Intelligent Search of Full-Text Databases

Susan Gauch and John B. Smith
Department of Computer Science
Sitterson Hall 083A
University of North Carolina
Chapel Hill, North Carolina, 27514

Abstract

This project applies expert system technology to the task of searching online full-text documents. We are developing an intelligent search intermediary to help end-users locate relevant passages in large full-text databases. Our expert system will automatically reformulate contextual Boolean queries to improve search results and will present retrieved passages in decreasing order of estimated relevance. It differs from other intelligent database functions in two ways: it works with semantically unprocessed text and the expert systems contains a knowledge base of search strategies independent of any particular content domain.

The goals for our current project are to demonstrate the feasibility of the approach and to evaluate the effectiveness of the system through a controlled experiment. While the work we report here has limited objectives, the system and techniques are general and can be extended to large, real-world databases.

1 Introduction

1.1 Motivation

As the cost of computers decreases and their capabilities increase, more and more professionals will use personal workstations to aid them in their work. In most instances, these powerful personal machines will be linked by networks to large mass storage devices, such as laser disks. Consequently, many knowledge workers have, or will soon have, access to large full-text databases in their fields. Without new tools to help them manage and use the large number of texts that will be available on-line to them, these professionals will soon be buried in information.

Searching existing full-text databases currently presents two basic problems:
1) the user must know the technical details of the retrieval system to use it effectively
2) the process is laborious

To avoid the technical details of the retrieval system, many users present their information needs to a trained search intermediary who then searches the online databases for them. This approach creates several problems. Because the user is not involved in the search process, he receives passages that the *searcher* believes are relevant, based on the searcher's understanding of the user's needs. Since the user often has only a vague idea in advance of topics and terms on which to search, several searches are often required, each based on the results of the preceding search.

If the user does his own searching, the process is likely to be laborious and time consuming, particularly if the user is a novice or infrequent searcher. These individuals may use inappropriate search terms and require many iterations to improve their queries. They may become frustrated, unable to find relevant information they are sure the database contains. Or they may be overwhelmed by a flood of marginally relevant passages.

We are attempting to address both problems by providing an online search assistant to handle the technical details and to reformulate search queries automatically. This approach offers the best of both worlds: the user is actively involved in the search process, but he will need less training to achieve satisfactory results.

1.2 Background

Research that relates to our project can be found in several areas. This includes work in user-interface design, information retrieval software, and artificial intelligence.

As the demand for direct access to existing online information retrieval systems has grown, so has interest in providing friendlier interfaces. Marcus [Marcus, 1981] and Meadow [Meadow, Hewett, & Aversa, 1982] describe research prototypes based on conventional programming techniques which make existing bibliographic databases easier to search. These projects have focused on providing menu systems to guide novice users. The menus provide information to the user about choosing the correct database, selecting search terms, and connecting to a remote database. Although many technical details are hidden from the user and information is available online to prompt him, the interaction is still laborious and these interfaces have not been extended to full-text databases.

Many projects have looked at the possibility of allowing users to query databases in natural language, removing the need for them to form Boolean queries. Euzenat and his team [Euzenat, Normier, Ogonowski, & Zarri, 1985] have produced a prototype of a transportable natural language interface to database management systems. This interface transforms the users query into one that can be answered by the relations defined in the database. Defude [Defude, 1984] has proposed a natural language interface to a bibliographic retrieval system incorporating an expert system. Both systems help in the query formation, but do not assist the user in refining that query to improve search results. Again, these systems have not been extended to full-text databases.

The artificial intelligence research that applies most directly to information retrieval is that on question answering systems. These systems build internal knowledge structures from documents in a given area and then synthesize answers to questions based on that structure. One example is "Researcher", under development by Michael Lebowitz [Lebowitz, 1985]. However, building a knowledge structure from natural language text is a slow and error-prone process that is currently not feasible for large, dynamic collections of documents. Even if the knowledge structures could be built and queried effectively for large document collections, most users will probably want to see the actual text of the original documents, not just a synthesized answer to their query. We agree with Karen Sparck Jones that "the language of documents is part of their information content" [Sparck Jones, 1983].

