using the built-in editor. The main organization is a tree, but there is no restriction on how frames may be linked. Search is done by traversing the frames until one finds the information one desires. ZOG's advantages are that it is easy to use, requires very little training, and is fast. The disadvantages stem from the lack of graphical capability. Users can get lost in the network of frames, since they have no "map" or global view to show them where they have been in the context of the entire knowledge store.

BROWSE (Palay & Fox, 1981) is a system built on top of ZOG and more oriented to document retrieval. The set of frames combined document abstracts with a concept classification hierarchy, author, and journal information. It overcomes some of the limitations of ZOG by including a partial map of the frame structure. This map is limited to only the concept hierarchy. A further improvement is the addition of a search capability. At any time the user could make a selection to perform a parameterized search of the documents. However, the query for the search still has to be manually constructed by the user.

The second kind of browsing is also found in the hypertext or dynabook systems (Kay & Goldberg, 1977; Weyer, 1982; Trigg, 1983). These kinds of systems are characterized by text units of approximately paragraph or page length connected by various kinds of links. One class of links organizes the text units hierarchically into subsection, section, chapters, and papers or books. This contrasts with traditional document retrieval systems which generally store only document abstracts. Other kinds of links provide citation referencing and editorial commentary. The TEXTNET system (Trigg, 1983) maintains over 50 kinds of links. Once the user has entered the system, he is free to meander through the network examining the text.

These systems also provide a sophisticated user interface, giving the user a number of ways to get information about the document that he is in.

A major problem with these systems is that browsing is, generally, the only way to find information in them. Either no facility or only a rudimentary one is provided to perform an associative search. This lack raises the question of the utility of these systems when the number of text units reaches the millions. Searching through that many units of text by browsing alone would be a formidable task. In one case, the dynamic book (Weyer, 1982), an index structure similar to that of a book is provided so that the user can "jump" from one place to another. However, this prototype was constructed from a history text, and therefore, the amount of information and the subject matter was constrained.

THOMAS (Oddy, 1977) was interface program that employed a different type of browsing. In this work, the system did not rely on a pre-existing complex, highly connected database of documents. Instead, a model was built by the system of the user's interest as the user evaluated the information presented to him. The model was different from the kind built by CANSEARCH in that was domain independent. This aspect of THOMAS is significant since it relieves the user of having to manually construct a query, other than the initial few terms.

Another difference from the previous characterization of browsing was that the system took the initiative, after the user entered a few initial terms, in selecting what to show the user. Even though this would seem constraining, the system was quite flexible in its interaction. It could determine when the user was not making progress, and ask him to reevaluate previously seen abstracts. It was similar to the ZOG systems in that it presented the user with only one item of information, a document abstract, at a time for evaluation.

This section has pointed to a number of requirements that a system that supports browsing must meet. First, it must support a rich set of meaningful links connecting the items it stores. It must also provide a flexible interface that can provide assistance to the user in order to keep him from getting lost in the maze of information. This assistance falls into two categories. The first is to provide context so that the user knows where he has been and where he can go. The second kind of assistance is to provide recommendations of what appears to be a promising direction to head given the user's current state. The final requirement is to provide a means of search other than browsing, so that when the user has adequately determined what he wants, he can simply have the system determine what documents might be relevant to his need.

3. System description

3.1. OVERVIEW

In this section we present I^3R, Intelligent Interface for Information Retrieval, a system designed to provide a highly flexible interface that allows a user to take advantage of a variety of means to find information in a text database. For more details see Croft & Thompson, 1987.

The organization of the system is based on the blackboard model (Nii, 1986*a*,*b*), which consists of a number of independently operating knowledge sources that work

cooperatively to solve a common problem. The problem is represented on a global data structure called a blackboard to which all the knowlege sources have access. The blackboard is composed of levels that represent different abstraction levels of the problem.

No communication between any experts is allowed. This means that no expert, with the exception of the scheduler or control expert, is aware of the capabilities of any other expert. In no sense can one expert ask another specific expert for information. Even the control expert is not aware of the specific capabilities of the other experts, but knows when to let them operate according to the information it uses to make control decisions.

The basic system cycle operates as follows. Each expert looks at the state of the problem on the blackboard and decides if there is anything that it can do to contribute to the solution of the problem. If it does, it places an instantiation of that action on the agenda for the scheduler to consider. The scheduler then determines what instantiated on the agenda is the most important or likely to advance the solution of the problem the most. That action is then performed.

One major difference between the organization of I^3R and a typical blackboard system is in the structure of the blackboard. In typical systems the blackboard is structured in such a way that it represents the problem as a hierarchy. The problem of information retrieval is not easily represented in this manner. In fact, an intelligent interface of this kind has multiple problems to solve. For example, besides the primary problem of retrieving documents likely to be relevant, the system develops a user model as a basis for adapting the interface to the specific user. Consequently, the blackboard consists of a number of places or models that represent information necessary for the operation of the system. The blackboard in this case may be more properly referred to as a short term memory (STM).

All the experts have access to a declarative knowledge-base or long term memory (LTM). This contains the document collection, the user's records, and the global domain knowledge. Communication with the user is handled by an independent Interface Manager.

This architecture allows I^3R to respond flexibly to different users in different situations by responding and adapting to situations as they arise. This kind of flexibility is required because of the great variability of users and their information needs. Studies (Brooks, Daniels & Belkin, 1985) of dialogues between search intermediaries and end users show that each interview session takes its own course. While each intermediary gathers the same basic information, how it is done and the order in which it is done varies considerably. One of the major downfalls of currently available systems is that they provide a single fixed interface to every user.

3.2. KNOWLEDGE BASE (LONG TERM MEMORY)

The knowledge base is composed of two parts, the user records and the concept/document knowledge. The user records are the system's knowledge of what a user has done in the past. It contains the essential information about past search sessions. For each session, the history information contains of the expression of the information need, the stereotypes that were applied, the request model that was constructed, and the documents that were judged relevant. Additionally, the

contents of the context and neighborhood maps may be saved at the user's request. The user records also contain a model of the user's domain knowledge. This last part is incorporated into the concept/document knowledge that the system uses when the user is looking for information.

Conceptually, the information in the concept/document knowledge is organized at three levels, the concept level, the document level, and the journal issue level. The concept level is the lowest level. Concepts are the fundamental indicators of meaning. A searcher selects concepts that he feels will best represent the information need. Authors are included at this level.

The organization of the concepts constitute 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, instance, 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, narrower,* and *instance* links represent the standard taxonomic hierarchy. *Phrase* links connect multiword concepts, like "Artificial Intelligence," to their single word constituents. *Nearest neighbour* links connects single word concepts (also referred to as terms) that are statistically similar. This relationship is based on the frequency that two concepts are used in documents together, and is computed using Dice's coefficient (Van Rijsbergen, 1979), which is:

$$2\,|A \cap B|/|A| + |B|$$

Only the most similar terms are connected by this link, and their similarity must be above the minimum threshold of 0·30. These links are divided into two types, nearest neighbour from, and nearest neighbour to, since the relationship is not necessarily symmetric. For example, term A's nearest neighbour might be term B, but term B's nearest neighbours are terms C and D.

The concept level, during a session, is a fusion of three knowledge bases, the global domain knowledge, the user's domain knowledge, and the text database. The global domain knowledge is derived from published thesaurus or other classification information. In the case of our test data, for example, it was derived from the old and new Computing Reviews classifications. The user's domain knowledge is gathered from the user as he interacts with the system. When a user is developing his query initially or is examining retrieved text, he may make connections between words. For example, the user may say that the words "distributed" and "processing" form the concept "distributed processing" and that it is related to concept "parallel programming." The information from the text database simply gives the mapping from terms to the documents in which they occur.

The next level is the document level. In this system, documents are the fundamental units of information. It is in a document that the user will find the desired information. Each document is represented by a collection of concepts that indicate its content. These are generally derived from the title, abstract, and whatever keywords are supplied by the author. The document level is, like the concept level, highly interconnected. A document nearest neighbour link; similar to the concept nearest neighbour link; is also used. It is based on the overlap of the concepts contained in two documents. Only the most highly related documents are

linked by the nearest neighbour link. Each document also contains citation links that connect it to other documents. These links are important since they represent the author's judgements of what other documents are related to the contents of their documents. These links can be used directly to facilitate finding other documents, or they can be used to generate two other kinds of similarity links, similar to nearest neighbour links. These are called bibliographic coupling links and co-citation links (Salton & McGill, 1983; Mansur, 1980). The first is based on the overlap of two documents' reference lists, and the second is based on the number of times that two documents appear together in reference lists of other documents. In our current system we were unable to implement these two links due to deficiencies in the test data. Specifically, the citation information available for each document only referenced other documents in the collection.

The journal issue level is based on the observation that many journals will have from time to time whole issues devoted to a specific topic. These issues might be the proceedings from a particular conference, or tutorial articles on a specific area. Journals also may have sections that are topic specific. This level is not highly interconnected.

The test collection used to test implementation of I^3R consists of 3204 documents taken from the Communication of the ACM, from the years 1960 to 1979. Included with this set of documents are 52 queries with relevance judgements.

