Query Reformulation Strategies for an Intelligent Search Intermediary

Susan Gauch and John B. Smith

Department of Computer Science
University of North Carolina
Chapel Hill, North Carolina, 27599-3175

Abstract

This paper describes an intelligent search intermediary to help end-users locate relevant passages in large online full-text databases. Passage retrieval has advantages in efficiency and effectiveness over traditional document retrieval yet is more computationally tractable than question answering systems under development by researchers in artificial intelligence. However, casual users need assistance with search strategies for full-text databases. We provide an expert system which automatically reformulates contextual Boolean queries to improve search results and ranks the retrieved passages in decreasing order of estimated relevance. It differs from other intelligent database functions in two ways: it works with semantically and syntactically unprocessed text and the expert systems contains a knowledge base of domain independent search strategies.

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 will make the storage and distribution of large collections of online text feasible. Proliferation of personal workstations, combined with modems, are allowing an increasing number of end-users to do their own searching of online databases. All these trends lead to end-user searching of online full-text databases, or *textbases*.

The main roadblock to wide-spread use of online textbases will soon be the inability of end-users to search effectively. Christine Borgman [1] finds that individuals differ greatly in their search ability. In fact, more than a quarter of the subjects in her study were unable to pass a benchmark test of minimum searching skills. Carol Fenichel [2] has found that many experienced searchers could greatly improve their searching, particularly in the area of search strategy. To address these problems we are developing a query reformulation expert system to act as a front-end to a textbase. By providing such a tool novice searchers should be able to search more effectively, thus removing a major roadblock to usage of online text.

1.2 Research Overview

Our goal is to demonstrate that an expert system can improve a novice searcher's retrieval performance. We will evaluate the system using two measures of performance described by Gerard Salton [3]: 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.

The expert system contains a knowledge base of domain independent search strategies. Knowledge specific to the textbase's domain is contained in a thesaurus. The user is asked to supply an initial Boolean query and a target number, an estimate of the number of paragraphs they would like to read. The expert system reformulates the user's query, using the rules in its knowledge base, and information from the thesaurus, to retrieve the target number of paragraphs, or passages, from the textbase. The retrieved passages are ranked in decreasing order of presumed importance before they are presented to the user.

Passage retrieval attempts to present the most relevant passages of a textbase in response to a user's query. John O'Connor [4] distinguishes between *answer-reporting* and *answer-indicative passages*. Answer-reporting passages contain information from which an answer to the question can be inferred, perhaps requiring the user to have specialized background knowledge. Answer-indicative passages allow the user to infer that the containing document also contains an answer-reporting passage. He asserts that an answer-indicative passage is usually near an answer reporting passage.

Answer-indicative, and answer-reporting, passage retrieval can be applied to a multi-document textbase to locate *answer-reporting papers*, papers containing answer-reporting passages. The user can use the retrieved passages to quickly identify the most promising documents, and reject irrelevant documents. In this case, passage retrieval performs a type of automatic abstracting function.

In a single or multi-document environment answer-reporting passage retrieval can perform a question answering function. The goal is to retrieve passages which directly provide the information necessary to answer the user's question. This is the type of retrieval performed by our system.

1.3 Related Work

Our system needs to be considered in the context of two areas of research: first, passage retrieval is compared and contrasted with other types of information retrieval; second, the relationship of our expert system to other expert system user-interfaces is discussed.

<u>1.3.1 Information Retrieval</u> Passage retrieval is a compromise between traditional document retrieval and knowledge-based question-answering systems. The former retrieves documents based on matching query terms with terms used to index documents [3]. The latter builds a knowledge structure from

natural language text from which answers are inferred, for example, RESEARCHER by Lebowitz [5]. Passage retrieval requires less pre-processing than either. The documents do not need to be indexed as their contents are searched directly. Since no knowledge base of the document contents is formed, the documents do not require syntactic or semantic pre-processing. O'Connor [6] identifies several more advantages of passage retrieval, and discusses its feasibility.

In terms of user efficiency, passage retrieval is more efficient than document retrieval since only selected passages from each document must be read. However, it is less efficient than question-answering systems as the user must read the retrieved passages, and infer his own answer rather than receiving a direct answer to his question. We believe that there are many applications, for example scholarly research, for which it is more appropriate to present the actual text of a document than its distilled contents. We agree with Karen Sparck Jones [7] that "the language of documents is part of their information content".

1.3.2 Expert Systems Several prototype expert systems for information retrieval are under development. Many are strongly tied to their domain of application, including domain specific information in the rule base.