A different use of AI techniques can be seen in several recent projects in Library Science. (Many of these projects are surveyed in [Jones, 1984] and [Smith, 1980].) Most interesting is the research on expert systems that help end-users to do their own searching. Pollitt [Pollitt, 1987] developed a prototype system which aids searches of cancer literature. Walker and Janes [Walker & Janes, 1984] have been working for several years on a system to help search part of the Chemical Abstracts database. More recently, the PLEXUS referral expert system [Vickery & Brooks, 1987] has combined a natural language interface with a knowledge base that contains both domain knowledge on gardening with strategies for searching that domain. Each of these systems represents an important contribution to the future of information retrieval, however all are tailored to searching bibliographic databases in only one specific content area.

What is badly needed is a system that can work with different full-text databases, that can be used by the end user, and that can moderate the output so that the user is not inundated . The system we are developing represents one step toward these goals.

1.3 Functional Overview

The expert system we are developing will serve as the front-end to a full-text database. Our goal is to provide many of the benefits of a search intermediary without the drawbacks. The user will interact with the expert system in a high-level query language. The expert system will deal with the technical details of the textbase. It will also work with the user to refine the query if the initial search is unsatisfactory. If the search produces too many passages, the expert system will reformulate the query by tightening constraints. If it produces too few, it will reformulate the query by loosening constraints and/or expanding the search terms. When an appropriate number of passages have been identified, the expert system will rank order them in terms of their probably interest to the user. Throughout the process, the user remains an active participant in the search, but with the system assuming responsibility for much of the detail. A sample scenario is presented in section 6.4.

2 System Architecture

The system we are developing has five major components:

1) MICROARRAS which serves as the full-text search and retrieval engine
2) a full-text database
3) a hierarchical thesaurus of words specific to the textbase's domain
4) an expert system which interprets the user's queries, controls the search process, analyses the retrieved text, and ranks the search results
5) a user interface which accepts the user's queries, presents requests for information from the expert system, and displays the search results. It is not a major thrust of this project, and is not discussed further in this paper.

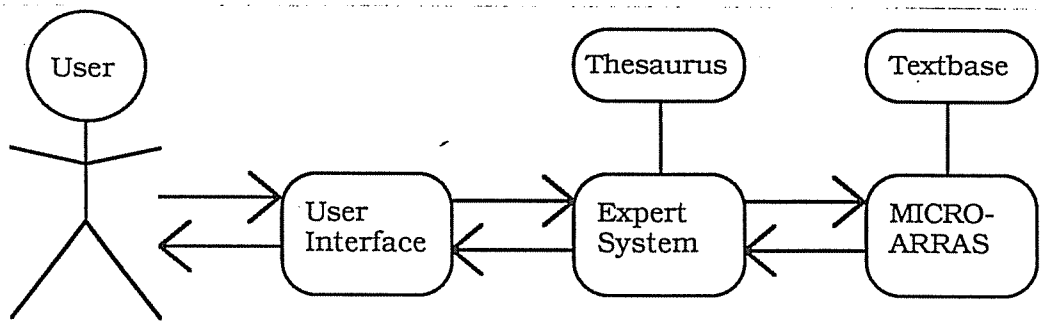

Figure 2.1 System Architecture

The system is being implemented on a Sun 3 workstation. MICROARRAS is written in the C language. The textual database for our current demonstration project consists of an unpublished manuscript on computer architecture written by F. P. Brooks, Jr., and Gerard Blaauw [Brooks & Blaauw, 1987]. The thesaurus construction and access routines are also written in C. For an expert system shell, we are using OPS83.

The search process consists of a dialogue between the user and the expert system. The user enters the initial contextual Boolean query which the expert system translates 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 evaluates the search results and decides whether or not to reformulate the query.

To expand a search query, the expert system may use three different strategies, alone or in combination. Using the thesaurus, it can expand individual search terms to the set of synonyms contained in the domain specific thesaurus. Since the thesaurus is structured as a tree, this process can be iterated several times to include ancestor as well as cousin sets. Second, it can relax contextual constraints. MICROARRAS provides complete generality in terms of segmental contexts. Thus, search expressions may contain contextual parameters in terms of any number of words, sentences, paragraphs, etc. to either the right or left of any term in the search expression. Thus, the expert system can increase the number of such units to generate more potential hits. Finally, it can change the Boolean operators, making the query less restrictive.