3.3. EXPERTS

In I^3R the knowledge sources are called experts because each one is implemented as a rule based system and each one is intended to capture a particular kind of expertise required to make the system respond intelligently to the user. They were derived from a functional analysis of the expertise required by a system to perform as a search intermediary. An expert is composed of a condition and a group of actions to be performed when the condition is met, just the same as the rules that make up the expert. These conditions are sensitive to changes in the blackboard. An expert's condition is the union of all the component rule conditions. In each expert, conflict caused by more than one rule being enabled is resolved by choosing the rule with the most complex precondition, as measured by the number of condition clauses (or patterns) in a rule's precondition part (left-hand side).

By using a blackboard organization and by implementing each expert as sets of rules, I^3R can be easily modified either to add new large scale functionality or to fine tune existing functionality. For example, as the work on the ADRENAL project (Croft & Lewis, 1987) progresses, the natural language capability it represents can be incorporated into the system by adding a new expert with a basic capability and incrementally increasing its competence. The additional information it provides will be used by the Request Model Builder and the Search Controller, so their functionality can also be expanded.

3.3.1. Control expert (scheduler)

The CE coordinates the activities of the other experts. Its operation is based on a generalized plan derived from the analysis of dialogues between search intermediary and end users (Brooks *et al.*, 1985). By using a dialogue structure as a basis for control decisions, the operation of the system is constrained in such a way that its

operation appears logical to the user. This is vital, since the system is not a black box in which the user puts his query in at one end and gets the result out at the other. The process of searching for information requires a lot of interaction. If the system presented information to the user in a seemingly random order due to the unconstrained opportunism of the experts, the user could easily become confused.

Optimizing the utilization of computational resources is not the main purpose of the CE. This is a problem of concern to the computational structure that supports I^3R. Whether the system operates on a uniprocessor, multiprocessor, or a distributed system is of no concern as long as the response is sufficiently quick.

The difference in emphasis is a result of the difference in the kind of problem being solved. In a system like the original Hearsay II (Nii, 1986), the emphasis is on getting the correct answer. The only concern about the process of arriving at that answer is that be as efficient as possible. In a system like I^3R, the process is just as important as the answer. In fact, in text retrieval systems, the "right" answer is completely dependent on the user.

Belkin's analysis shows that in presearch interviews there are a number of phases that every interview goes through. However, the order varies considerably. In fact, the interview may pass through a phase and then return to it a number of times if the intermediary is not satisfied with the information acquired the first time. Another way of looking at this is that a search intermediary has a number of information goals that he must satisfy about the user and the information need before he can conduct a search. For example, the intermediary may make some initial assessments about the user, and then realize further on in the session that these initial assessments are in error or at least need some revision. He may then return to asking the user about some topic that was covered earlier.

In the control expert, these phases or information goals are represented by discrete states connected by transitions. There are two types of transitions, normal and exception. The normal transitions define the path through the states that a normal retrieval session should take. At each of the leaf states in the goal tree formed by the normal transitions, the experts that can operate and their relative priorities specified. The exception transitions allow the CE to take alternative paths if the session is not making good progress. The progress of the session is monitored by the CE at each cycle, and is based on values provided by the User Model Builder. These values are the number of relevant documents that the user is expected to find, and the maximum number of formal searches that is should take to find them. Figure 1 shows the state structure of the CE's plan.

The top level state describes the goal for the entire session. This is broken down into the three subgoals that represent the major phases of a session. While meeting the first subgoal, the system is determining who the user is and what stereotypes apply for this particular session. The second subgoal is where the system elicits the user to specify the information need in as much detail as possible. The third subgoal is where the system attempts to meet the need by retrieving potentially relevant documents, and where the user judges these results of the system's efforts. The second and third subgoals are further broken down into subgoals.

The exception transition from **Search For Relevant Documents** to **Characterize Information Need** is taken when the number of searches it was expected to take to find the expected number of relevant documents has failed to do so. The assumption

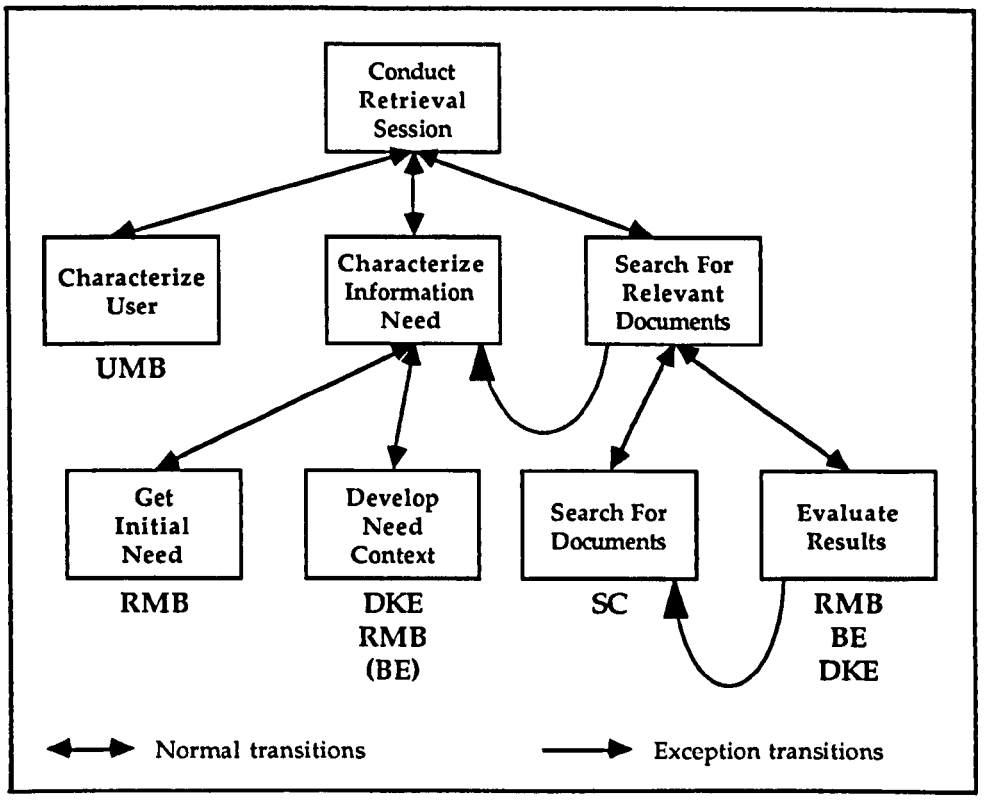

FIG. 1. Control expert's state/transition network.

here is that the query might need more refinement so that the search controller will have better information with which to conduct a search. The exception transition from **Evaluate Results** to **Search For Documents** is taken when the expected number of documents has not been found but the expected number of searches has not been exceeded. This loop implements what is normally referred to as relevance feedback.

Browsing is controlled by the browsing expert, which will be described in a subsequent section Depending on the system's categorization of the user, browsing will be available at different times. For a system novice, browsing only becomes available after the system has failed to achieve acceptable results using its formal search methods. For an expert, browsing becomes available in the **Develop Need Context** state. This requires every user to provide at least some minimal description of the information need before beginning to browse.

With the structure of the control expert, as well as the other experts in mind the basic operating cycle of the system is as follows:

(1) The control expert determines the state of the session based on the state of the blackboard, determining if it should remain in the current state or move to a new state. This determines what experts are allowed to run and their priorities for the current cycle.

(2) Each expert that is allowed to run checks the state of the blackboard to determine if it has any sets of actions it wants to perform. This procedure is simplified by a blackboard monitor which maintains information about what rules in the various experts are interested in change in the particular models. After any conflict resolution each expert places an activation of a rule on the agenda.
(3) —Execute the highest priority rule activation on the agenda. Priority is determined by the control expert's state.
—After a rule fires, the subsequent changes in the blackboard may render the conditions of the remaining rules on the agenda invalid. The preconditions of the next rule on the agenda are rechecked to see if it is still valid.
—If it is still valid, execute the actions.
—If there are any remaining rules, go to step 3b.
(4) Go to step 1

3.3.2. User model builder

The user model builder, UMB, collects information about the user that relates to the goals and aims for the particular session. This information is kept in the form of stereotypes that are established on the basis of questions answered by the user about the current session and the stereotypes established in previous sessions. The stereotypes represent information about the user's search goals, experience with the system, and experience with the search topic, and are used to define values that the control expert uses to monitor the search session. These include the expected number of relevant documents to be found, the expected number of formal searches that will be required to find the relevant documents, and the number of nodes to show in a neighborhood on the neighborhood map used in browsing. These values are summarized in Fig. 2.

3.3.3. Request model builder

The request model builder, RMB, constructs a detailed representation of the user's information need. The model is based on the probabilistic model of information retrieval (Van Rijsbergen, 1979). It records evaluations and frequency information of terms, evaluations of documents, and term dependencies that have been identified by the user (Croft, 1986).

