Search Improvement via Automatic Query Reformulation

SUSAN GAUCH and JOHN B. SMITH
University of North Carolina

Users of online retrieval systems experience many difficulties, particularly with search tactics. User studies have indicated that searchers use vocabulary incorrectly and do not take full advantage of iteration to improve their queries. To address these problems, an expert system for online search assistance was developed. This prototype augments the searching capabilities of novice users by providing automatic query reformulation to improve the search results, and automatic ranking of the retrieved passages to speed the identification of relevant information. Users' search performance using the expert system was compared with their search performance on their own, and their search performance using an online thesaurus. The following conclusions were reached: (1) the expert system significantly reduced the number of queries necessary to find relevant passages compared with the user searching alone or with the thesaurus. (2) The expert system produced marginally significant improvements in precision compared with the user searching on their own. There was no significant difference in the recall achieved by the three system configurations. (3) Overall, the expert system ranked relevant passages above irrelevant passages.

Categories and Subject Descriptors: H.1.2 [**Models and Principles**]: User/Machine System—*human factors*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*search process*; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems

General Terms: Human Factors

Additional Key Words and Phrases: Expert Systems, full-text information retrieval, online search assistance, query reformulation, textbases

1. INTRODUCTION

1.1 Driving Problem

Technological advances are causing a revolution in information retrieval. Optical character recognition, word processors, and computer publishing software are capable of producing massive quantities of online text. The development of optical storage media is making the storage and distribution of large collections of online text feasible. Proliferation of personal

This work was supported in part by ONR contract N00014-86-K-0680. Some of this research was presented at the SIGIR Conference on Research and Development in Information Retrieval held at Brussels, Belgium in September 1990.

Authors' address: College of Computer Science, Northeastern University, Boston, MA 02115.

workstations, combined with modems, are allowing an increasing number of end-users to do their own searching of online databases. *Textbases*, online full-text databases, are becoming more common. All these trends lead to end-user's performing their own textbase searches.

The main roadblock to widespread use of online textbases will soon be the inability of end-users to search effectively. Borgman [5] identifies two types of knowledge necessary to search: knowledge of the mechanical aspects of searching (e.g., syntax and semantics of both the query language and the system interaction commands) and knowledge of the conceptual aspects (e.g., ways to broaden and narrow searches using alternative vocabulary, choosing alternative search paths). She summarizes the results of many different user studies, concluding that, whereas system mechanics are rarely a problem for any but very inexperienced and infrequent users, even experienced searchers have significant problems with search strategy and output performance.

User difficulty with search strategy shows up in many different studies on searching online bibliographic databases. Fenichel [15] finds that even experienced searchers could improve their search results. The searchers lost sight of the search logic, missed obvious synonyms, and searched too simply. Search performance is often measured by recall, the ratio of relevant documents retrieved to the number of relevant documents in the entire database. The searchers were satisfied with 51 percent recall on average, indicating that almost half of the relevant information was not retrieved. The lack of successively refining queries, called *iteration*, is another problem identified. In spite of the low recall, half of the searchers never modified the original query in an attempt to improve their results.

Studies of inexperienced searchers find even more problems with search strategy. In one study [6], a quarter of the subjects were unable to pass a benchmark test of minimum searching skill. In another experiment [18], contrasting the searching of novices versus experienced searchers, the novices found some relevant documents easily, but they failed to achieve high recall and were unable to reformulate queries well. The experienced searchers in this study were more persistent and willing to experiment than the novices.

Blair and Maron [3] paint an even bleaker picture for searching full-text databases. Legal assistants searching a legal database achieved only 20 percent recall, although they were attempting to do a high recall search. The factors, as identified by the authors, leading to this poor performance were poor searching technique (failure to use stemming and synonyms), stopping the query iteration too soon, and the inability to search on interdocument relationships. The authors argued that vocabulary problems make high recall impossible on full-text databases.

1.2 Related Work

Research to improve access to online information is proceeding in many directions. The hope is that by helping users with the mechanics of their search, and by providing access to an online thesaurus, better user interfaces can lead to improved search results with existing databases. Similarly, allowing the user to query the database in his natural language, rather than requiring him to form a Boolean query, may lead to simpler searching.

Accessing relevant information may also become easier by improving the quality of the information that is stored in the database. The three main approaches are (1) representing documents and search term as an associative network; (2) using natural language processing techniques to select index terms; and (3) building a knowledge base from the document contents. Researchers in artificial intelligence are investigating systems which, based on the contents of a knowledge base, produce direct answers to user queries, rather than documents or document passages.

Search performance may be improved by using statistical methods to reformulate the query. The user's initial query is used to rank-order the documents in the database. The top-ranked documents are presented to the user who indicates which are relevant. Index terms from the relevant documents are used to reformulate the query [22]. Finally, queries may be reformulated by a knowledge-based online search assistant acting as the front-end to existing retrieval systems. Research in this area is summarized in Section 1.2.1. The knowledge base for our system is built on existing searching practice. Current knowledge on good search technique is presented in Section 1.2.2.

1.2.1 *Expert Systems.* The exploration of possible applications of expert system techniques to information retrieval systems has generated interest. Early projects are surveyed by Sparck Jones [28] and Brooks [9], whereas Belkin et al. [2] discusses design issues for distributed expert-based information systems. The most common goal is to develop an expert system to help with the retrieval process by assuming some of the tasks of the search intermediary. An exception is Driscoll et al.'s [14] expert system whose task is to index documents. This section will give an overview of expert system projects designed for bibliographic retrieval and those which work with full-text databases.

Pollitt [19] has built one of the earliest expert systems for bibliographic retrieval. It is designed to search the MEDLINE medical database for cancer literature. The expert system can search cancer literature only, since the knowledge base contains information on cancer, rather than on search strategies in general. The performance of this prototype system has not been formally analysed. The current version of CANSEARCH [20] is an expert system to guide users in the use of menus to form their own queries.

IR-NLI II [8] incorporates user modeling into a domain-independent bibliographic retrieval expert system. Domain knowledge is supplied by an online thesaurus. A user model is built based on the user's amount of domain knowledge and search experience. This model is used to tailor the dialogue between the system and the user. Initially, the user lists some terms which describe his interests. The expert system, through a lengthy dialogue, clarifies its model of the query, proposes terms to expand the query, and comments on the user's search strategy. No automatic query reformulation is done.

IOTA [11] is a bibliographic expert system which incorporates a natural language interface. Whereas the expert system does passage retrieval from an online book, we include the system with the bibliographic systems because retrieval is done on keywords which index each passage. Much of the research effort has gone into processing the user's queries, but some simple query reformulation is also done. Specifically, queries are broadened by replacing a term by its parent from an online thesaurus and narrowed by removing OR terms. Their results show an increase in precision and recall using the expert system. These results are tentative, since the textbase is very small (3,000 words), the thesaurus is small (118 classes), and only 12 queries were run.

PLEXUS [33] is an expert system to help novice users find information about gardening. The initial query formation consists of a dialogue with the user. Natural language queries are accepted, and information is extracted to fill in frames. If a frame is too incomplete, the user is asked for more information. Once the frames contain enough information, a query is sent to the online database. The system has a knowledge base of search strategies and term classifications similar to a thesaurus. Most of the domain knowledge is in the classification, but some appears in the rule base, itself. If queries are too broad (defined as more than 10 references), no narrowing is attempted. The references are displayed 5 at a time to the user. If the query is too narrow (defined as nothing retrieved at all), three strategies are attempted: (1) if two or more terms appear in the same subcategory, OR them together rather than AND; (2) drop one of the terms; (3) replace a term by its parent.

Shoval [24] developed an expert system to assist users in selecting the right vocabulary terms for a database search. The knowledge base of words, concepts, and phrases and their semantic relationships is stored in a semantic network. Decision rules are used to locate appropriate vocabulary terms in the semantic network and suggest them to the user for possible query expansion. These rules were based on descriptions and observations of the search practices of information specialists. The user's initial search term's node is located in the semantic network. Candidate search terms are identified by expanding along directed links from the original node to nodes containing related terms. Terms which are linked to at least two active nodes are presented to the user. The user then decides whether or not the candidate terms are relevant and should be used to replace the terms in the nodes which point to it. This process may be continued until no new terms are generated.

I^{3}R [12] incorporates user modeling and relevance feedback. The query formation process is a dialogue between the user and the system, during which the user supplies a short natural language query or an initial relevant document. The domain knowledge expert infers related concepts from the query and presents them to the user for confirmation. If the thesaurus-like knowledge base does not contain related information and the initial query contained too many high-frequency terms, the user may be asked to provide additional keywords. A ranked list of documents is presented to the user. The user then indicates which terms in each document are interesting. These new terms may be used to modify the query. This specification of exactly which parts of the documents are relevant is an improvement on traditional relevance feedback. A blackboard architecture is employed to control the search process, which consists of the user dialogue, probabilistic search, cluster-based search, and user feedback.

Fewer projects have attempted to provide intelligent assistance for full-text searching. The earliest such system is RUBRIC [17, 31] which has the user describe his query in terms of rules. These rules describe the domain knowledge for the system as a hierarchy of topics and subtopics. Rules may have weights representing the certainty and/or importance of the defined relationships. The lowest level subtopics define patterns in the text which indicate the presence of that subtopic. Whereas the query language is very powerful, it places a heavy burden on the user.