To restrict a search, the expert system uses the same strategies as those described above, but in reverse. That is, it may reduce sets of search terms to only the head term listed in the thesaurus, contract contexts, and replace Boolean operators.

Once an appropriate number of passages are identified, the expert system attempts to rank order them in terms of probable relevance. It does this by performing a rudimentary content analysis on the passages retrieved by MICROARRAS and computing a relevance index for each. The relevance index for each passage is a function of the number of search terms actually found in that passage, the number of distinct types for each (for terms that are sets), and the number of different thesaural categories represented. The retrieved passages are then ranked by their relevance indices and presented to the user in order of probable interest.

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. Once the search strategy rules have been developed and tested with the existing textbase, the expert system could be extended to other content domains by simply providing a suitable thesaurus for the new textbase.

3 MICROARRAS

3.1 Capabilities

MICROARRAS is an advanced full-text retrieval and analysis system [Smith, Weiss, & Ferguson, 1986]. 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 also provides Boolean search on any word or set of words in the text. 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. MICROARRAS can also compute and report various frequency of occurrence statistics in the form of distribution vectors over a text or set of texts.

3.2 FLANGE

FLANGE is a two-way command language that was developed as part of the MICROARRAS system. It provides communication between the user interface and the analytic engine that performs all search and analysis operation. Since it is written in a BNF-like notation, programs can easily construct command expressions which, in turn, can easily be parsed. The components of a FLANGE "sentence" are also strongly typed

to further simplify processing and to ensure reliable transmission across a communication interface.

The following example shows 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 enables for an expert system to work iteratively with the textbase.

4 Textbase

The database we are using for our current demonstration project is a manuscript by Brooks and Blaauw consisting of some 188,278 words. While this textbase is small compared to large commercial databases, it is large enough to provide a realistic demonstration environment. It differs from standard full-text databases in that the result of a search is a collection of passages from one large text rather than a collection of passages from a database of many documents.

Texts to be used as MICROARRAS textbases require initial processing. First, format marks of interest to users must be inserted in the text. For the Brooks and Blaauw text, we included format marks which will be used in the display of the retrieved text (line, tab, italics, line, label), as well as those which provide context information (section, paragraph, sentence, item). Second, a series of programs are run on the text to produce an inverted file. Finally, this inverted file is converted to fixed length records for fast access.

5 Thesaurus

All domain-specific knowledge is contained in a hierarchical thesaurus. The expert system uses this information to reformulate queries. The thesaurus was derived from the Brooks and Blaauw text and strongly reflects the word usage of that textbase. For a commercial database of many texts or documents the thesaurus would need to be broader in scope. Ideally, individual users should be able to tailor the thesaurus for better coverage of their areas of interests.

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 Brooks and Blaauw 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*. Throughout this discussion, word types will be written in lowercase, stemgroup names with a leading uppercase letter, and thesaurus classes in uppercase.

At the lowest level words with the same root are grouped into stemgroups. Consider the grouping of for words with the root 'structure':

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

Next, stemgroups pertaining to technical concepts are identified. Synonyms among these stemgroups are combined to form thesaurus classes. Non-technical terms are not included in the thesaurus. Extremely low frequency stemgroups are also excluded, since low frequency stemgroups represent unimportant, little-discussed concepts and excluding them detracts very little from recall while decreasing the size and complexity of the thesaurus.

High frequency stemgroups represent broad concepts discussed throughout the text. Most thesauri replace high frequency stemgroups with multiple word phrases or exclude them altogether to improve precision. In this system high frequency technical stemgroups, for example 'data' and 'structure', are combined to form lower frequency multiple word phrases, e.g. 'data structure', which are included in the thesaurus. However, high frequency stemgroups are also included because the user is likely to use these words, the expert system will use them to direct the user to narrower terms. The expert system can decide how to deal with these stemgroups if they are introduced during thesaurus expansion.