3.3.4. Domain knowledge expert

The domain knowledge expert, DKE, is responsible for suggesting additional concepts to the user which may be relevant to the developing description of the information need, and for recording any domain knowledge, expressed as connections between concepts, that the user provides. Suggesting additional concepts is limited to when the CE in the **Develop Need Context** state. This is the only time when the DKE takes an active role in interacting with the user. When the system is in the **Evaluate Result** state it remains in the background accepting the information that the user provides and placing it in the user's domain knowledge model. Figure 3 shows the DKE asking the user to evaluate concepts it has found in the global domain knowledge model. The system uses a variety of windows to present

Search Emphasis	Domain Knowledge Expertise	
	Novice	Expert
Exhaustive	D = 15 S = 4	D = 20 S = 2
Selective	D = 5 S = 2	D = 5 S = 1
	D= #of Relevant Documents Expected S= Maximum # of Searches Expected	

System Experience	#of Nodes to be shown in Browse Map Neighborhood	# of Documents judged relevant while browsing triggering a new search
Novice	4	3 (automatic)
Expert	7	5 (confirmed)

FIG. 2. Summary of values generated by stereotype information.

information to the user and are explained in more detail in the section on the interface. The Fig. 3 has one window with two associated menus.

Figures 4 and 5 show an experienced user entering domain knowledge as the result of viewing a retrieved document. In this example, the user is identifying two phrases as related. To accomplish this, the user selects **Related** from the content menu at the left of the document, this causes the interface manager to display a related entry

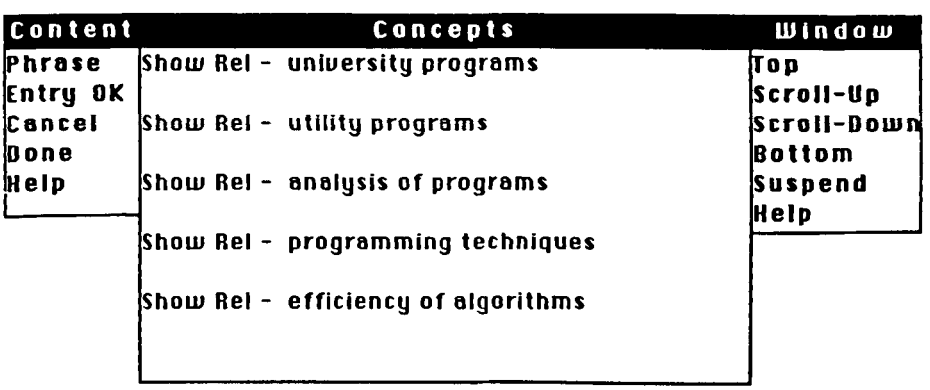

FIG. 3. DKE presenting concepts for the user's approval. The initial query contained the terms "programs" and "algorithm." The domain knowledge contained only the old Computing Review classification structure.

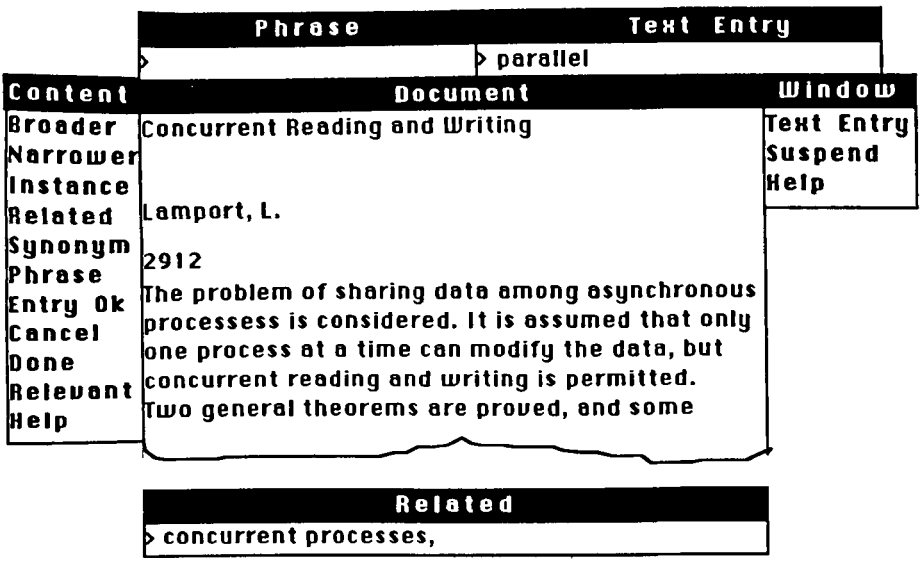

FIG. 4. Selecting a phrase from the text.

window at the bottom of the document. Since the first concept is a phrase, the user then selects the **Phrase** option from the concept menu. As the terms *concurrent* and *processes* are selected from the text of the document, they appear in the phrase entry window. If the user is dissatisfied or makes a mistake, he can select the **Cancel** option and start over; if satisfied, then he selects **Entry OK.** This causes the words composing the phrase to appear in the related entry window.

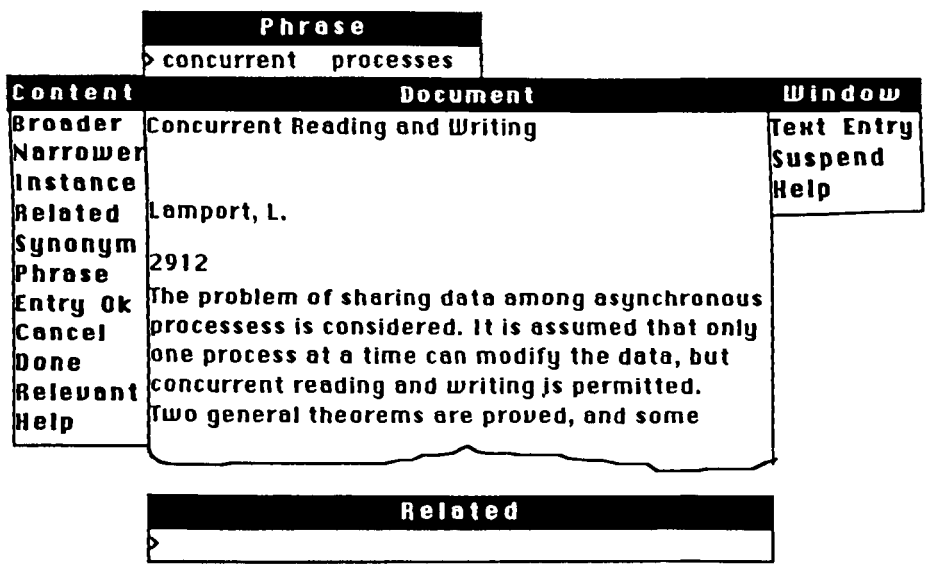

FIG. 5. Text entry to phrase window.

The second phrase is formed from one word in the document and one word that the user wants to enter. To do this, the user again selects the phrase option from the content menu. In addition, he selects the **Text Entry** option from the Window menu. This window allows him to enter a word which will be included in the phrase entry window. Once the return key is pressed the word *parallel* will appear in the phrase window, and the user can select the word he desires from the text of the document. Once the phrase is formed, **Entry OK** is selected and the phrase appears in the related entry window at the bottom of the document. Selecting **Entry OK** again signals that the process is complete. If the user had been a novice, the only kind of domain knowledge entry allowed would have been phrases composed only of words taken from the text of the document. No text entry would have been allowed.

3.3.5. Search controller

The search controller, SC, is responsible for selecting search methods for an associative search of the document portion of the knowledge base. The basic search method is based on a probabalistic model (Croft, 1986). In this model, the inference is based on frequency information of single word concepts in the collection as a whole and in individual documents to determine the likelihood that a particular document implies the query. Selection is made by ranking the documents by the likelihood and taking the top 20. The likelihood is computed according to the score:

$$\sum_i T(x_i)W(x_i)x_i + A$$

where x_i is the ith term in the set of terms describing the document, $T(x_i)$ is a weight related to the term's importance, $W(x_i)$ is a weight related to the collection frequency of term i, and A is a factor that depends on the presence of dependent groups of single word concepts in a particular document. The summation is over the single word concepts that represent the information need in the request model.

The SC currently has two alternative search methods. The first is a cluster search, which uses the same inference process as the basic search method, but instead of retrieving a single document, retrieves a cluster of documents. This cluster is defined by the document nearest neighbor links. This search is useful since it tends to retrieve different documents than the standard probabilistic search. The second alternative is a cluster search that uses citation links to define a cluster instead of the nearest neighbor links. This search is still experimental.

3.3.6. Browsing expert

The BE assists the user by suggesting paths to take in the knowledge base that are likely to lead to information relevant to the specified need. This expert will be discussed in more detail in a subsequent section.

3.4. INTERFACE

The interface to the system is handled by an interface manager that is separate from the experts and not under the control of the control expert. The interface manager communicates with the rest of the system by placing messages on and reading

messages from the short term memory. It is primarily window oriented, and the following are the kinds of windows supported by the system.