At OCLC, the emphasis to date has been on providing an intelligent online help function, but a few basic reformulation strategies were provided in a demonstration full-text system [30]. Queries are broadened by asking the user to OR together ANDed concepts, or to drop a concept altogether. Narrowing is suggested when a single broad search term retrieves more than 30 passages. If this happens, the system first searches for multiword phrases containing the term in the back-of-the-book index and the table of contents. These phrases, if found, are presented to the user as alternate queries. If no such phrases are found, the system returns the passages which are clustered. If there is no clustering of hits, a random selection is shown.

1.2.2 *Search Strategies.* The automatic query reformulation incorporated in the systems described in the previous section are, in general, very primitive. However, search strategies employed by both novice and experienced searchers have been widely studied. These studies formed the basis of our expert system's searching knowledge base, which is described in detail in Section 2.6.

Searching studies. The most thorough catalogue of search tactics was compiled by Bates [1]. She outlines 29 search tactics in four areas: monitoring, file structure, search formulation, and term manipulation. The tactics for search formulation and term manipulation describe the available techniques to broaden and narrow queries. The search formulation tactics include the selection of appropriate initial search terms and the manipulation of query structure; the term manipulation tactics describe the use of context, thesaural terms, and stemming to modify queries. The tactics she lists provide the basic operations for our expert system; however, she includes no guideline as to when each tactic is appropriate. Bates concludes by saying that knowing when to stop a search is a difficult problem.

Smith et al [27] identify a set of search tactics, including 19 which were domain-dependent. They analyzed discourses between an expert intermediary and 17 real information seekers interested in the environmental literature of Chemical Abstracts. They noted when each of these tactics was applied, and whether the intermediary used the tactic spontaneously or in response to some cue in the retrieved document. The results of this study are being used as the basis for EP-X, an online search intermediary. EP-X represents the meanings of concepts and topics in the domain of interest in the form of hierarchically defined semantic primitives and frames. This knowledge is used to identify and resolve ambiguities in user queries (currently expressed as lists of keyword phrases). In retrieving documents, the hierarchy of concepts can be used to broaden the query to include specific cases of that concept.

Williams [34] has developed a model of all possible search situations and all possible responses, to be used as an expert system's knowledge base. Based on the desired, versus the achieved, values of three variables (numbers of documents, precision, and recall), he identifies 64 search situations which result in 27 unique states. He defines four variables (generality, exhaustivity, simplicity, ambiguity) which can be manipulated to respond to each state in an attempt to achieve the desired search results. Although he describes some techniques to manipulate the four variables, he does not indicate how the techniques should be combined or when they should be applied. In addition, several of the states have conflicting demands which are hard to resolve. It is an interesting categorization of searching situations, but it is not yet developed enough to become the basis of an automatic search assistant.

Effects of query expansion. Smeaton and van Rijsbergen [25] have studied the effects of query expansion on retrieval performance. They find that automatically adding terms based on their statistical relationships to the user's search terms degrades retrieval performance. They argue the need for better criteria for selecting terms to add. Harman [16] also shows performance degradation when adding terms from a statistically constructed thesaurus. However, when only those thesaural terms which occur in documents already flagged as relevant by the user are added, retrieval performance improves over that achieved by the original query. Allowing the user to add variants of the original search terms to the queries proves to be better than selecting from thesaural terms. However, statistically selected terms from the relevant documents proves to be the best candidate for query expansion. Finally, the best performance is achieved when user filtering of the three types of candidate terms (thesaural, term variants, and statistically selected from relevant documents) is simulated.

Crouch [13] has investigated the use of terms from an automatically constructed thesaurus for query reformulation. She concludes that augmenting a query with thesaurus terms, rather then replacing the user's original search terms, improves performance. She also advises that, for document ranking, terms included from a thesaurus should receive lower weights.

2. SYSTEM ARCHITECTURE

2.1 Overview

The prototype system consists of five major components (see Figure 1):

(1) MICROARRAS [26], which serves as the full-text search and retrieval engine,

(2) a full-text database of over 188,000 words,

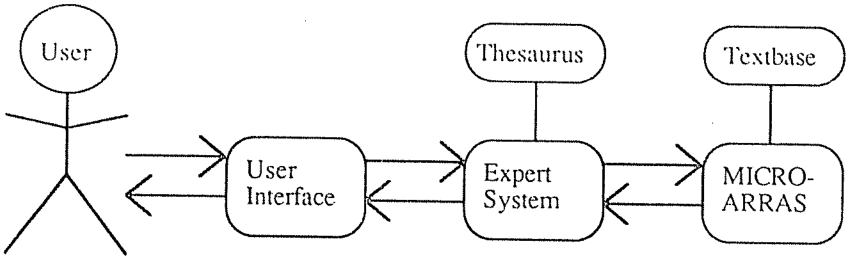

Fig. 1. System architecture.

(3) a hierarchical thesaurus of approximately 7,424 words specific to the textbase's domain,

(4) an expert system of 85 OPS83 rules and over 5,000 lines of C code, which interprets the user's queries, controls the search process, analyzes the retrieved text, and ranks the search results, and

(5) a user interface, which accepts the user's queries, presents requests for information from the expert system, and displays the search results.

The system is implemented on a Sun 3 workstation. MICROARRAS and the thesaurus construction and access routines are written in the C language. The expert system consists of a knowledge base of production rules, written in OPS83, and a set of C language functions to carry out the actions prescribed by the rule-base. The textual database for the current demonstration project consists of an unpublished manuscript on computer architecture written by Gerrit A. Blaauw and Frederick P. Brooks, Jr. [4]. The search process consists of a dialogue between the user and the expert system. The user enters the initial Boolean query and the number of passages he would like to retrieve. The expert system parses the query and translates it into a request for information from MICROARRAS. MICROARRAS retrieves text passages from the full-text database and informs the expert system of the number of passages that satisfy the request. The expert system compares the number retrieved with the target number to decide whether or not to reformulate the query, and, if so, how. Once the target number has been reached, or the expert system has run out of reformulations to try, the retrieved passages are presented to the user in rank-order.

A major advantage of this architecture is the separation of strategic knowledge, contained in the knowledge base for the expert system, from domain knowledge, contained in the thesaurus. Now that the search strategy rules have been developed and tested with the existing textbase, the expert system can be tested with other content domains by simply providing a suitable thesaurus for the new textbase.

2.2 MICROARRAS

2.2.1 *Capabilities.* MICROARRAS is a full-text retrieval and analysis system. The system provides immediate access to any passage in the textbase, regardless of the length of that document. Users can browse through a document's vocabulary as well as its text. MICROARRAS provides Boolean search on any word or set of words in the text and can compute and report various frequency of occurrence statistics in the form of distribution vectors over a text or set of texts. Contexts for searches can be indicated in terms of words, sentences, paragraphs, etc., for the entire search expression or for different parts of it. One particularly important feature for this project is a generalized categorization option by which one may define sets of words or text locations as well as recursive categories whose members are, themselves, categories. Any command that accepts a word as a parameter will accept a category name instead. Thus, categories can be used in search expressions, making MICROARRAS particularly well-suited to work with a hierarchical thesaurus.

To be inserted into MICROARRAS' textbase, documents must first be inverted (i.e., a dictionary is created with an entry for each word in the text. Each entry contains the word and the numerical position in the text of each occurrence of that word). However, they require no semantic preprocessing. Once stored in the textbase, they can be examined individually or in groups. They can also be moved from one textbase to another. Thus, documents can be processed on a workstation or microcomputer, uploaded into a textbase on a mainframe or textbase server, searched and analyzed there, or downloaded for local use once again.

2.2.2 *FLANGE.* FLANGE is a two-way command language that was developed as part of the MICROARRAS system. It serves two major functions: it provides communication between the user interface and the analytic engine that performs all search and analysis operation, and it provides a formal specification for the system. It is written in a BNF-like notation. Consequently, programs can easily construct command expressions which, in turn, can easily be parsed. Additionally, the components of a FLANGE "sentence" are strongly typed to further simplify processing and to ensure reliable transmission across a communication interface.

One particularly useful feature of FLANGE is its two-way communication capabilities. The following example outlines a typical interaction between MICROARRAS' user interface program and its analytic engine. Suppose the user wishes MICROARRAS to display concordance information for a particular word in a text in the textbase. The user's request for a concordance is first translated by the interface program into a FLANGE expression. That expression is then sent to the MICROARRAS engine, either running on the same machine or on a remote computer. The engine parses the message and performs the operation requested. It then encodes the results in the conventions of the return portion of FLANGE and sends that message to the user interface. The user interface parses the messages, interprets the result, and either displays the requested information to the user or engages the engine in a further FLANGE dialogue.

It is FLANGE's capability of providing a formal high-level text analysis language and its capability of delivering its results in a structured and typed form—rather than as a stream of data—that makes it feasible for an expert system to work iteratively with the textbase.

2.3 Textbase

The textbase contains the Fall (1986) draft of *Computer Architecture, Volume 1 - Design Decisions* by Blaauw and Brooks. The manuscript consists of 188,278 words comprising 8 chapters, titled: "Introduction", "Machine Language", "Addresses", "Data", "Operations", "Instruction Sequence", "Supervision", and "Input/Output".

Texts to be used as MICROARRAS textbases require format marks of interest to users to be inserted in the text. TeX format marks were already present and were used as the basis for the MICROARRAS segments. These included format marks to be used in the display of the retrieved text (line, italics, label), as well as those which provide context information (chapter, section, subsection, subsubsection, paragraph, sentence, item). A series of programs are then run on the formatted text to produce an inverted file. Finally, this inverted file is converted to fixed length records for fast access.