Finally, an ordering is imposed on the thesaurus classes. Conceptually, a thesaurus class can be viewed as a node in a lattice structure. The thesaurus is a lattice rather than a tree as nodes may have more than one parent. Parent nodes—nodes higher in the thesaurus structure—represent more general concepts than the current node. Children 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
Children Nodes(s): ARRAY, QUEUE, STACK, LIST

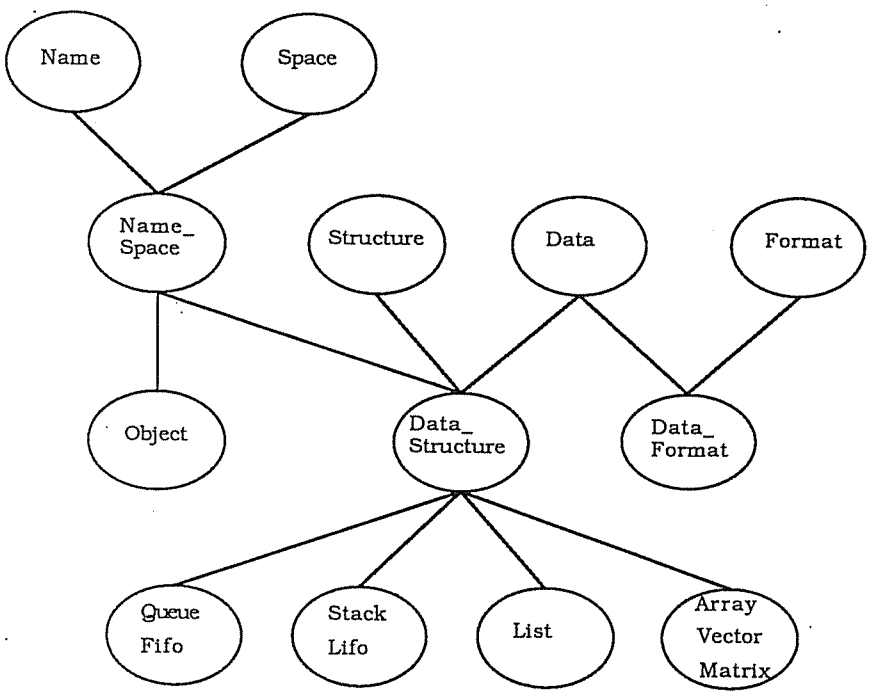

Figure 5.1 A Sample Thesaurus.

The thesaurus was constructed manually from the 8313 different word types in the database. Removing numbers, punctuation, stop words, proper names, and words which

appeared only once left 5726 types. These were grouped into 1993 stemgroups; and common word forms missing from the stemgroups were added, bringing the total to 6990 types. Using a concordance, we selected 936 technical stemgroups from the 1993 to be arranged hierarchically in the thesaurus.

Before the stemgroups were placed into classes of synonyms to be arranged hierarchically, they were first partitioned into loose collections of related terms, called *buckets*. Each stemgroup was placed into one of the following buckets: architecture, sequencing, languages, hardware, formats, control operations, data operations, data representation, operating systems, memory, input/output. High frequency stemgroups were combined to form multi-word phrases which were added to the appropriate buckets. Ambiguous stemgroups were placed in more than one bucket as necessary.

The synonym stemgroups in each bucket were grouped into thesaurus classes, and a lattice structure was imposed on all the classes in the bucket. The concordance was used again to direct the formation and arrangement of the classes. Dealing with one bucket at a time allowed us to use divide and conquer to decrease the complexity of the hierarchical arrangement task. Finally, the lattices for all the buckets were merged.

6 Expert System

The expert system performs two main functions: it reformulates the Boolean query based on previous search results and it ranks the retrieved passages in decreasing order of estimated relevance for presentation to the user. To perform these functions, it uses a knowledge base of search strategies and text analysis procedures. As we pointed out above, all domain knowledge is contained in the thesaurus.

6.1 Query Formulation

To invoke the system, the user forms the initial contextual Boolean query. The Boolean operators available are 'and', 'or', and 'andnot'. The user may specify a context for the query's evaluation in terms of words, sentences, or paragraphs. If no explicit context is given, the expert system assumes a default context of one sentence. The expert system then maps the query into FLANGE (the MICROARRAS two-way control language mentioned above) and sends the FLANGE query to the MICROARRAS engine. The engine performs the search, packages the results into a FLANGE return message, and sends the formatted information back to the expert system. The expert system unpackages the FLANGE message and decides whether to display the results to the user or whether to reformulate the query.