- **System Messages Window**—displays textual messages from the system to the user.
- **Choices Window**—display choices to the user which may be selected by the mouse.
- **Query Entry Window**—a window produced by a text editor that allows the user to enter a query in a variety of different ways.
- **Menus**—there are two types, window and content. Window menus allow the user to manipulate a window by scrolling the contents, suspending it, etc. Content menus let the user make choices about the content of a window. For example, getting the bibliography of a document.
- **Text Entry Windows**—these allow the user to enter character strings. Used in acquiring domain knowledge.
- **Document Window**—shows the document title, author, abstract. Also used for displaying the query after it has been entered.
- **Document List Window**—shows the titles of documents. Used for search results, bibliography lists, and citation lists.
- **Concept Window**—displays the full text of a concept as well as all of the other words it is related to.
- **Concept List Window**—displays concepts chosen by the domain knowledge expert to the user for approval.
- **Context Map**—gives a graphic "road map" of where the user has been while browsing.
- **Neighbourhood Map**—shows the immediate neighbourhood of a node in the Context Map.

4. System support for browsing

With the organization of the system in mind, we can now concentrate on the support it provides for browsing. The first aspect of the system which supports browsing is the structure of the concept/document knowledge. Since it has large variety of links, the user has many possible ways to navigate through the information. Furthermore, because of the way that the concept knowledge is fused with the document knowledge the user is given much more latitude to find information of interest. The user can explore the structure of the domain knowledge in a variety of ways. For example, he can look at all the phrases that a particular word is used in, as well as how the concepts are used in the documents. This is in contrast to the BROWSE system (Palay & Fox, 1981) where the user is restricted to the tree structure and the few cross reference links of the hand coded domain knowledge.

The browsing expert also provides assistance to the user by giving advice on paths that should lead to relevant information. This advice is given to guide and not restrict the user options, and is reflected by what is displayed on the browsing maps, which are the neighbourhood map and the context map. These consist of nodes representing concepts, documents, and journal issues connected by links marked as to their type or frequency (in the case of document to concept links). The user can always ignore the advice of the browsing expert, going off in a direction of his own choosing, and the browsing maps will be updated to reflecting the choices made.

The possible number of links that can emanate from any node, representing a concept, is potentially large. A moderately frequent single word concept may be found in as many as thirty documents even in the CACM test collection. Besides this, a concept can be linked to many other concepts by the different links. The situation for documents is similar. A document may have anywhere from just a few links, in the case of a short correspondence with one or two references, to over a hundred, in the case of a review article. The same document also may be cited by a large number of documents. This potentially large number of links cannot be effectively displayed on a graphic screen. Consequently, the browsing expert makes a choice about what nodes are most likely to be useful. The upper bound on the number of nodes is eight, due to the way that the map is organized as cells in a square grid. The actual number of nodes shown depends on the connectivity of the node and the user model. A novice user will have fewer nodes presented to him than an experienced user. The selection of these nodes in based on heuristics about the importance of the evidence that each kind of link represents. This importance is reflected by a weight associated with each link type. The browsing expert takes the highest weighted nodes and displays them on the map.

The browsing heuristics make use of the evidence provided by the links and the request model to recommend nodes that are strongly related to the node currently in view. Those that are related by multiple pieces of evidence are selected as the most promising to examine. None of the heuristics infer recommendations by looking more than one link away. The heuristics are divided into two kinds, those that deal with concepts and those that deal with documents. The concept heuristics contain one underlying assumption which is that since documents are the fundamental units of information in the system, the user should be directed towards documents rather than other concepts. The most important links between concepts are the *nearest neighbour* and the *synonym* links. In the domain knowledge there should be relatively few synonym links, since these represent a very strict view of synonymy. Similarly, for any single word concept there should be few nearest neighbours, since only at most the top five most similar ones are saved (see Section 3.2). The following is the method of choosing concepts.

(1) Retrieve concepts connected by nearest-neighbour-to links. These are concepts on the nearest neighbour list of the *current* concept, and are given a weight of 3.
(2) Retrieve concepts connected by nearest-neighbour-from links. These are concepts that have the current concept on *their* nearest neighbour list, and also are given a weight of 3.
(3) Retrieve concepts connected by *synonym* links. These are given a weight of 2.
(4) Retrieve concepts connected by *related-to* links. These are given a weight of 1.
(5) Retrieve concepts connected by *phrase* links. These are given a weight of 1.
(6) For any term that has been marked relevant and occurs in documents that have not been viewed add an additional weight. This weight varies depending on the frequency of the term in the collection. Terms occurring in 5 or fewer unviewed documents are given a weight of 2, those occurring in 10 or fewer are given a weight of 1, the rest are given no extra weight.
(7) If there are any tied scores, order by document frequency giving higher preference to those in fewer documents.

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

While a user is viewing a concept, he may decide to look at the documents in which it occurs. The BE will act differently depending on the frequency of the concept in the collection and the user specific values determined by the UMB (refer to Fig. 2). The selection of the documents to display as node is determined by the frequency of the concept in the document and by the date of the document; the newer is given preference over older. If the number of documents is less than the numbers of nodes specified by the user model then all of the documents are displayed as nodes connected to the concept (Fig. 6b) and the links are marked by the concept frequency. If the number of documents is more than the number specified by the UMB or if there is no room around the node, then some of the documents are displayed as individual nodes and the remaining are represented by a single node (box) that is marked with the number of documents it represents (Fig.

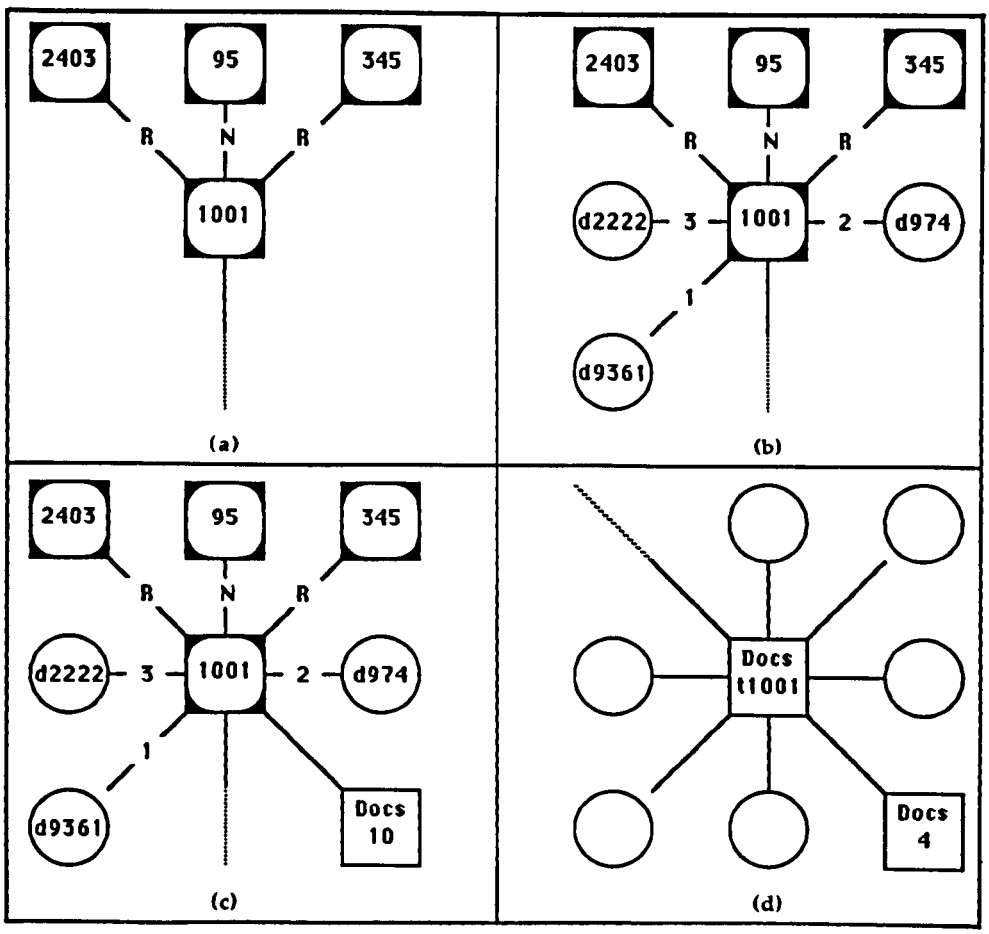

FIG. 6. Example of term neighborhoods.

6c). If the user decides to examine the documents represented by the box node, then that node is extended and as many document nodes as possible are shown with the remaining documents again being represented by a box node (Fig. 6d). If the concept occurs in more than 15 documents and the user is a system expert, the system asks if he really wants to look at that many documents. If so, then the box nodes are expanded as before. If the user is a system novice, he simply is not allowed to look at that many documents, so a message informing him of that fact will be displayed in the system messages window.

The document heuristics are similar to the concept ones. The most important links are the nearest neighbour links (to and from, both given a weight of 3). After this is the citation link (given a weight of 1), and then the reference link (given a weight of 1). However, since a document can potentially have many references and be cited many times, an evaluation of the documents connected by the citation links must be made. A document that is cited many times is given a higher value than one cited few times. This is based on studies of citation patterns (Salton & McGill, 1983) indicating that citation of a document is a relatively rare occurrence. Consequently, a document that is referenced many times is significant. The quantification of this significance for the browsing heuristics is:

(1) Cited 2 to 3 times, an additional weight of 1.
(2) Cited 4 to 5 times, an additional weight of 2.
(3) Cited 7 to 9 times, an additional weight of 3.
(4) Cited 10+ times, an additional weight of 4.