2.4 Thesaurus

All domain-specific knowledge is contained in a hierarchical thesaurus. The expert system uses this information to reformulate queries. The thesaurus was built by the author from the Brooks and Blaauw text, and it strongly reflects the word usage of that textbase. In general, it should not be necessary to provide a unique thesaurus for each textbase. An existing thesaurus for the domain could be used, as long as there is a good match between thesaurus classes and textbase word usage.

2.4.1 *Logical Structure.* This section describes the structure of the thesaurus. There are several thesaurus constructs that require definition. Word types which share a common stem are grouped into *stemgroups.* The members of a given stemgroup are called *stemwords.* Each word type in the Blaauw and Brooks text appears in exactly one stemgroup. Thesaurus classes contain stemgroups which are synonyms for each other. Stemgroups may appear in zero, one, or more than one thesaurus class. Because the thesaurus classes are linked together with parent-child links, they are also referred to as *nodes*. The arrangement of the words into stemgroups, stemgroups to thesaurus classes, and the classes into a hierarchy is discussed. Throughout this discussion, word types will be written in lowercase, stemgroup names with a leading uppercase letter, and thesaurus class names in uppercase.

At the lowest level, words with the same root are grouped into stemgroups. A stemgroup contains all the words that lexically share the same root. Most are easily identified by sorting the dictionary of word types in the database. Common forms of word types not used in the textbase—for example if there was no plural of a noun—are added to the stemgroup. Consider the grouping of words with the root, *structure.*

Stemgroup Name:	Structure
Stemwords:	structure, structuring, structured, structures

In addition to words that were lexically similar, words that are semantically forms of the same stem were included. Thus, *run* is the same stemgroup as *ran.* Finally, each stemgroup also contains words formed from the stem by the use prefixes. Thus, *undecided* is in the same stemgroup as *decided.*

Next, stemgroups pertaining to technical concepts are identified. Synonyms among these stemgroups are combined to form thesaurus classes. Nontechnical terms are not included in the thesaurus. Extremely low-frequency stemgroups, those occurring only once in the textbase, are also excluded.

High-frequency stemgroups represent broad concepts discussed throughout the text. They are often excluded from thesauri since they are too general. However, they are included in this thesaurus because they often occur in meaningful word phrases. If the user enters a high-frequency word, like *data*, the expert system could suggest the word phrases containing that word, for example *data structure* and *data type*, as possible replacements to narrow the query. If the high-frequency words are introduced during query reformulation they are filtered out. For a given high-frequency word, the phrases containing that word were identified by looking at all the sentences in which it appeared. The meaningful word phrases which occur more than once are included.

Finally, an ordering is imposed on the thesaurus classes. Conceptually, a thesaurus class can be viewed as a node in a directed acyclic graph (see Figure 2). Each node contains a name, a list of synonym stemgroups, the names of zero or more parent nodes, and the names of zero or more child nodes. Parent nodes—nodes higher in the thesaurus structure—represent more general concepts than the current node. Child nodes—nodes lower in the thesaurus structure—represent more specific terms. Nodes containing multiword phrases have as parents the nodes containing each of the component stemgroups. For example, consider the thesaurus entry for Data_Structure:

Node Name:	DATA_STRUCTURE
Node Stemgroups:	Data_Structure
Parent Node(s):	DATA, STRUCTURE, NAME_SPACE
Child Node(s):	ARRAY, QUEUE, STACK, LIST

2.4.2 *Thesaurus Words.* The thesaurus was manually constructed from the 8,313 different word types in the textbase. Removing numbers, punctuation, English function words, proper names, and words which appeared only once left 5,726 types. These were grouped into 1,993 stemgroups; common word forms missing from the stemgroups were added, bringing the total to 6,990 types. Using a concordance and frequency of occurrence, 936 technical stemgroups were selected to be arranged hierarchically in the thesaurus from the 1,993 available, resulting in 753 thesaurus classes. The construction of the thesaurus relied on the procedure outlined by Salton and McGill [22], the

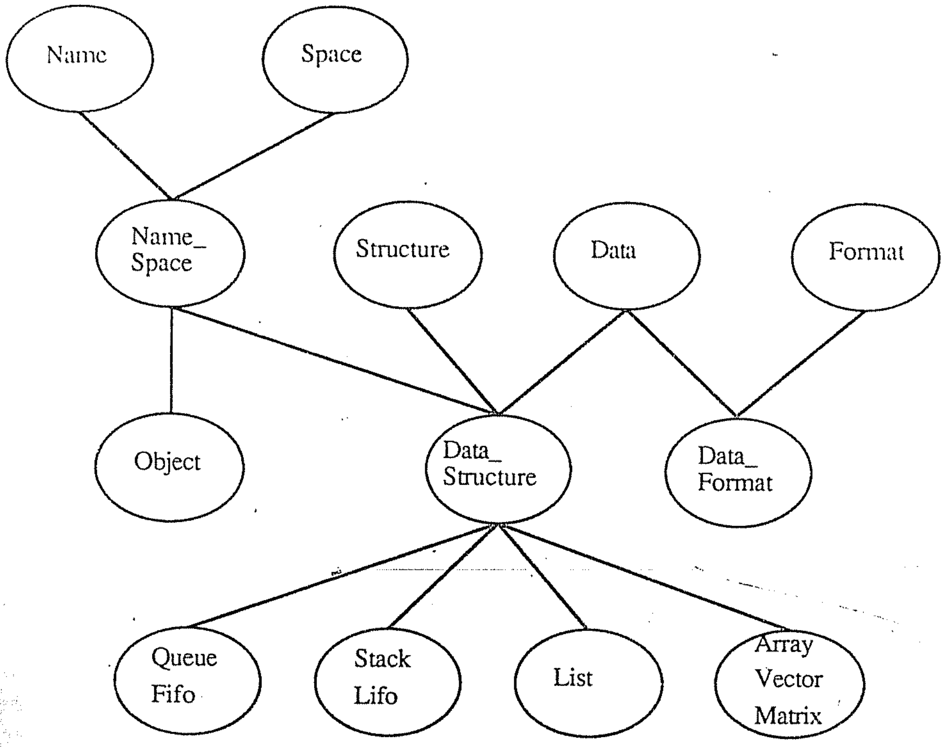

Fig. 2. A sample thesaurus.

author's knowledge of computer architecture, and the hierarchical arrangement of sections in the Blaauw and Brooks text.

2.5 Query Language

When the user starts the system, the following prompt appears:

Enter a query, or quit, terminated by ⟨return⟩:

The system expects a Boolean query. A Boolean query language was chosen because it is the most common type available on existing systems. We wanted the main difference between this prototype and conventional full-text retrieval systems to be the searching knowledge base so that any improvement in search performance could be attributed to the encoded search strategies, rather than the user interface. Possible improvements to the interface are discussed in Section 4.

2.5.1 *Operators.* The operators provided, in decreasing order of operator precedence, are: ANDNOT, AND, and OR. A logical equivalent to any Boolean expression can be constructed using these operators. Where there are two or more operators of equal precedence, they are evaluated left to right. Parentheses have the highest priority and can be used to override the default order of evaluation. The operators are distinguished from the search words by their position.

2.5.2 *Search Terms.* When a query is parsed, the expert system interprets each search term to represent a unique area of interest, or *concept*, specified by the user. The concepts, and the operators, are flagged as positive or negative based on whether they are specifying information the user does, or does not, wish to receive. For example, the query 'i/o ANDNOT (device OR interrupt)' contains three concepts: *I/O*, *device*, and *interrupt*. *I/O* is a concept on which the user wishes information, so it is considered a positive concept. *Device* and *interrupt* indicate concepts on which the user does not wish information, so they are considered negative concepts. The ANDNOT and OR operators are followed by negative concepts, so they too are flagged as negative.

2.5.3 *Context.* A default context of one sentence is used for the AND and ANDNOT operators. For example, 'virtual AND memory' will retrieve all passages in which *virtual* and *memory* appear within the same sentence, regardless of order. Similarly, 'page ANDNOT fault' will retrieve passages in which *page* appears, but not those in which it appears within the same sentence as *fault.*

When the user is searching with the expert system, the expert system controls the context. Initially, the default of one sentence is used, but the expert system may adjust the context during query reformulation. However, when the user is searching without the expert system, the AND and ANDNOT operators may be augmented with a user-specified context. The user may define the search context for AND or ANDNOT in terms of words, sentences, or paragraphs. The most general context definitions have the form:

left-expression operator [integer1 to integer2 units] right-expression

where integer1 must be smaller than or equal to integer2, and units is either words, or sentences, or paragraphs. The integers specify the range around the tokens satisfying the left-expression (tokenL) in which the tokens satisfying the right-expression (tokenR) must appear. 0 represents the unit containing the token from the left hand side. Thus, the default context of one sentence is equivalent to

0 to 0 sentences

indicating that tokenR must appear in the same sentence as tokenL. An abbreviation for this context, *sentence*, is provided for the user's convenience.

Negative integers indicate that tokenR must precede tokenL; positive integers indicate that it must follow. Thus,

-5 to $+3$ words

specifies that tokenR must appear in the region around tokenL that includes the five words preceding tokenL and the three words following it. If one is looking for paragraphs containing a specific phrase, for example *computer* *architecture*, one would use the query

computer AND [+1 to +1 words] architecture

which requires that *architecture* immediately follow *computer*. There is also a shortform for this relationship, called *nextword*.