CANSEARCH [8] is a rule-based expert system written in PROLOG to search cancer therapy literature. Knowledge of the terms used to index cancer literature are built into a hierarchically structured menu-based user interface. The expert system guides the user through the menus, and forms a syntactically correct query to retrieve indexed documents from the MEDLINE database. Query reformulation, if needed, is left to the user.

RUBRIC [9] searches for word patterns in online text, rather than retrieving pre-indexed documents. However, RUBRIC requires users to tailor the rule base by specifying the word patterns for which to search. While this allow to have specialized rule bases for each user's area of interest, it requires a lot of work to adapt the system for new subject areas.

PLEXUS [10] is an expert system to retrieve references on gardening. Much of the rule base for PLEXUS is used to translate from a natural language query to a syntactically correct bibliographic search request. PLEXUS does have a module of domain independent search strategies which contains no techniques for narrowing queries, and three techniques for broadening queries. The broadening techniques it employs are 1) replacing *ands* with *ors*, 2) dropping terms from the query, and 3) replacing a term by its parent.

OCLC [11] is also in the early stages of developing an expert system interface for passage retrieval. Their system uses the table of contents and the index at the back as sources of domain knowledge for an online book. The prototype does not automatically reformulate queries, but rather makes suggestions to the user based on whether the current query was too broad or too narrow. Currently its rule base contains only one broadening technique - replacing broad terms with multi-word phrases, and two narrowing techniques - dropping or *oring* together terms which have been *anded*.

Similar to RUBRIC and the OCLC project our system searches directly in the text. We also separate the search strategies knowledge base from the domain knowledge, as do PLEXUS and OCLC. Unlike any of the above systems, however, we use an online thesaurus as the source of domain knowledge. Our system also contains a much richer collection of reformulation rules than any of the expert systems mentioned above.

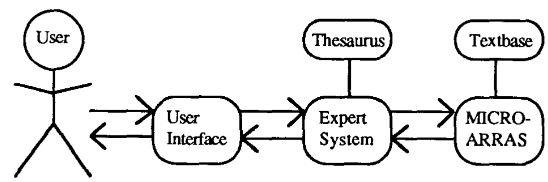

Figure 1. System Architecture

2 System Architecture

The system we are developing has five major components (see Figure 1):

1) MICROARRAS, a full-text retrieval and analysis engine developed by Smith, Weiss and Ferguson [12];
2) a full-text database on computer architecture by Brooks and Blaauw [13];
3) a hierarchical thesaurus of words specific to the textbase's domain;
4) an expert system controls the search process, reformulates the query, 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.

This paper describes the query reformulation rule base of the expert system in detail. A description of the other system components is presented by Gauch and Smith in [14]. The system is being implemented on a Sun 3 workstation. The expert system is written in OPS83, and the other system components are implemented in the C language.

2.2 Functional Overview

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. The Boolean operators available are *and*, *or*, and *andnot*. The expert system assumes a default context of one sentence for *and* and *andnot* operators. In this prototype, the user is also asked to supply the number of passages they would like to see, the target number.

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 reformulate a search query, the expert system uses three different techniques, alone or in combination: first, using the thesaurus, it can add to the sets of search terms; second, it can adjust contextual constraints on the Boolean operators; third, it can replace the 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.

3 Knowledge Base

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, it uses a working memory elements to describe the current state of the reformulation, a knowledge base of search strategies and passage ranking procedures. As we pointed out above, all domain knowledge is contained in the thesaurus.

One of the advantages of rule-based systems is that they are data-driven. The knowledge base is a collection of rules which fire based on the current contents of working memory. Thus, the control of the system can be explained in terms of the rules in the knowledge base and the working memory elements (data) that cause them to fire. Section 3.1 describes the reformulation process used by the expert system and section 3.2 describes the structure of the knowledge base which realizes this process. The passage ranking algorithm has been implemented but not yet evaluated. Our preliminary work in this area is not described here for reasons of brevity.

3.1 Query Reformulation Techniques

Following the initial search, the decision to reformulate the query is based on the target number, the number of passages retrieved, and the history of broadening and narrowing techniques already applied. Figure 2 diagrams the flow of control among these techniques. This figure is somewhat simplified as 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 left side of the Figure 2 diagrams the broadening techniques, the right side the narrowing techniques.

Examining the figure in more detail we see that the first broadening technique used is adding stemwords to positive concepts, followed by adding synonyms. Next, the expert system increases context simultaneously in three ways: strict adjacency between terms in multi-word phrases is relaxed to the component words appearing, in any order, within three words of each other; the context around positive operators is loosened from the same sentence to +/- one sentence; and the context around negative operators is tightened to +/- seven words. Related terms from the hierarchical thesaurus are added next: words from parent classes first, followed by siblings, and finally children. Context is then further broadened such that terms from multi-word phrases are required to appear within the same sentence, positive operators are evaluated with a context of the same paragraph, and negative operators have their context decreased to +/- three words.