6.2 Query Reformulation

Following the initial search, the decision to reformulate the query is based on estimates of the recall (the percent of all relevant passages identified by the search) and the precision (the percent of retrieved passages that are relevant). The expert system makes these estimates using frequency statistics for the query terms in the textbase as a whole, their frequency in the retrieved passages, and the number of passages retrieved.

If the system decides to reformulate the query, it may do so by manipulating three variables, alone or in combination. They are the number of context units between terms, the search terms, and the Boolean operators.

If the recall estimate is very low, the query will be broadened to match more passages. This may be done by replacing individual words in the search expression by sets of words (categories, in MICROARRAS' terms). Initially, the expert system expands words into stem groups (words with the same root). Next, it replaces words with synonym sets. If

necessary, related word sets will be concatenated. In each case, the sets added are derived from the hierarchical thesaurus. Alternatively, contextual constraints can be relaxed, so that the search extends over adjacent sentences, the whole paragraph, adjacent paragraphs, etc. Finally, the Boolean operators may be changed from "and" to "or", or by removing "not" components from the expression.

Recall that is too high usually does not pose a problem so long as the precision remains high. Since the retrieved passages will be displayed in decreasing rank order, the user can simply stop reading the passages whenever he wishes. The only time this is a problem is when the number is so large that it requires excessive time to rank-order them. In those cases, the expert system manipulates the three variables in reverse.

If precision is too low – i.e. too many irrelevant passages are retrieved – a more specific search expression is required. In this case, the expert system first tries increasing the contextual constraints. If this does not produce the desired results, it replaces query terms with more specific terms. The system derives a candidate term from the thesaurus and then asks the user for confirmation before reformulating the query. A final strategy involves changing the Boolean operators from "or" to "and".

Precision cannot really be too high, since ideally all relevant passages and no irrelevant passages would be retrieved. However, high precision may mask another problem. It may indicate that the query was not broad enough and that, in fact, recall was low. This possibility was discussed above.

The expert system must decide when to stop the reformulation process. This decision will be based on a combination of user supplied *a priori* knowledge of the amount of information desired and analysis of the results of the searches to date. Certainly, the expert system will try to improve recall if nothing at all is retrieved. Similarly, if a great many passages are retrieved, precision must be improved. If the results of this query are worse than previous results, the expert system will backtrack to an earlier query. Finally, if the expert system runs out of things to try, control is returned to the user regardless of the amount of information retrieved.

6.3 Relevance Estimation

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 an index of probable interest. Factors which affect this index value are the number of different concepts represented in the passage, the number of different word types for each concept, the number of tokens 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 index values and presented to the user in decreasing order of estimated relevance.

6.4 Sample Scenario

Since the system is still in the early stages of development, this example is intended to illustrate the type of interaction we expect. It does not yet describe the actual actions of a working prototype.

The user might enter a query "array and [nextword] processor". From our sample textbase, this would retrieve only one passage, although 'array' appears 102 times in the textbase and 'processor' appears 179 times. The expert system would decide to expand the retrieval set by relaxing the context between the search terms to 'sentence', the default

context. The new query would be "array and processor". This would retrieve two passages, still not very satisfactory.

The next step would be to replace the word types 'array' and 'processor' with their stemgroups. The resulting query would be "(array or arrays) and (processor or processors)". Two additional passages would now be retrieved.

Since recall is still low, the expert system would further broaden the query by including synonym stemgroups for the each of the search terms. As the 'array' stemgroup appears fewer times (146) in the textbase than 'processor' (331), its synonyms would be included first, leading to the query "(array or arrays or vector or vectors or matrix or matrices) and (processor or processors)". This improves the number of passages retrieved to six. Adding processor's synonym stemgroups (computer, machine) drastically increases the number of passages retrieved to seventeen. Since an adequate number of passages have now been retrieved, the expert system would rank the retrieved passages and present them to the user in decreasing rank-order. If the user indicates that he wishes the query to be further broadened, stemgroups from the search terms parent and/or sibling nodes in the thesaurus could be added.