Finally, since the retrieved passages are one paragraph long, search expression contexts should not be larger than one paragraph. Thus, the only valid context involving the unit paragraph is

0 to 0 paragraphs

and this context is abbreviated by *paragraph*.

Users may accidentally define contexts which, when evaluated, cross paragraph boundaries. For example, if the user has specified a context of plus or minus three sentences, tokenR may appear in a different paragraph than tokenL. In this case, MICROARRAS retrieves the paragraph containing tokenL. The user might be confused as to why a particular paragraph containing only one of the tokens of interest was retrieved. In contrast, when the expert system controls the context, it builds more complicated contexts which specify that the tokens must appear in the same paragraph. For the above example, the expert system would specify a context of plus or minus three sentences within the same paragraph.

2.6 Knowledge Base

2.6.1 *Overview.* Professional search intermediaries use four main types of knowledge—their knowledge of how particular databases are constructed, knowledge about the domain being searched, knowledge of the user, and knowledge of general search strategies—to form and improve queries. The expert system handles all interactions with MICROARRAS, the text retrieval software used; the user will need no specific knowledge of this system. Domain knowledge is all incorporated in the hierarchically structured thesaurus. This system has no knowledge of the user's true information needs, other than the target number they specify to indicate how many passages they wish to retrieve. The rest of this section discusses the knowledge base of search strategies that forms the core of the expert system.

The expert system performs three main functions:

(1) it controls the operation of the system as a whole;
(2) it reformulates the Boolean query based on previous search results;
(3) it ranks the retrieved passages in decreasing order of estimated relevance for presentation to the user.

To perform these functions the expert system contains a knowledge base of the search process, search strategies, and passage ranking procedures.

2.6.2 *Query Reformulation Rules*

Overview. Queries are reformulated based on the target number, the number of passages retrieved, and the history of broadening and narrowing techniques already applied. The expert system has a collection of reformulation tactics at its disposal. Bates [1] and others have identified successful search tactics. However, no one has outlined an overall query reformulation strategy combining these tactics. The guiding principles for this expert system's query reformulation knowledge base were (1) each search term in the initial query represents one concept on which the user does, or explicitly does not, want information; (2) the user's initial search terms are the best indication of the user's areas of interest; (3) some terms from the thesaurus may be helpful, but others will not; (4) the expert system should never discard concepts in which the user has indicated an interest.

Query reformulation techniques. The expert system reformulates queries using three different techniques: (1) expanding concepts; (2) adjusting context; and (3) changing the query structure.

Expanding concepts. To broaden a query, search terms are added to the positive concepts, whereas narrowing a query adds search terms to negative concepts. Concepts may be expanded by stemming, adding synonyms, and adding related search terms for the thesaurus. Crouch [13] found that augmenting a query with thesaurus terms, rather than replacing the original search terms, leads to improved results. With this in mind, concepts are expanded by adding thesaural terms (ORing them with the terms already in the concept) rather than by replacing the terms already present.

The belief that some stemgroups from the thesaurus will be useful, while others will not, is the basis for providing user filtering of the candidate thesaurus terms. The domain-dependent search strategies identified by Smith, Shute, and Galdes [27] involved the use of domain knowledge to choose the appropriate terms from a thesaurus. In addition, Harman [16] showed that search results improved when thesaural terms were filtered by the user. Based on these two studies, we decided to allow the users to select which stemgroups to add from a set of thesaural candidates.

Finally, candidate search terms selected from the thesaurus are filtered to remove those which already occur in the query and extremely high frequency terms. The remaining terms are added one at a time, in reverse order of frequency, and the new number of retrieved passages is compared to the target number.

Adjusting context. The expert system manipulates four different contexts; it adjusts the distance between words in positive and negative multiword phrases as well as the distance between positive and negative search concepts. The expert system broadens queries by increasing the positive contexts and decreasing the negative ones. Conversely, narrowing is done by decreasing the positive contexts and increasing the negative ones.

Changing query structure. The final variable the expert system can manipulate is the query structure. The query can be broadened in two different ways: first, the positive AND operators can be switched to OR operators (and the negative OR operators switched to ANDs); second, the negative parts of the query can be dropped altogether. All of the AND operators are replaced at the same time. A better strategy would be to replace them one at a time, in inverse order of the frequency of occurrence of the concepts. Similarly, the query can be narrowed by replacing OR operators with ANDs. The expert system does not have enough information about the user's information needs to decide which positive parts of the query to drop, so this technique is not employed to narrow queries.

Flow of control. Figure 3 diagrams the flow of control among the reformulation techniques. The left side of the Figure 3 diagrams the broadening techniques, the right side the narrowing techniques. This figure is somewhat simplified since it does not show the use of context to converge to the target number once queries have been found which bracket the target number from above and below.

The expert system records the type of initial query reformulation as the global objective. If the reformulations in the original direction overshoot the target number without achieving success, reformulations in the opposite, or local, direction are tried, beginning at the top node on that side of the diagram. Reformulation never continues in the local direction farther than it reached in the global direction. Queries have been formed which bracket the target number from below and above, otherwise the system would not have tried both narrowing and broadening techniques. Rather than using techniques which are considered less likely to produce good results, the expert system adjusts the context.

Expanding a concept. The first reformulation technique tried, whether broadening or narrowing, is adding the rest of the initial search term's stemgroup to each appropriate concept in term. Next, synonyms are added, followed by related terms from the thesaurus. The order in which the terms are added from the thesaurus is parents, then siblings, then children. Replacing a term with its parent to broaden a query is a common practice, both by searchers [1, 23], and in systems which automatically reformulate queries [11, 33]. The rationale is that since parent terms represent broader concepts, adding the parent term should broaden the scope of the query. Thus, parent terms are added first. Siblings are added second since they represent related concepts, and children terms are added third since they represent narrower concepts and seem less likely to broaden the concept. While the expert system uses this ordering, the reasoning is based on experience with searching bibliographic databases using keywords. In full-text databases, we believe that the reverse order may make more sense. Broadening a concept containing *apple* with children terms, yielding 'apple OR mcintosh OR granny_smith', seems more likely to retrieve relevant passages than broadening with the parent terms, yielding 'apple OR fruit'. Observing the system, adding parent and sibling terms currently takes a lot of time since there are so many candidates, but rarely do they increase the number of passages retrieved. Adding child terms usually retrieves more passages. Attempting this technique sooner would increase the speed of the expert system because the reformulation might stop several steps earlier.

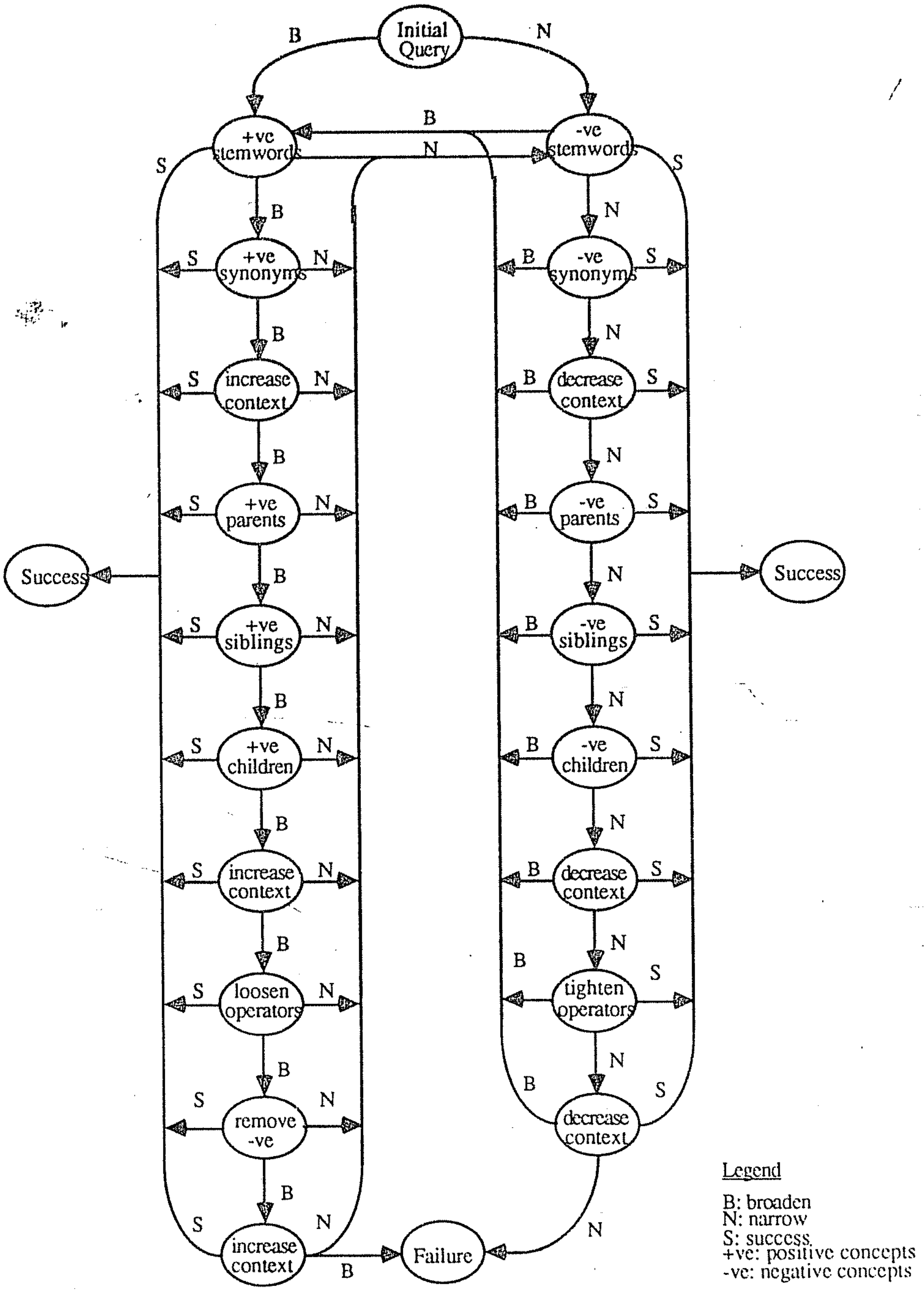

Fig. 3. Query reformulation techniques.