More drastic approaches are attempted if the previous techniques do not broaden the query enough. These affect the Boolean operators in the query. First the positive operators are loosened from 'and' to 'or', while negative operators are tightened from 'or' to 'and'. If the query still requires broadening the expert system removes the negative portions of the query altogether.

Narrowing techniques are identical to broadening techniques but are applied to the opposite parts of the query. We narrow by expanding terms in the negative concept groups, tightening positive context, loosening negative context, tightening positive operators and loosening negative operators. The right side of Figure 2 shows the order in which these techniques are applied.

The expert system stops the reformulation process when the target number has been reached, or it has run out of techniques to try.

3.2 Knowledge Base Design

In conventional programs design is discussed in terms of data structures and algorithms. The analogous discussion for and expert system covers working memory elements (in section 3.2.1) and the rule base (in section 3.2.2). The working memory elements contain the information available to the expert system describing the current state of affairs, while the rule base contains the system's knowledge of search strategies.

3.2.1 Working Memory Elements There are seven types of working memory elements, *wme*:

1) start: created to startup the system
2) goal: the current high-level goal
3) reform: contains information about the reformulation state
4) query: contains information about the query as a whole
5) concept: contains information about the concept group; one per concept
6) stem: contains information about the stemgroup; one per stemgroup
7) passage: contains information about the retrieved passage; one per passage

The wme attributes, and their contents will now be described in more detail. Several attributes contain indices into C structures which contain further information. Information is duplicated, and additional is information stored, in C structures mainly for reasons of efficiency, and they will not be described in this paper.

start. No attributes.

Created upon system startup, it triggers the creation of a MICROARRAS process and a goal wme, then is removed.

goal. Attributes:
type: current goal, e.g. 'get query', or 'reformulate'
subgoal: temporary goal during reformulation, e.g. 'add stemword'
justright: target number entered by user
toomany: maximum number of passages, set to justright*1.5
toofew: minimum number of passages, set to justright * 0.5

The first goal type is 'get query'. This causes a rule to fire which receives a query from the user and creates the query, concept, and stem wmes. Each search term entered by the user is assumed to represent a distinct concept and a concept and stem wme is created for each term.

query. Attributes:
status: indicates queries reformulation status, e.g. 'new', or 'reformulated', or 'final'
version: index into C array for this query's parse tree
name: name of MICROARRAS's representation of this query
numpassages: the number of passages retrieved by this query