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

An evaluation of a document by the size of the reference list is not so straightforward. A document with a large reference list may be good if the user is a novice in the domain of interest, since it might be assumed that the article is a survey of some field. However, a document that has only a few references may also be valuable, if it cites the current document of interest. This implies that the citing document may be closely related. If a document with few references is cited by the current document nothing can really be determined about its potential value. So, if the user is a domain novice, a document with a large (10, because of the test set) reference list is given an additional weight of 2.

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

The user is not limited to viewing the nodes that are displayed on the neighbourhood map. When he selects a node for viewing from the map, a window containing the textual information appears. He may decide, for example, to examine the reference list of a document node. A list of titles appears, from which he can choose any one. If he chooses one that is not on the maps, a node will be put up

representing the document chosen. In this way, the map always maintains the path that the user has taken. Should the user care to backtrack, he can scroll the window back over the node of interest, select it for view, and move in a new direction. These maps can be saved, so if a user wants to continue at a later time, he can resume previous search.

5. Example browsing session

In this section we present a representative retrieval session to demonstrate how the system supports browsing. We will not describe in detail all of the system's activities, just those that are relevant to browsing. The session begins in Fig. 7 with the user identifying herself to the system. From this identification, the user model builder examines the long term memory to see if it has information about her. In this case, the user is new, so the system initializes with empty models.

At this point, the UMB will ask some questions to assess the user's level of expertise in the use of this kind of system, expertise in the domain she is searching, and in her interest in either an exhaustive or selective search. From these questions, the stereotypes are established.

In Fig. 8 the system has displayed a selection menu asking the user to select how she wants to enter her query. The user has responded by indicating a full text query.

The system provides an editing facility to let the user enter her query, the text of which is:

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

This text will be processed by the RMB to extract terms from it. The DKE will ask the user to highlight words that are important and to enter any domain knowledge connections that may exist between query terms. It will also look in the global domain knowledge for connections to other words that might be related to those that the user has entered. If the user had used the system previously, her domain knowledge model would be examined for possible relevant concepts. Upon approval, they will be added to the request model. This set of actions constitutes the initial query formulation phase.

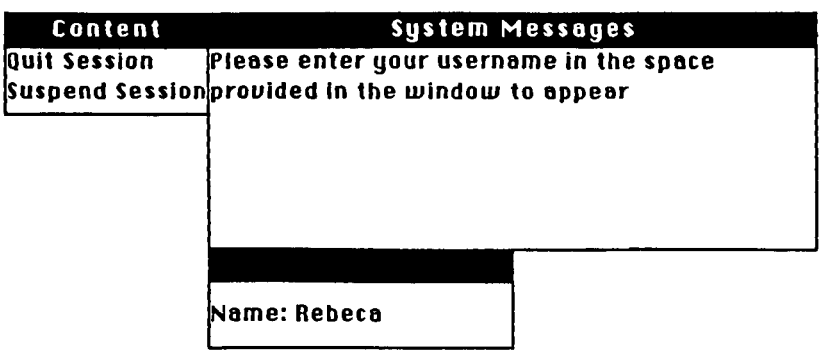

FIG. 7. User identifies herself to the system.

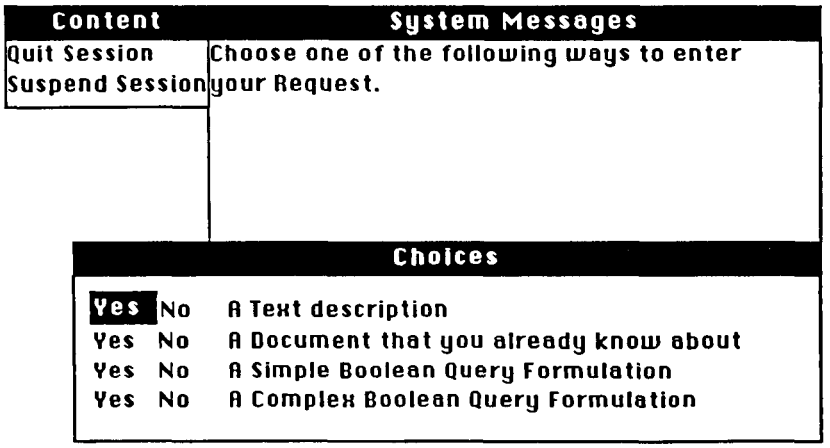

FIG. 8. User chooses text input for query.

At this point, the SC conducts a search, returning a list of documents. Figure 9 shows the results. The user now looks at the list of documents to make relevance judgements.

The left hand bar charts indicate the relative likelihood that a document is relevant. The highest ranked document is used as the basis for measurement. This gives the user a way of assessing the potential value of a document before actually examining its contents. The user can select a document for examination by choosing the **Show** option. This will cause a window to appear containing the authors, title, and abstract. From this window she can make judgements regarding the relevance of

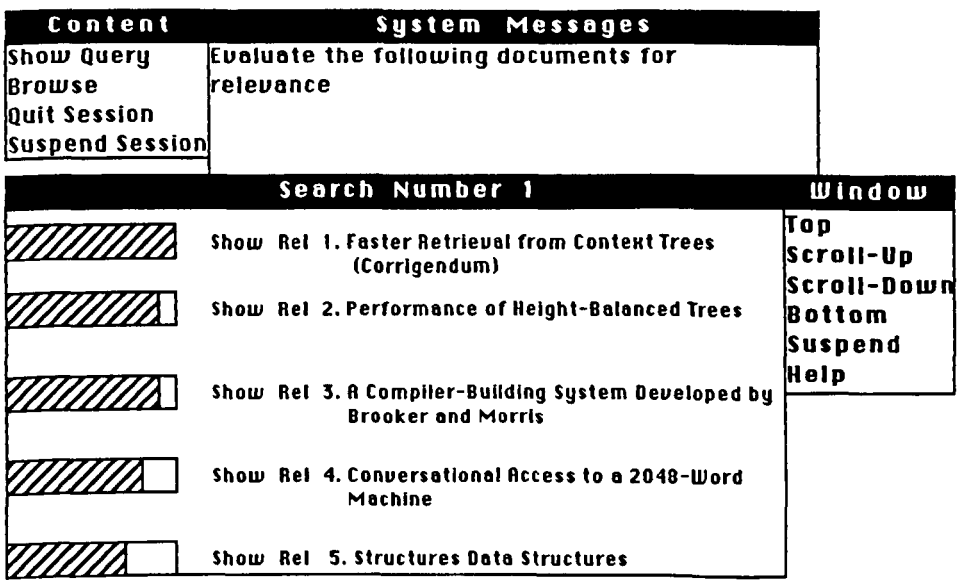

FIG. 9. First search results.

terms in the same way that she did when refining the query. By selecting the **Rel** option the user indicates that a document is relevant. After examining the list of titles, the user goes back to the first document that was judged relevant. She selects the choice **Browse** from the content window attached to the system messages window. The availability of browsing varies depending on the system's assessment of the user. An experienced user has the option of browsing early in the course of a session, whereas a novice can only browse after an initial search has been made.

Selection of the **Browse** option tells the system that the user is done with the search results. The interface manager sends the evaluation of the search to the experts for their use. This causes the RMB to update the request model by adding new terms from the documents that were marked relevant by the user. It also causes the SC to evaluate the results of its performance, which determines, in part, what search method will be used the next time it invokes a search during this session. If she has made any connections between terms the DKE will establish them in the user's domain model.

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

Commencement of browsing is indicated by a message in the system messages window and by the appearance of the neighbourhood and context maps on the screen. Figure 10 shows the initial configuration of the neighbourhood map. In the centre of the map is the document that the user selected as relevant when she was examining the retrieval results. The surrounding nodes are documents that the browsing expert has decided are likely to be interesting, with one node being

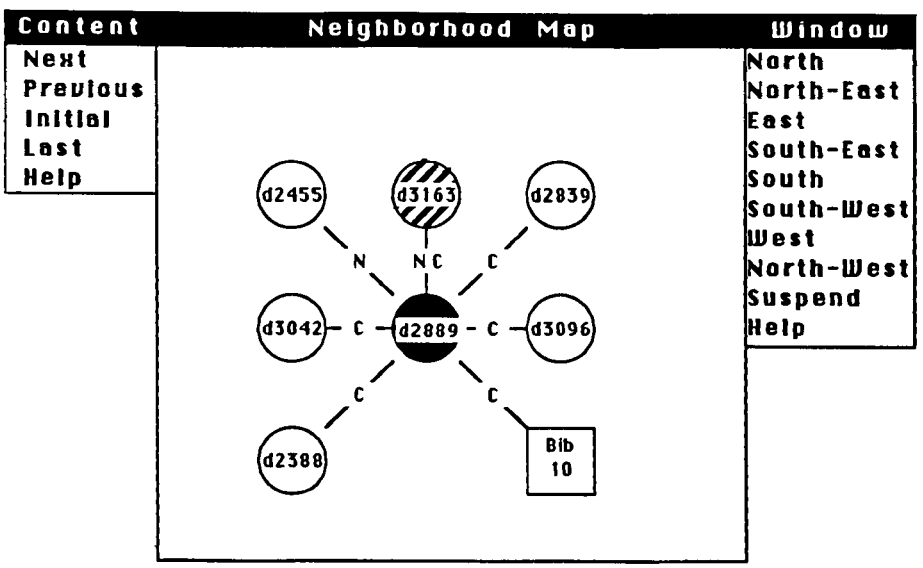

FIG. 10. Initial display on the neighbourhood map (the document 2889 and context window are not shown).

marked as the most likely to be interesting. The boxes represent lists of documents that cannot be shown but are related to the current document. The relationship is marked on the link and the number of item that the box represents is in the centre of it. The menu to the right labelled **Window** allows the user to scroll around the context map, since it is really a viewport on a much larger map display. The menu on the left allows the user to trace the path that she has chosen. The selections mean:

- **Next**—select the recommended node for viewing.
- **Previous**—go to the previously viewed node.
- **Initial**—go the first node displayed.
- **Last**—go to the last relevant node.