Adjusting context. Whereas there were several sources of information to draw from to determine the order in which to apply the search term expansion techniques, there was less information available on the use of context to reformulate queries. Contextual searching on document contents is not available on standard bibliographic systems, and full-text systems usually supply operators for adjacency, same sentence, same paragraph and same document, only. There is no established practice on when to adjust context rather than expand search terms.

We chose to adjust context in four specific places in the expert system: after adding all stemgroups from the same thesaurus class, but before adding any stemgroups from related thesaurus classes; after adding all related stemgroups from the thesaurus, but before changing the Boolean operators; after changing the Boolean operators but before declaring failure; and after the local reformulations have progressed as far through the search techniques as was used in the original reformulation direction. These places were chosen because it seems desirable to try adjusting context, which does not alter the concepts being searched, before moving on to a new group of reformulation techniques which may move the query farther from the user's original intentions.

Changing query structure. Manipulating query structure causes major changes to the user's original query. These techniques are only tried after all close relatives from the thesaurus have been added and context has been broadened twice. It is not likely that the new query will find passages that the user will find highly relevant, but the goal is to find somewhat relevant passages that users can read in order to reformulate their own queries and try again.

Stopping. Bates [1] stated that knowing when to stop a search is a difficult problem. We partially side-step this problem by having the user explicitly state the number of passages he wishes to retrieve. Since the target number he supplies is likely to be a rough guess, a range of 20 percent is considered successful. A larger range may be desirable, but since the user is able to stop the reformulation process himself, the size of the range is not important. Left on its own, the expert system stops the reformulation process when it achieves success, or it has run out of techniques to try.

2.6.3 *Passage Ranking Rules.* The dialogue between the expert system and MICROARRAS normally produces a set of passages to be displayed to the user. The last task performed by the expert system is to rank order those passages in terms of their probable interest to the user. To do this, it performs an elementary content analysis on each passage and computes a weight representing probable interest.

Ranking algorithms for document retrieval systems have been extensively studied. There has been less work done on ranking for passage retrieval systems. The FAIR system [10] performs a simple ranking based on the distance between word pairs, the number of search terms represented and the number of occurrences of the terms. The ranking algorithm used by the expert system considers the following factors: the number of different concepts represented in the passage; the number of different word types for each concept; the relationship of the concept's word types to the user's original search terms; the number of occurrences for each word type from the search expression appearing in the passage; and the contextual distance between search terms. The passages are then ranked according to their respective weights and presented to the user in order of decreasing rank.

Calculating passage weights. The weight W_{pq} of passage p for query q, $0 <= W_{pq} <= 1$, is a function of the weight C_{ip} of each query concept i in p, the relationship between the concepts (determined by the parse tree), and the contextual closeness between the concepts. The concept weights are combined by applying the rules for fuzzy logic [35] to the Boolean structure of the query. Additionally, a closeness factor is associated with each of the AND and ANDNOT operators. The closeness factor for the AND operator is set to one of three values (1.0 for same sentence, 0.9 for adjacent sentences, 0.8 for same paragraph). The closer two positive concepts appear in the passage, the higher weight that passage receives. Complementary closeness values are used for the ANDNOT operator (0.8 for same sentence, 0.9 for adjacent sentences, 1.0 for same paragraph).

$$W_{p(Ci \text{ AND } Cj)} = \min(C_{ip}, C_{jp}) * \text{PositiveCloseness} \tag{1}$$

$$W_{p(Ci \text{ OR } Cj)} = \max(C_{ip}, C_{jp}) \tag{2}$$

$$W_{p(\text{NOT } Cj)} = (1 - C_{jp}) \tag{3}$$

From (1) and (3)

$$W_{p(Ci \text{ ANDNOT } Cj)} = \min(C_{ip}, 1 - C_{jp}) * \text{NegativeCloseness} \tag{4}$$

The concept weights and closeness factors fall in the range [0, 1], therefore the passage weights also fall in the range [0, 1].

Calculating concept weights. The weight of concept i in passage p, Cip, is a function of the weight of each concept term T in query q, denoted T_{jq} for search term j, and the weight of each concept term in the passage, denoted T_{jp} for search term j, and the number of search terms for the concept. The weight of a search term in the passage is multiplied by the weight of that search term in the query. Thus, the highest weight search terms are those which are important in the query as well as the passage. The weights for all the concept's search terms are summed together and normalized by the number of search terms for the concept, N.

$$C_{ip} = 1/N \sum_{j=1}^{N} T_{jq} * T_{jp} \qquad \text{where term } j \text{ is in concept } i \tag{5}$$

The term weights fall in the range [0, 1], therefore, the concept weights also fall in the range [0, 1].

Calculating term weights. Two different term weights, T, are calculated: the weight of the search term i in query q, T_{iq}, and the weight of the search term i in passage p, T_{ip}.

Query term weights. The weight of the search term i in query q, T_{iq}, reflects the relationship of the search term to the user's original term. The relationships, from closest to most remote, are same word, stemgroup, synonym, parent, sibling, child. These distances reflect the order in which search terms are added to the concepts, which in turn reflects confidence in the closeness of the relation of the search term to the original term.

$$T_{iq} = 1.0\ (\text{word}), 0.9\ (\text{stemgroup}), 0.8\ (\text{synonym}), 0.6\ (\text{parent}), 0.5\ (\text{sibling}), 0.4\ (\text{child}) \qquad (6)$$

The query term weights fall in the range [0, 1] as required, with the original word receiving a weight of 1.0. Terms added by the expert system receive weights which decrease by 0.1 for every step away from the original term, except for the step from synonym to parent terms. This step decreases the term weight by 0.2, reflecting the large decrease in confidence which occurs when terms are added from outside the thesaurus class.

Passage term weights. The weight of the search term i in passage p, T_{ip}, reflects the frequency of the search term in the passage, f_{ip}, and the frequency of the search term in the textbase, f_{it}. Ro [21] evaluated several full-text ranking algorithms and concluded that those based on relative document frequency provided the best performance. Thus, we chose relative frequency for the term passage weights.

$$T_{ip} = f_{ip}/f_{it} \qquad (7)$$

The term passage weights fall in the range [0, 1], as required.

2.7 Sample Scenario

Before the system components are described individually, a sample scenario will be presented to illustrate how they work together to provide an intelligent online search assistant. Since our current textbase concerns the domain of computer architecture, the following example describes the interactions of the system and a user searching for information on the alignment of word boundaries in memory.

The user might enter a query 'boundary AND word ANDNOT page', which indicates that he wishes to retrieve passages containing information on word boundaries but not page boundaries. Assume a target number of 15. Applied to this textbase, the original query would retrieve only one passage, so the expert system would attempt to broaden the query. The first step would be to replace the word types *boundary* and *word* with their stemgroups. The resulting query would be 'Boundary AND Word ANDNOT page', where the capitalized search terms indicate the whole stemgroup is included. Notice that *page* has not been expanded to its stemgroup, as it is a negative, or excluded, concept. Four passages would now be retrieved.

The next step would be to broaden the query by including synonym stemgroups for each of the positive search terms, in turn. From the thesaurus it is found that Boundary has one synonym, Limit, however there is no synonym for Word. The query now becomes '(Boundary OR Limit) AND Word ANDNOT page', which retrieves seven passages. Relaxing the context around the AND operator to adjacent sentences while decreasing the context around the ANDNOT operator to within 5 words increases the number of passages retrieved to nine. To further broaden the query, the parent stemgroups for the positive concepts are added. Block and Segment are added to the concept Boundary. The Word concept remains unchanged, since Word has no parent in the thesaurus. The query becomes '(Boundary OR Limit OR Block OR Segment) AND Word ANDNOT page', which retrieves twelve passages. Twelve is within 20 percent of the fifteen passages requested, so the reformulation stops. If the user requests to see the retrieved passages, the expert system would rank the retrieved passages and present them to the user in decreasing rank-order.

3. EVALUATION

Evaluating an interactive system is difficult. Tague and Schultz [29] have defined a framework for evaluating information retrieval system interfaces. They identified three ways to measure the information retrieval system: informativeness, time, and user friendliness. Informativeness is measured by retrieval output (search effectiveness) and retrieval order (ranking). The search efficiency of the system is related to Tague's time factor. Finally, the user friendliness of the system can be evaluated by a post-search questionnaire.

Our primary goal is to demonstrate that using an expert system to reformulate queries can improve search performance for novice searchers. Ideally, both their effectiveness and efficiency would be improved. The second, less important, goal is to show that the expert system can rank the retrieved passages in decreasing order of relevance.