7 Experiments

Gerard Salton [Salton, 1983] describes two measures of performance: system effectiveness and efficiency. Basically, the effectiveness of an information system is a measure of the system performance whereas efficiency is a measure of the amount of user effort required to perform a task. Once the system is built, we will run controlled experiments to test whether the expert system can improve a novice searcher's effectiveness and efficiency.

We will use Computer Science graduate students as the subjects since they are proficient computer users but novice searchers. The subjects will be asked to perform several retrieval tasks differing in the amount of information to be retrieved and the difficulty of the searches. Each subject will perform searches with and without the expert system front end, and data will be collected to evaluate their performance. The subjects will also be asked for relevance feedback on the passages retrieved.

Effectiveness will be measured by precision and recall, and efficiency will be measured by the time necessary to perform each search. The system effectiveness and user efficiency, with and without the expert system, will be compared to evaluate the the impact of the online search assistant. Furthermore, the user's relevance judgements will be compared with the relevance estimates produced by the expert system.

8 Project Status

The text retrieval software, textbase, and thesaurus are complete; and the high level strategies for the expert system have been designed. We are currently writing the production rules to be used by the expert system. We expect to have a working prototype by spring 1988 and to run the experiments described above during the summer.

We view our current project as a beginning, rather than an end in itself. As mentioned above, it is intended to demonstrate the concept of using an expert system as an intermediary function between a user interface and an analytic engine. In the future, we will extend the search and analysis operations that are leveraged by the expert system. These include a broader range of retrieval algorithms and more sophisticated content analysis to determine probable relevance. We will also explore computing and interpreting a variety of statistical and stylistic measures, and we plan to develop an informal graphical query language in which to specify the initial search request.

9 Bibliography

Brooks, F.P. and Blaauw G.A., "Computer Architecture, Volume 1 – Design Decisions", Draft, Spring 1987.

Defude, B., "Knowledge Based Systems Versus Thesaurus: An Architecture Problem about Expert Systems Design", *Research and Development in IR: Proceedings of the 3rd Joint BCS and ACM Symposium*, C.J. von Rijsbergen (ed.), Cambridge University Press, 1984. pp. 267–280.

Euzenat, Bernard, Normier, Bernard, Ogonowski, Antoine and Zarri, Gian Piero, "SAPHIR + RESEDA: A New Approach to Intelligent Data Base Access", *Proceedings of the 9th IJCAI*, Vol. 2, 1985. pp. 855–857.

Jones, Kevin P., "The Effects of Expert and Allied Systems on Information Handling: Some Scenarios", *ASLIB Proceedings*, Vol. 36, No. 5, May 1984. pp. 213–217.

Lebowitz, Michael, "RESEARCHER: An Experimental Intelligent Information System", *Proceedings of the 9th IJCAI*, Vol. 2., 1985. pp. 858–862.

Marcus, Richard S., "An Automated Assistant for Information Retrieval", *Proceedings of the 44th ASIS Annual Meeting*, Vol. 18, 1981. pp. 270–273.

Meadow, Charles T., Hewett, Thomas T., and Aversa, Elizabeth S., "A Computer Intermediary for Interactive Database Searching. I. Design", *Journal of the ASIS*, Vol. 33, No. 4, November 1982. pp. 325–332.

Pollitt, A.S., "CANSEARCH: An Expert Systems Approach to Document Retrieval", *Information Processing & Management*, Vol. 23, No. 2, 1987. pp. 119–136.

Salton, G., and McGill, M.J., *Introduction to Information Retrieval*, McGraw-Hill, New York, 1983.

Smith, Linda C., "Artificial Intelligence Applications in Information Systems", *Annual Review of Information Science and Technology*, American Society for Information Science, Vol. 15, 1980. pp. 67–105.

Sparck Jones, Karen, "Intelligent Retrieval", *Informatics 7: Intelligent Information Retrieval*, Kevin P. Jones (ed.), 1983. pp. 136–143.

Vickery, A., and Brooks, H.M., "PLEXUS – The Expert System for Referral", *Information Processing & Management*, Vol. 23, No. 2, 1987. pp. 99–117.

Walker, G., and Janes, J., "Expert Systems as Search Intermediaries", *Proceedings of the 47th ASIS Annual Meeting*, Vol. 21, 1984. pp. 103–105.