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

The context map is a smaller scale version of the neighbourhood map. The links are still marked as to type, and the nodes marked as to whether they have been recommended, visited, or judged relevant. No node identification information is given since that is available from the neighbourhood map. Both maps remain in correspondence with each other, centered on the same node. When the user selects a node in the neighbourhood map, the context map is updated at the same time, and vice versa. An example of a context map is Fig. 17.

When the user selects one of the nodes with the pointing device, the document that it represents appears on the screen (Fig. 11) and the node is shaded to indicate that it has been visited. When the user indicates that a node is relevant and says that she is done evaluating it the evaluations are added to the request model and any domain knowledge is added to her domain knowledge model. After a number of documents are judged relevant the system will, depending on the user stereotype, seek to initiate a search. If the user is a system novice, as well as a domain novice, the search controller will automatically be given control to initiate a search. If the user is system expert, the system will indicate, via the system messages window and choice selection window, that it can make a search based on the new information and requests the user's permission to do so. The number of new documents judged relevant that causes a search to be performed is determined by stereotypes set by the UMB.

The neighbourhood of a document is not determined unless the user judges it to be relevant or selects the **Expand** option from the content menu associated with the document text window. If either of these choices is made, the node is redrawn farther away to provide space for its neighbourhood. The user selects the **Expand** option and the map is redrawn showing the neighbourhood of the selected document (Fig. 12).

The user is not restricted to selecting documents that appear on the neighbourhood map. He may go back to the current document of interest and select any list of documents associated with it. These are the bibliography list, the citation list, the author list, or, the journal issue list. Figure 13 shows the neighbourhood map after the user has examined the recommended node and has decided to examine the other node connected by the nearest neighbour link, marked **N**.

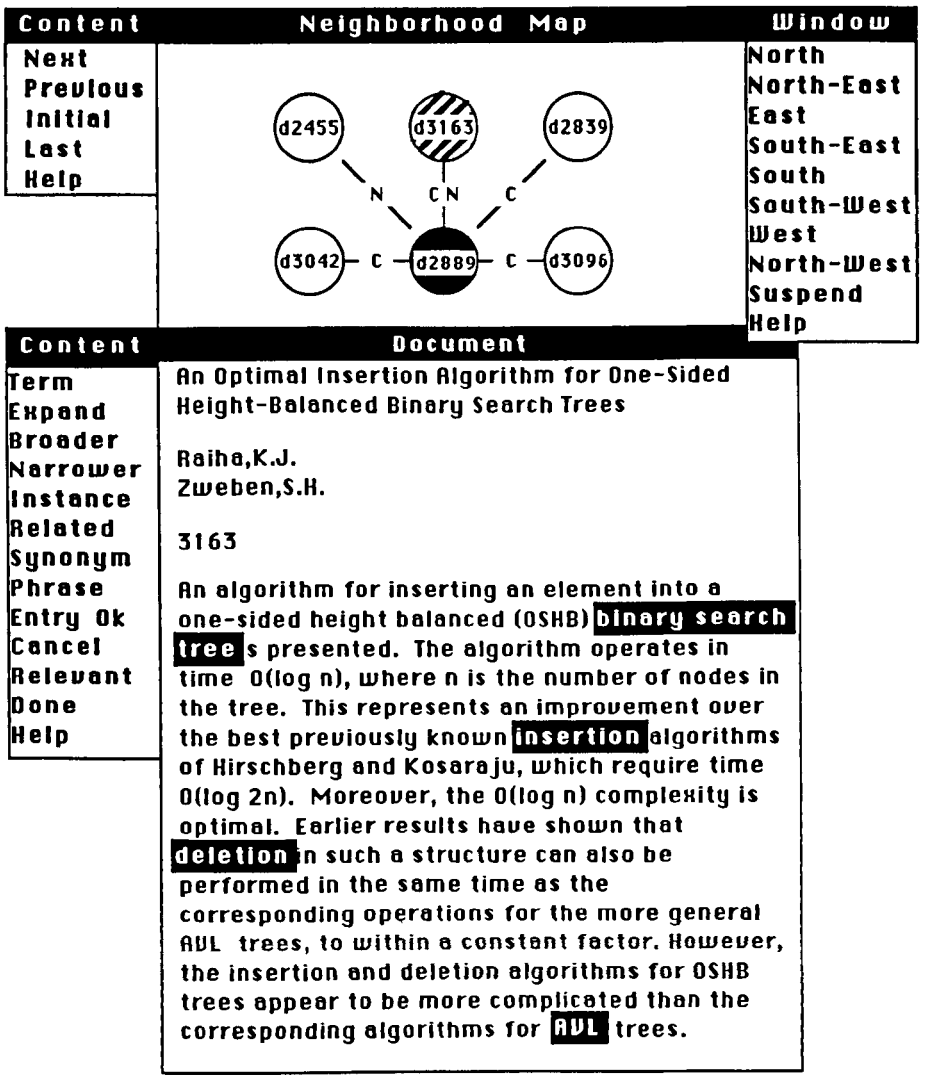

FIG. 11. User selects a recommended node-to view its contents, and selects terms that are particularly relevant or interesting.

The user is not restricted to just moving from document to document, but may also examine concepts. In Fig. 14, the user has decided that the term "multidimensional" is possibly a fruitful path to follow. She selects the **Term** option from the content menu and then the term of interest. This tells the system that she is selecting this as a node to examine and not that it is just an interesting term. The node icon with the term's number appears. The link is marked with the number of occurrences of that term in the document. Figure 15 shows the display for the term. In this case, there is as of yet no connection to any other concepts. The user could add synonym phrases like "*n*-dimensional" at this point. The content menu shows the different

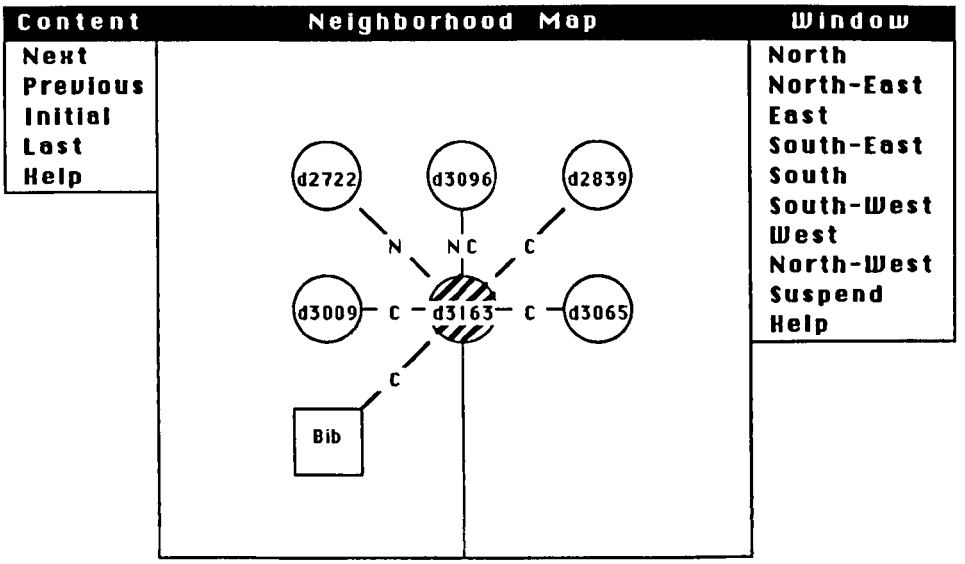

FIG. 12. Neighbourhood map with expanded document neighbourhood.