To evaluate the expert system, subjects attempted to find relevant passages in response to high-level questions. They queried MICROARRAS with three interfaces with different capabilities: an interface whose only function was to accept contextual Boolean queries and display search results; a similar interface which also allowed the user to explore the online thesaurus; and a third which incorporated the searching expert system. Each subject's search performance with the three interfaces was monitored and compared.

3.1 Hypotheses

Hypothesis 1: The expert system improves the search effectiveness for a novice searcher.

Hypothesis 2: The expert system improves the search efficiency for a novice searcher.

Hypothesis 3: The expert system can rank the passages retrieved by the search in decreasing order of relevance.

The effectiveness of the retrieval output is evaluated by looking at recall (the number of relevant items found / the total number of relevant items in the database) and precision (the number of relevant items retrieved / the number of items retrieved). Two estimates of the number of relevant items retrieved are examined: the number of passages the users mark as relevant and the number of passages retrieved from the set of passages deemed relevant by the author.

The efficiency of the system is measured by the number of Boolean queries the subjects entered for each of several high-level questions, and by the amount of time they spent searching for relevant passages for each question.

The ranking algorithm was evaluated by comparing the order of appearance of relevant passages after they have been ranked with a random order of appearance.

3.2 Method

3.2.1 *Subjects.* Twelve computer-science graduate students participated as subjects in the study. All subjects were knowledgeable in the use of computers, but unfamiliar with online searching. Thus, they were representative of the anticipated users of future information retrieval systems.

3.2.2 *Apparatus*

Information retrieval systems. The *user-alone* configuration consisted of a Sun 3 running MICROARRAS and a rudimentary expert system. This expert system performed only the system control function, and did no query reformulation or ranking of retrieved passages. The user was prompted for a contextual Boolean query, this query was sent to MICROARRAS, and the number of passages retrieved was reported back to the user. The user could display the passages retrieved, if there were fewer than 25, or try another query. Typing was minimized by using the Sun's windowing package to cut and paste the previous query, edit, and rerun it.

The *user-thesaurus* version consisted of a Sun 3 with one window running MICROARRAS, as in the user-alone system, and a second window running a thesaurus access function. In the thesaurus window the user had access to all the thesaurus information available to the expert system. He could find out the stemname for a specific word's stemgroup. For any stemname, he could ask for the stemnames of the corresponding synonym, parent, sibling, or child stemgroups. These stemnames could be used in the user's query to MICROARRAS. Typing was minimized by using the Sun's windowing package to cut the stemgroup from the thesaurus window and paste it into the appropriate concept of the query.

In the *user-expert system* version the user did not have access to the online thesaurus. Context and the addition of stemgroups were controlled by the expert system. Thus, the user entered a Boolean query and a target number of passages and the expert system reformulated the user's query to attempt to get close to the target number. The user was prompted to filter search terms found in the thesaurus, and to continue or abandon the current reformulation.

To keep the response time approximately the same as for the other two configurations it was necessary to run MICROARRAS remotely on the Sun 4 file server containing the textbase. The user worked with one window on a Sun 3 which ran the full version of the query reformulation expert system. The expert system communicated with MICROARRAS over the network. This setup was approximately twice as fast as when MICROARRAS was run on the user's Sun 3. This speed up was necessary, not because the expert system code itself was slow, but rather because the expert system tended to form very long queries involving many MICROARRAS categories, and MICROARRAS slows down linearly with the number of search terms in a query.

Questions. Three sets of five questions were devised. Each set contained one training question and four questions on which the subjects were monitored. The questions covered material ranging over the whole textbase. The number of relevant passages found by the author (see Definitions) follows each monitored question.

Query Set A

Practice:

What are some sources of error in floating point arithmetic?

Monitored:

(1) How is computer architecture distinguished from the other computer design domains? (16)

(2) What are some upward pressures on the level of a machine language? (16)

(3) Fixed length multiplication produces a double length result. How have different machines handled this? (14)

(4) How are interrupts handled? Do not consider techniques to disable them. (23)

Query Set B

Practice:

I/O devices have moving parts. What is the effect of this motion on the architecture of computers?

Monitored:

(1) What are some design principles that lead to clean architectures? Do not consider the economic advantages of a quality design. (16)

(2) What techniques have been used to reduce bit traffic? (10)

(3) How are control structures implemented? (13)

(4) What role does buffering play in I/O transfers? (22)

Query Set C

Practice:

Fragmentation of memory is one problem of using a segmentation scheme. How is paging used to fix this?

Monitored:

(1) Discuss the two fundamentally different ways to formally specify an architecture. (19)

(2) What are the effects of having two zeros, as in the sign magnitude representation of fixed point numbers? (7)

(3) What is done to save state upon a procedure call? (15)

(4) Besides I/O, where is concurrency practiced in the implementation? (16)

3.2.3 *Procedure.* Subjects were asked to try to find on the order of ten relevant passages from the textbase in response to the questions they would be given. They were informed that they might not always be able to find that many, and they were allowed to stop working on a query whenever they were satisfied that they had found as much as they could. The target number of ten was chosen because it was large enough to require a high recall search, yet small enough that the users would not become tired reading passages. For similar reasons, Vernimb [32] also used a target number of ten when developing an automatic query reformulation system for document retrieval.

Each subject worked with each of the three systems, in turn. This was done to compensate for the large individual differences found in searching ability [7]. To compensate for learning during the experiment, the order of presentation of the three systems was counterbalanced among subjects. The order of presentation of the question sets was the same for all subjects (Set A first, then B, then C). Thus, each question set was searched on each system four times. The subjects received a training session with each system before they began their monitored searches. When they had completed all three sessions, they were asked to fill out the questionnaire stating their preferences and opinions.

3.2.4 *Data Collection*

Raw data. Data was collected in a trace file while the subjects worked with the system. Each communication from the subject to the retrieval system, and vice versa, was stored with a time stamp. Thus, timing information was collected along with the history of queries entered by the subject and the search results. When the subject chose to display the retrieved passages, those passages and the subject's relevance judgement of them were also stored.

Several parameters were chosen from the trace file to represent each subject's sessions. Measurements were taken on time, number of queries, and number of relevant passages. Before the variables to be compared are described, we will provide a few definitions.

Definitions. A *unique query* was any error-free query entered by a subject. If a subject entered a query which contained a typographic or logical error, and he indicated that he noticed the error by aborting the search and reentering a corrected version, then the erroneous query was not considered a unique query. However, if the subject gave no indication that he was aware of the error, but instead moved on to a different query altogether, then the erroneous query was considered unique.

The *relevance weight* of a passage is the relevance number assigned to the passage by the subject. A *very relevant* (*user*) passage is one assigned a relevance weight of two. A *somewhat relevant* (*user*) passage has a relevance weight of one. A *relevant passage* (*user*) is one that is either very relevant or somewhat relevant, as judged by the user. An *irrelevant passage* (*user*) is a passage given a relevance number of zero.

It is necessary to have an estimate of the total number of relevant passages available for each question, in order to calculate recall. This estimate was calculated by forming the union, for each question, of the set of passages judged very relevant by any subject. Passages in this set judged irrelevant by the author were removed. The remaining passages form the *absolute retrieval set* are called the *relevant passages*. It was necessary to remove some passages marked very relevant by a subject because, perhaps due to a misinterpretation of the question or a misunderstanding of the passage, some subjects gave a relevance weight of two to irrelevant or marginally relevant passages. This tendency to overestimate the relevance of passages may also be because, in some cases, subjects were unable to find the truly relevant passages, and thought that they had retrieved the best passages available when in fact they had not.

A *successful retrieval set* is a retrieval set containing at least five relevant passages. Since the subjects were attempting to find ten relevant passages, a successful retrieval set contains at least half the number for which they were looking. The textbase contained approximately the same number of relevant passages for each question, allowing the target number and size of the successful retrieval set to be held constant.

The *final retrieval set* was chosen as the last successful retrieval set. If a subject never retrieved a successful retrieval set for a given question, the retrieval set with the highest number of relevant passages, as judged by the subject, was chosen. The *final query* is the query input by the user which resulted in the final retrieval set.

Variables. Total time per question is calculated from the entry of the subject's first query for the question until after the display, or decision not to display, of the final set of retrieved passages.

Number of queries per question is determined by counting the number of unique queries the subject entered for a given question.

Number of relevant passages (*user*) *found per question* is determined by counting the number of user indicated relevant passages in the final retrieval set for the question.

User precision is calculated for the final retrieval set using the standard formula of number of relevant passages (user) retrieved / number of passages retrieved.

Number of relevant passages found per question is determined by counting the number of passages in the final retrieval set for the question that are members of the absolute retrieval set.

Precision is calculated for the final retrieval set using the standard formula of number of relevant passages retrieved (absolute) / number of passages retrieved.

Recall is calculated for the final retrieval set using the standard formula of number of relevant passages retrieved (absolute) / total number of relevant passages available.

The *ranking balance point* (R) for each retrieval set (not just the final one) is calculated by

$$\frac{\sum_{i=1}^{n} i^{*}\text{relevance}_i}{\sum_{i=1}^{n}\text{relevance}_i}$$

where n = number of passages in the retrieval set
i = position of the passage in the retrieval set
relevance$_i$ = relevance weight of passage i

This calculates where the midpoint of the relevant passages lies, accounting for the relevance weight. The earlier in the retrieval set the relevant passages occur, the smaller their midpoint. For example, consider a retrieval set of five passages of which the first two are very relevant (weight = 2), the next two irrelevant (weight = 0), and the last passage somewhat relevant (weight = 1). The ranking balance point for this set would be

$$(1*2) + (2*2) + (3*0) + (4*0) + (5*1)/5 = 2.2.$$

The *random balance point* (R) for each retrieval set is calculated by (n + 1)/2 where n is the number of passages in the retrieval set. A random distribution of relevant passages in the set would have the midpoint (M) of the retrieval set as the balance point. Therefore, the random balance point for the set of five passages in the previous example would be 3.

The *best case balance point* (BC) for each retrieval set is calculated by applying the ranking balance point formula to the case where all very relevant passages preceded all somewhat relevant passages which in turn preceded all non-relevant passages in the set. In this case, the ranking balance point would be

$$(1*2) + (2*2) + (3*1) + (4*0) + (5*0)/5 = 1.8.$$

The *normalized ranking balance points* were calculated from the ranking balance points by moving the random balance point to 0 and adjusting the range so that the best case balance point fell on 1, and the worst case balance point at −1. The normalization performed was

Normalized ranking balance point (NR) = (M − R)/(M − BC).

For the example retrieval set, the normalized ranking balance point would be

$$(3 - 2.2)/(3 - 1.8) = 0.67.$$

Summaries calculated for each system. For each system the means calculated were

—number of queries per question
—time per question (seconds)

—number of relevant passages (user) per question
—user precision
—number of relevant passages (from absolute retrieval set)
—precision
—recall

For each ranking algorithm (the expert system's, and randomness) the normalized balance points were calculated.

3.3 Results

The means were compared to determine if their differences were statistically significant. Pairwise two-tailed t-tests were performed. A difference was considered significant if its probability of occurring due to chance was less than 5 percent at the 95 percent confidence level (a 10 percent chance at the 95 percent confidence level was considered marginally significant). Pairs of means with statistically significant differences are flagged with asterisks.

3.3.1 *Search Effectiveness.* All three systems retrieved comparable numbers of relevant passages. Whereas there seemed to be higher recall with the thesaurus, shown by a mean of 7.688 compared to a mean of 7.292 with the expert system, this difference was not significant ($p = 0.5333$).

—number of relevant passages (user) per question

—user alone	7.375
—user and thesaurus	7.688
—user and expert system	7.292

All three systems produced comparable precision, based on the subject's relevance judgements.

—user precision

—user alone	0.763
—user and thesaurus	0.786
—user and expert system	0.761

All three systems retrieved approximately the same number of passages from the absolute retrieval set.

—number of passages from absolute retrieval set

—user alone	5.521
—user and thesaurus	5.708
—user and expert system	5.729

Recall was comparable across all three systems. There was a slight improvement in recall for the user and expert system configuration, but the advantage over the user-alone configuration was not significant ($p < 0.6988$).

—recall		
—user alone	0.364	
—user and thesaurus	0.368	
—user and expert system	0.379	

The user and expert system configuration produced marginally significant improvements in precision when compared with the user-alone configuration.

—precision		
—user alone	0.530*	($p < 0.0817$)
—user and thesaurus	0.576	
—user and expert system	0.604*	

3.3.2 *Search Efficiency.* The expert system was not significantly slower than the other two systems. However, the user was marginally significantly slower when using a thesaurus. However, MICROARRAS was being executed by a Sun 4 with the user-expert system configuration resulting in approximately a doubling of its speed.

—mean time per question (seconds)		
—user alone	474.5*	($p < 0.101$)
—user and thesaurus	571.5*	
—user and expert system	539.8	

The expert system improved search efficiency, as measured by number of user queries over both the user alone and user plus thesaurus.

—number of queries per question		
—user alone	4.833*	($p < 0.0001$)
—user and thesaurus	5.458**	($p < 0.0001$)
—user and expert system	2.354*,**	

3.3.3 *Ranking.* The expert system ranked relevant documents more highly than would be predicted by randomness. The expert system's ranking was compared to a random distribution for 74 sets of retrieved passages.

—balance points		
—random	5.00 *	($p < 0.0165$)
—expert system	4.53 *	

—normalized balance points (on range of -1 to $+1$)		
—random	0.000*	($p < 0.0025$)
—expert system	0.195*	

3.4 Analysis

The first hypothesis that the expert system can improve the search effectiveness for a novice user was not supported by this study. However, the expert system produced marginally significant improvements in precision, and seemed to indicate improvements in recall. Providing the online thesaurus produced no improvement in search effectiveness.

The possible improvements in precision may result from the expert system applying better broadening techniques. The subjects, when searching alone, would often stop with a very broad query and examine a large set of retrieved passages (over fifteen) looking for relevant information. This type of strategy results in the lower precision observed when the subjects search on their own.

However, this browsing strategy also accounts for the ability of the subjects to produce recall comparable to the expert system when there were a large number of relevant passages in the textbase. For example, in two questions with large absolute retrieval sets the subjects were able to retrieve, on average, 10 and 10.25 relevant passages on their own compared with the expert system's retrieval of 8 and 7.75 passages respectively. By using a target number of 10 for these broader questions, the expert system was operating at a disadvantage. More relevant information was easily found, judging by the high recall of the subjects, but the expert system did not even attempt to further broaden the query. Clearly, 10 is not the ideal target number for all queries.

The second hypothesis that the expert system can improve the search efficiency of novice searchers was supported. Using the expert system significantly reduced the number of queries subjects needed to answer a given question. Subjects required fewer than half as many queries per question on average versus systems in which the user queried without it, a substantial improvement. The expert system reduced the amount of user effort required by decreasing the number of queries a user needs to design to express their information needs. If efficiency is measured in terms of total user time the expert system fares less well. The expert system was not significantly slower than either of the other two systems but it was necessary to run MICROARRAS on a faster machine to achieve this. However, this version of the expert system was designed with correctness rather than efficiency in mind, and there are several ways that it could be sped up. In particular when a stemgroup is added to a concept, the entire query is reevaluated against the textbase. A large speed improvement could be gained by unioning the passages retrieved by the new stemgroup with those retrieved by the rest of the concept (which has already been calculated). Then, the resulting set of passages could be merged with those retrieved by the rest of the query (which has also been already calculated).

Allowing the subjects to access the online thesaurus actually decreased the subject's efficiency. They took significantly more time than when they searched on their own, and required no fewer queries. This seems to indicate that the improvement in efficiency seen above was due to the expert system's searching knowledge base, not just the provision of an online thesaurus.

The third hypothesis that the expert system could rank passages in decreasing order of relevance was supported. Although the expert system did present relevant passages significantly earlier than would be predicted by randomness, the improvement was not large enough to be considered truly successful. The current algorithm needs to be evaluated with different weights, or a somewhat different algorithm needs to be tried, in order to further improve the ranking function. Decreasing the query term weights more quickly as the query terms move farther from the original may improve the ranking by placing more emphasis on the user's original search terms. Using a more sophisticated closeness factor, one that took into account how many words apart the search terms were in the passage, as well as sentence and paragraph measures considered in this version, may also lead to improved ranking.

Finally, some discussion of the number of reformulations performed by the expert system seems appropriate. The number of reformulations performed for a given question varies since some unsuccessful starting queries were reformulated before (and sometimes after) a successful starting query is found. The following statistics are given for the final query. The average number of reformulations performed on the starting query for the twelve questions, in order, were 4.25, 5.75, 4.25, 2.75, 6, 5.75, 3.25, 3.25, 4.25, 2.75, 4, 1. This gives an average of 3.65 reformulations over all final retrieval set queries. It is interesting to note that the highest average number of reformulation is six. If the expert system is continually broadening the query (which is the most common case), this means that even on the question requiring the most reformulations it stops, on average, just after adding child stemgroups. In fact, examining the 48 final retrieval set queries reveals that only in 3 cases did the expert system go past this point. Twice it went one more step and adjusted the context, and once it performed all ten reformulations on the broadening side before the user was satisfied with the number of passages retrieved.

3.5 Questionnaire

The twelve subjects were asked which features of the expert system they liked best. The automatic addition of terms from the thesaurus was the most frequently mentioned (8 subjects), whereas the automatic context adjustment was the second most popular feature (3 subjects). Many subjects (8) mentioned the decreased amount of work needed to perform a search, with three of them specifically mentioning that they did not have to think as much. Other features mentioned which decreased the user effort were the simplified syntax, decreased typing, and the fewer queries to remember.

System slowness was the feature most disliked (6 subjects). Although the amount of time necessary to answer a question was no greater with the expert system (see Section 3.3.2), there was less work for the user so time seemed longer. The other main complaints concerned the user interface. The subjects were fairly evenly split between wanting the system to proceed more automatically, with less prompting from them (4 subjects), whereas others wanted the system to explain what it was doing and/or allow the user to direct it (5 subjects). These comments lead to the conclusion that if a usable system is to be built based on the success of this research prototype, the execution of the system must be sped up and more work on interface design is needed.

Almost all the subjects (10) found the user-expert system version the easiest to use with the remaining two subjects split between the other two versions. Not surprisingly, given the comparable effectiveness of the three systems, the subjects were split on which system they felt gave the best results. Three voted for the user-alone version, two for the user-thesaurus, and three for the expert system. Three said it was a tie between the user-thesaurus and the expert system, and one abstained.