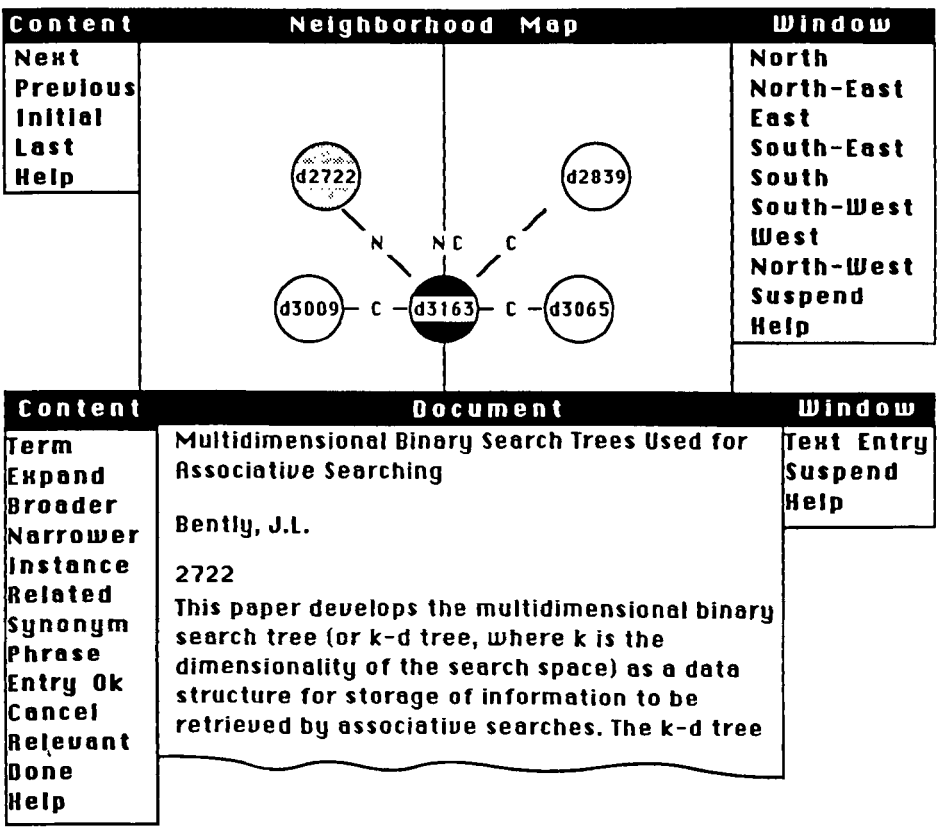

FIG. 13. User views document 2722 (text is incomplete).

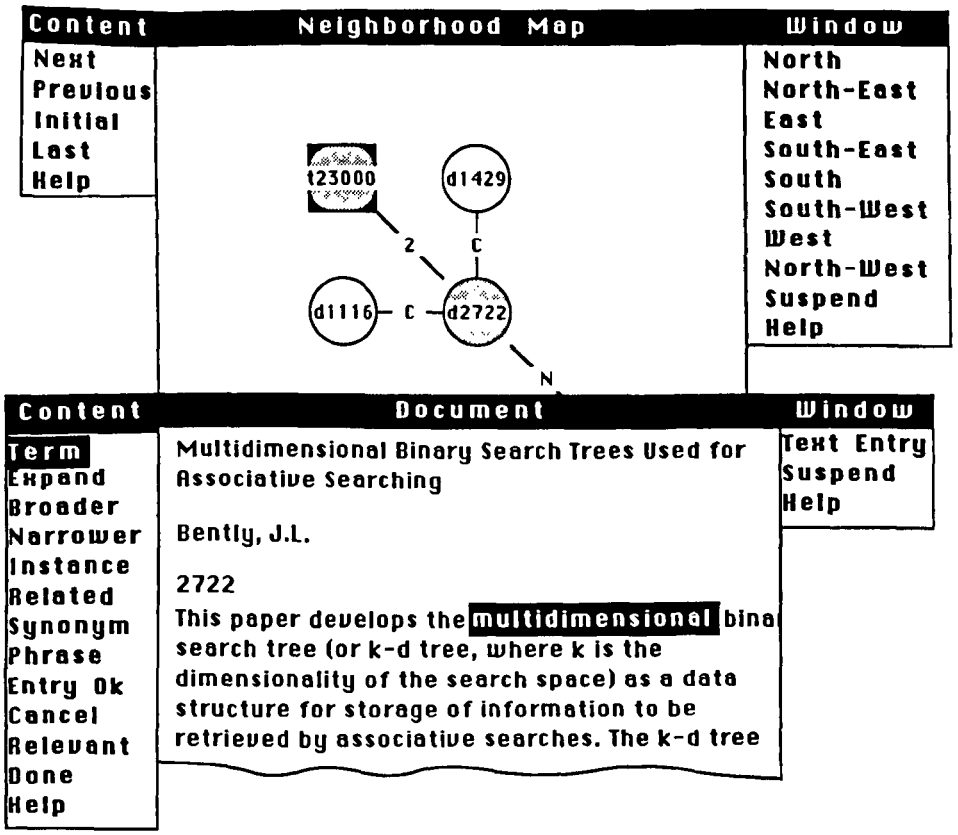

FIG. 14. User selects a term to examine from document 2722.

possibilities for domain knowledge entry. The **Select** option allows the user to follow a concept in the same way that the **Term** option was used in a document display content menu. The **Documents** option tells the system that she would like to see the documents that are associated with this term, where the **Expand** option would signal to expand the neighbourhood using concept links.

Upon selecting the **Documents** option, the browse maps are both expanded and a document list similar to the search results list appears (Fig. 16) with the titles of the documents that have the term "multidimensional" in them. The user selects the seventh document as an interesting one since it has the term "tree" in it. In this document, she sees the term "trie," and decides to find out what documents are connected to it. In the collection, there are only three, but one of the remaining two is interesting.

Figure 17 shows the context map that results from the session as it has developed so far. From this map, the user can go to any node by scrolling it into view and selecting it. This will cause the neighbourhood map to be redrawn with the new node of interest and its immediate neighbourhood. For example, in this session the user might decide that she wants to continue investigating the area around the second she determined to be relevant (marked with an "*a*"). One problem that may arise in this

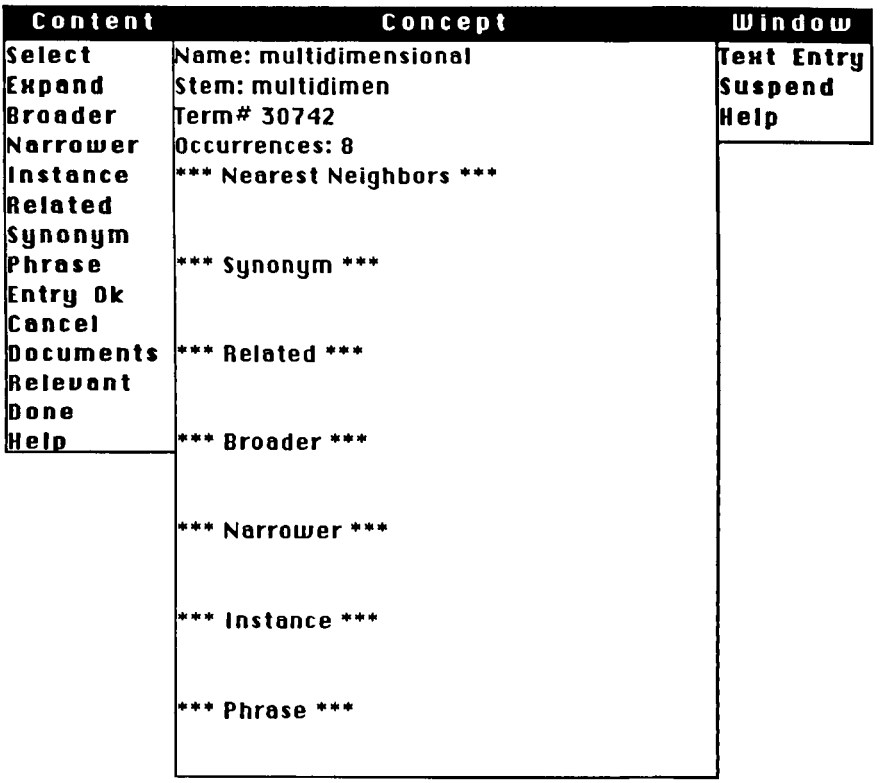

FIG. 15. Display for the concept "multidimensional".

session occurs if the user decides to expand in the direction of the node marked "*b*". If the user would take the recommended node from the expansion of the marked node, it would overwrite the term node representing the term "trie." The interface manager attempts to avoid this by expanding in a direction that appears to be clear, such as the location marked "*c*". If this location was taken, the system would draw

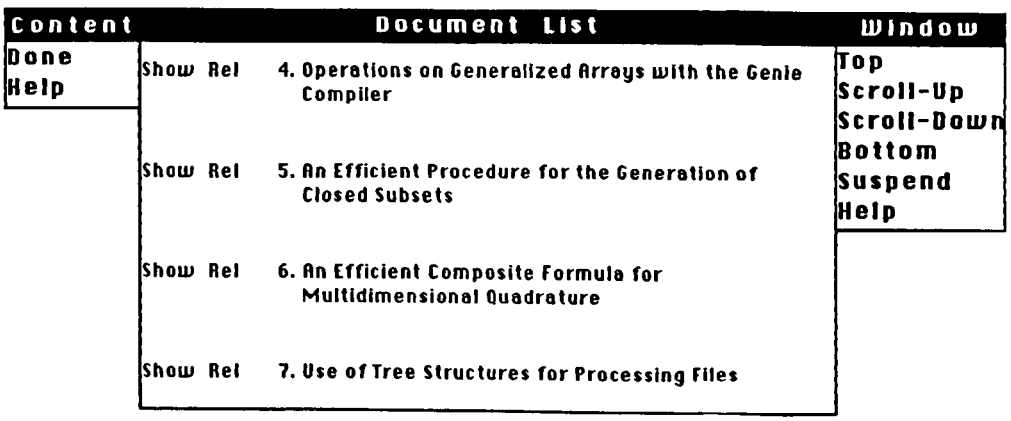

FIG. 16. User selects "documents" option.