4. FUTURE WORK

Running the experiment suggested several possible refinements to the system. The experimental subjects had many useful comments, the bulk of which dealt with the desire for a more sophisticated user interface. Desirable changes include provision of a non-Boolean query language; allowing users to adjust the amount of system interaction; having the user specify the type of search desired, rather than having him give a specific target number; and increasing the speed of the system by improving the way the expert system uses MICROARRAS.

Observing the expert system reformulate real queries gave invaluable insight into which types of queries it handled well, and which it did not. Some possible knowledge base improvements will be suggested, but the whole issue of the order in which to apply the search tactics need further investigation.

Currently, the Boolean operators are loosened before negative concepts are removed. It is a more drastic change to replace ANDs with ORs than to drop the negative concepts from the query so the order of application of these two reformulation techniques should be swapped.

A common type of query that required broadening was one that contained the intersection of three or more concepts. In this case, broadening context and adding search terms to each concept fails to address the fundamental problem of intersecting too many concepts. The next step of replacing the ANDs with ORs is too drastic a change. It invariably leads to too broad a query. Instead, the expert system should take the original query and drop each of the concepts in turn. For example, the query 'A AND B AND C' would have partial queries 'A AND B', 'B AND C', and 'A AND C'. The number of passages retrieved by each of the new partial queries should be reported back to the user, and he could restart the expert system on whichever partial query best reflects his interests.

The most common type of query requiring narrowing consisted of a single, high-frequency concept. None of the current reformulation techniques were of any use in this case. There were no operators to change, no context to adjust, and adding search terms just makes the query broader. This type of query should be treated as a special case. The concept's child concepts from the thesaurus should be presented as alternative, more specific, queries. The user could also be encouraged to AND this concept with another.

The treatment of multiword phrases entered by the user which do not appear in the thesaurus should be changed. Currently, the only expansions done are expanding each word to its stemgroup and loosening the context allowed between the words of the phrase. It would be preferable to treat the words of the phrase as separate concepts which are ANDed together with adjacent context. Each phrase word could then be expanded using the full range of thesaural relationships, as is the case with regular search terms.

Finally, more work is needed to improve the ranking of the retrieved passages. The current ranking algorithm should be tried with different weights for the query search terms and the closeness factor. It may be necessary to try entirely different algorithms, possibly incorporating syntactic or semantic information, to achieve high quality ranking.

In conclusion, we have demonstrated that an expert system can provide online search assistance to improve the efficiency of novice searchers. Whereas more research is necessary to develop a better search assistant, I have been able to prove that a useful search assistant can be developed which separates the search strategies from the domain knowledge, and that implementation of such a system is feasible now.

REFERENCES

1. Bates, M. J. Information search tactics. *J. ASIS 30*, 4 (1979), 205–214.
2. Belkin, N. J., Borgman, C. L., Brooks, H. M., Bylander, T., Croft, W. B., Daniels, P., Deerwester, S., Fox, E. A., Ingwersen, P., Rada, R., Spark Jones, K., Thompson, R., and Walker, D. Distributed expert-based information systems: An interdisciplinary approach. *Inf. Process. Manage. 23*, 5 (1987), 387–399.
3. Blair, D. C., and Maron, M. E. An evaluation of retrieval effectiveness for a full-text document-retrieval system. *Commun. ACM 28*, 3 (1985), 289–299.
4. Blaauw G. A., and Brooks, F. P., Jr. *Computer Architecture, Volume 1—Design Decisions*. Draft, Spring 1987.
5. Borgman, C. L. Why are online catalogs hard to use? *J. ASIS 37*, 6 (1986), 387–400.
6. Borgman, C. L. The user's mental model of an information retrieval system: an experiment on a prototype online catalog. *Int. J. Man-Mach. Stud. 24*, 1 (1986), 47–64.
7. Borgman, C. L. Individual differences in the use of information retrieval systems: Some issues and some data. In *Proceedings of the Tenth Annual International ACMSIGIR Conference on Research & Development in Information Retrieval* (New Orleans, June 1987), ACM Press, 1987, 61–69.
8. Brajnik G., Guida, G., and Tasso, C. IR-NLI II: Applying man-machine interaction and artificial intelligence concepts to information retrieval. In *Proceedings of the Eleventh Annual International ACMSIGIR Conference on Research & Development in Information Retrieval* (Grenoble, June 1988), ACM Press, 1988, 387–399.
9. Brooks, H. M. Information retrieval and expert systems—approaches and methods of development. *Informatics 7: Intelligent Information Retrieval* (1983), 65–78.
10. Chang, S., and Chow, A. Towards a friendly adaptable information retrieval system. In *Proceedings of RIAO 88* (Cambridge, MA, March 1988), MIT, 1988, 172–182.
11. Chiaramella Y., and Defude, B. A prototype of an intelligent system for information retrieval: IOTA. *Inf. Process. Manage. 23*, 4 (1987), 285–303.
12. Croft, W. B., and Thompson, R. H. I^{3}R: A new approach to the design of document retrieval systems. *J. ASIS. 38*, 6 (1987), 389–404.
13. Crouch, C. J. A cluster-based approach to thesaurus construction. In *Proceedings of the Eleventh Annual International ACMSIGIR Conference on Research & Development in Information Retrieval* (Grenoble, June 1988), ACM Press, 1988, 309–320.
14. Driscoll, J. R., Rajala, D. A., Shaffer, W. H., and Thomas, D. W. An application of artificial intelligence techniques to automated keywording. In *Proceedings of RIAO 88* (Cambridge, MA, March 1988), MIT, 1988, 500–511.

15. FENICHEL, C. H. Online searching: measures that discriminate among users with different types of experience. *J. ASIS. 32*, 1 (1981), 23-32.
16. HARMAN, D. Towards interactive query expansion. In *Proceedings of the Eleventh Annual International ACMSIGIR Conference on Research & Development in Information Retrieval* (Grenoble, June 1988), ACM Press, 1988, 321-331.
17. MCCUNE, B. P., TONG, R. M., DEAN, J. S., AND SHAPIRO, D. G. RUBRIC: A system for rule-based information retrieval. *IEEE Trans. Softw. Eng. SE-11*, 9 (1985), 939-945.
18. OLDROYD, B. K. Study of strategies use in online searching 5: Differences between the experienced and the inexperienced searcher. *Online Rev. 8*, 3 (1984), 233-244.
19. POLLITT, A. S. A 'front-end' system: An expert system as an online search intermediary. *ASLIB Proc. 36*, 5 (1984), 229-234.
20. POLLITT, A. S. CANSEARCH: An expert systems approach to document retrieval. *Inf. Process. Manage. 23*, 2 (1987), 119-136.
21. RO, J. S. An evaluation of the applicability of ranking algorithms to improve the effectiveness of full-text retrieval. II. On the effectiveness of ranking algorithms on full-text retrieval. *J. ASIS 39*, 3 (1988), 147-160.
22. SALTON, G., AND MCGILL, M. J. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, 1983.
23. SALTON, G. Another look at automatic text-retrieval systems. *Commun. ACM 29*, 7 (1986), 648-656.
24. SHOVAL, P. Principles, procedures and rules in an expert system for information retrieval. *Inf. Process. Manage. 21*, 6 (1985), 475-487.
25. SMEATON, A. F., AND VAN RIJSBERGEN, C. J. The retrieval effects of query expansion on a feedback document retrieval system. *Comput. J. 26*, 3 (1983), 239-246.
26. SMITH, J. B., WEISS, S. F., AND FERGUSON, G. J. MICROARRAS: An advanced full-text retrieval and analysis system. In *Proceedings of the Tenth Annual International ACMSIGIR Conference on Research & Development in Information Retrieval* (New Orleans, June 1987), ACM Press, 1987, 187-195.
27. SMITH, P. J., SHUTE, S. J., GALDES, D., AND CHIGNELL, M. H. Knowledge-based search tactics for an intelligent intermediary system. *ACM Trans. Inf. Syst.* 7, 3 (1989), 246-270.
28. SPARCK JONES, K. Intelligent retrieval. *Informatics 7: Intelligent Information Retrieval* (1983), 136-143.
29. TAGUE, J. AND SCHULTZ, R. Some measures and procedures for evaluation of the user interface in an information retrieval system. In *Proceedings of the Eleventh Annual International ACMSIGIR Conference on Research & Development in Information Retrieval* (Grenoble, June 1988), ACM Press, 1988, 371-385.
30. TESKEY, N. Extensions to the advanced interface management project. *OCLC Res. Rev.* (July 1987), 1-3.
31. TONG, R. M., APPLEBAUM, L. A., ASKMANN, V. N., AND CUNNINGHAM, J. F. Conceptual information retrieval using RUBRIC. In *Proceedings of the Tenth Annual International ACMSIGIR Conference on Research & Development in Information Retrieval.* ACM Press, 1987, 247—53.
32. VERNIMB, C. Automatic query adjustment in document retrieval. *Inf. Process. Manage. 13*, 6 (1977), 339-353.
33. VICKERY, A., AND BROOKS, H. M. PLEXUS-The expert system for referral. *Inf. Process. Manage. 23*, 2 (1987), 99-117.
34. WILLIAMS, P. W. A model for an expert system for automated information retrieval. In *Proceedings of the 8th International Online Meeting* (London, Dec. 1984), 139-149.
35. ZADEH, L. A. Fuzzy sets. *Inf. Control. 8*, 3 (June 1965), 338-353.

Received October 1990; revised December 1990; accepted January 1991