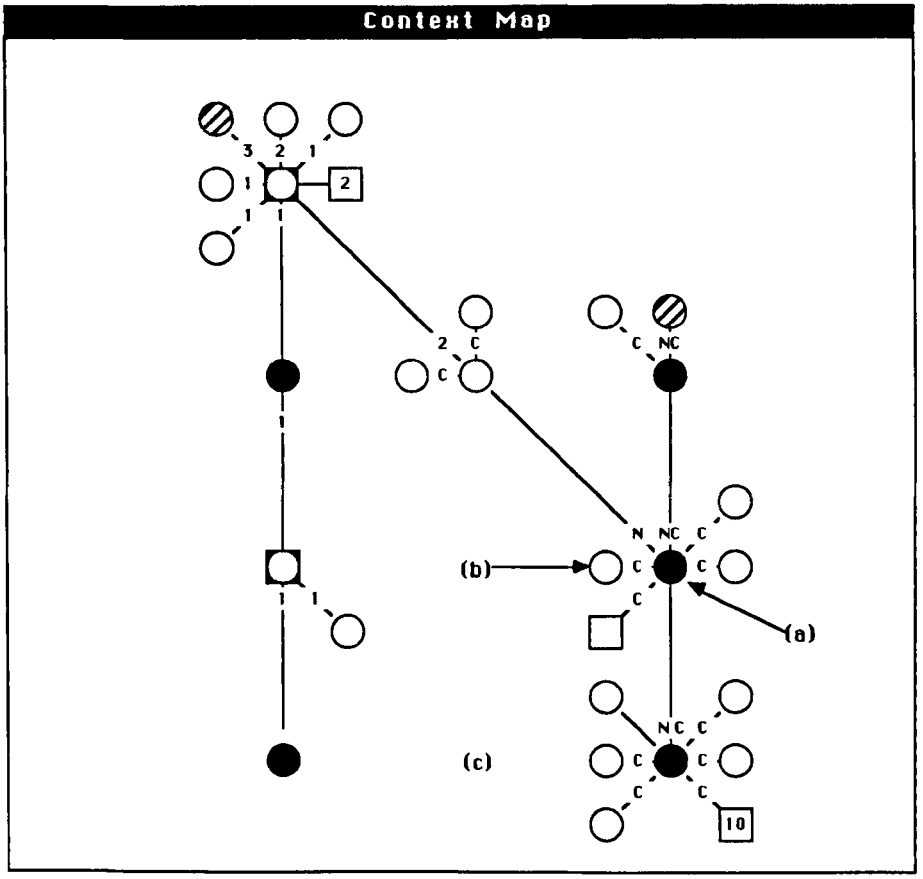

FIG. 17. Context map after examining document 2846 (menus not shown, but are the same as those with the neighbourhood map).

the node in another region of the map and place a connector icon on a link leading to it. Selecting the connector would put the map back in the region where the node originally was drawn.

6. Conclusion

We have described a system that demonstrates the feasibility of using diverse kinds of knowledge to provide, as an intelligent interface, assistance to help users find information in a textual database. One of the major ways it helps the user is to allow him to browse the knowledge base. While the user does this, the system records the evaluations made of concepts and documents refining its model of the information need. The refined model can then be used to infer new documents for evaluation. The system helps the user while browsing by making recommendations of concepts and documents that, based on evidence derived from knowledge of the document collection, global domain knowledge, user domain knowledge, and

request model appear to be strongly related to the user's information need. The system also provides visual assistance by means of the context and neighbourhood maps so that a user can arbitrarily move to concepts and documents for display and evaluation.

I^3R is implemented in Common Lisp on a DEC VAXStation II under VMS. Nearly half of the code is devoted to the operation of the interface manager. A little under 40% implements the actions and predicates that constitute the rules, and the remainder supports the system architecture. The total number of rules at the moment is about 250. The browsing expert has about 70 of those.

Further developments in I^3R lie in the utilization of natural language processing techniques. The primary area of interest is in indexing of both the query and the documents. Conceptual indexing would provide better evidence for inference than the current single word methods. Another use is to analyse the value of citation links. A citation may be made for a variety of reasons. It may point to alternative views of a topic, to a synthesis method, or similar work to name just a few of the possibilities. Some citations will be more important than others depending on the user's current interest and expertise. By being able to categorize citations the browsing expert will have more information at its disposal for assisting users.

References

BATES, M. J. (1986). Terminological assistance for the online subject searcher. In *Proceedings of the Second Conference on Computer Interfaces for Information Retrieval*, p. 7. Defense Technical Information Center, Cameron Station, Alexandria, Virginia, 1986.

BROOKS, H. M., DANIELS, P. J. & BELKIN, N. J. (1985). Using problem structures for driving human-computer dialogues. *Proceeding of RIAO-85, Recherche d'Informations Assistee par Ordinateur.* Grenoble, pp. 645–660.

COHEN, P. R. & KJELDSEN, R. (1987). Information retrieval by constrained spreading activation in semantic networks. *Information Processing and Management*, **23**, 255–268.

CROFT, W. B. (1985). User-specified domain knowledge for document retrieval. In *Proceedings of the International Conference on Research and Development in Information Retrieval, Pisa*, pp. 201–206.

CROFT, W. B. (1986). Boolean queries and term dependencies in probabilistic retrieval models. *Journal of the American Society for Information Science*, **37**, 71–77.

CROFT, W. B. & LEWIS, D. D. (1987). An approach to natural language processing for document retrieval. In *Proceedings of the Tenth Annual International ACM SIGIR Conference*; *Research and Development in Information Retrieval*, New Orleans, pp. 26–32.

FREI, H. P. & JAUSLIN, J, -F. (1983). Graphical presentation of information and services: a user-oriented interface. *Information Technology*: *Research and Development*, **2**, 23–42.

GOLDBERG, A. & ROBSON (1983). *Small talk–80.* Reading, Mass: Addison-Wesley.

KAY, A. & GOLDBERG, A. (1977). Personal dynamic media. *IEEE Computer*, **10**(6), 31–41.

MANSUR, O. (1980). On the selection and combining of relevance indicators. *Information Procedure and Management*, **16**, 139–153.

MCCRACKEN, D. L. & AKSCYN, R. M. (1984). Experience with the ZOG human–computer interface system. *International Journal of Man–Machine Studies*, **21**, 293–310.

MONARCH, I. & CARBONELL, J. (1987). CoalSORT: a knowledge-based interface. *IEEE Expert*, **2**, 39–53.

MOTRO, A. (1985). *BAROQUE*: *an exploratory interface of relational databases.* Department of Computer Science, University of Southern California, Los Angeles, California (unpublished manuscript).

NII, H. P. (1986*a*). Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architectures. *AI Magazine*, **6**(1), 38–53.

NII, H. P. (1986*b*). Blackboard systems: blackboard application systems, blackboard systems from a knowledge engineering perspective. *AI Magazine*, **6**(2), 82–106.

ODDY, R. N. (1977). Information retrieval through man-machine dialogue. *Journal of Documentation*, **33**, 1–14.

PALAY, A. P. & FOX, M. S. (1981). Browsing through databases. In *Information Retrieval Research*. Guildford: Butterworths.

POLLITT, A. S. (1984). End user touch searching for cancer therapy literature—a rule based approach. In *Proceedings of the Sixth Annual International ACM SIGIR Conference*; *Research and Development in Information Retrieval*. **17**(4), 136–145.

PORTER, M. F. (1980). An algorithm for suffix stripping. *Program*, **14**, 130–137.

SALTON, G., FOX, E., WU, U. (1983). Extended Boolean information retrieval. *Communications of the ACM*, **26**, 1022–1036.

SALTON, G. & MCGILL, M. J. (1983). *Introduction to Modern Information Retrieval*, New York: McGraw-Hill.

TRIGG, R. H. (1983). *A network-based approach to text handling for the scientific community*. Technical Report no. 1346, Department of Computer Science, University of Maryland, College Park, Maryland.

VAN RIJSBERGEN, C. J. (1979). *Information Retrieval*. Boston, MA: Butterworths.

VAN RIJSBERGEN, C. J. (1986). A non-classical logic for information retrieval. *Computer Journal*, **29**, 487–485.

VIGIL, P. J. (1983). The psychology of online searching. *Journal of the American Society for Information Science*, **34**, 281–287.

WEYERS, S. A. (1982). The design of a dynamic book for information search. *International Journal of Man–Machine Studies*, **17**, 87–107.