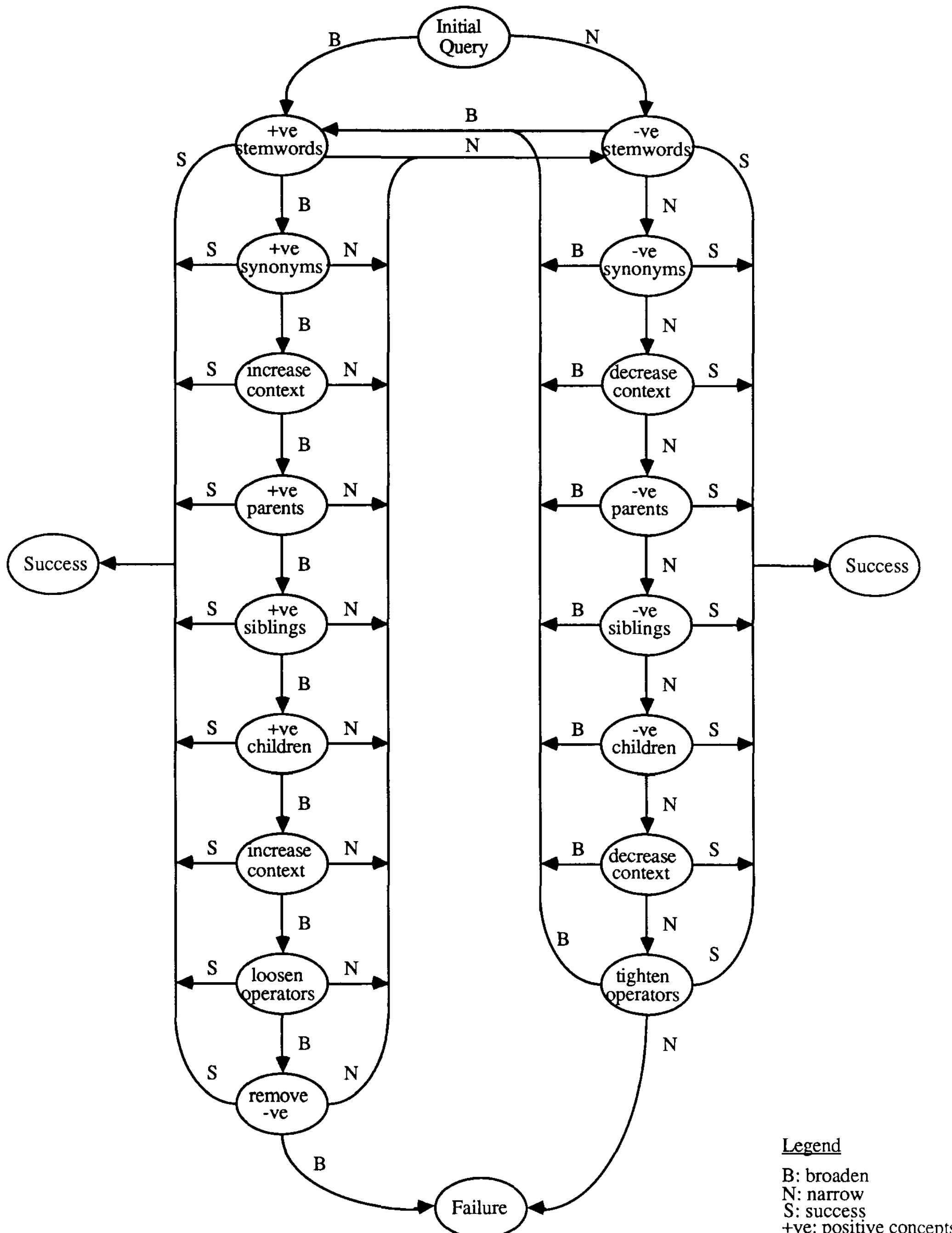

Figure 2. Reformulation Techniques

context: number representing the current context for the query
hi: the broadest context tried with this query so far
lo: the narrowest context tried with this query so far

There is only one query active at a given time. Information about the structure of the query is stored in a C array of parse trees. When the Boolean operators are changed or removed, the current query's status is set to 'reformulated' and a new parse tree is created. A new query wme is created with a new version and name. Changes to context or sets of search terms do not require new query wme, merely an update to the name field to reflect the new MICROARRAS category, or the context and hi or lo fields to reflect the new context. The numpassages field is also updated to reflect the number of passages retrieved at each step along the way.

concept. Attributes:
status: indicates whether or not this concept is in current query, e.g. 'active'
id: index into C array for this concept
sign: indicates whether concept is positive or negative, e.g. '+' or '-'
freq: number of occurrences of all stemgroups in the concept in the textbase
state: the last reformulation performed on the concept, e.g. 'add synonyms'

A concept wme is created for each search term in the user's initial query. The expert system distinguishes between the concepts on which the user wishes information, the positive concepts, those which are to be excluded, the negative concepts. For example, the query 'boundary and word andnot page' contains two positive concepts, 'boundary' and 'word', and one negative concept, 'page'. Additionally, the *and* operator is considered positive, while the *andnot* is negative. The operators are so tagged in the parse tree.

stem. Attributes:
id: index into C array for this stemgroup
concept: identifier for the concept containing this stemgroup
name: English name for the stemgroup, e.g. 'boundary'
freq: number of occurrences of this stemgroups in the textbase
added: reformulation step which caused this stemgroup to be added to the query

Initially, a stem wme is created for each concept in the user's query which contains only the user's search term. Reformulation may cause the other members of the stemgroup to be added, which updates the freq field, but doesn't cause a new stem wme to be created. Further reformulation may add whole new stemgroups to the concepts, causing new stem wmes to be created.

reform. Attributes:
global: the type of reformulation required by the user's initial query, e.g. 'broaden'
local: the opposite of global
lastglobal: the last reformulation technique tried in the global direction
lastlocal: the last reformulation technique tried in the local direction
laststate: the last reformulation technique
step: counts the number of reformulations performed

If the user's query needs reformulation, i.e. does not retrieve the target number of passages, a reform wme is created. The type of reformulation to be performed on the user's query is stored in global. Our sample query of 'boundary and word andnot page' would retrieve one passage. If the target number was 15, global would be set to 'broaden', local to 'narrow'. Lastglobal, lastlocal, and laststate would all be set to 'original' as no reformulation has yet been done. Step is initialized to 0.

passage. Attributes:
status: 'new', or 'ranked', or 'displayed'
contents: index into C array for this passage
rank: estimated rank of this passage

After the query reformulation stops, the passages corresponding to the final query are retrieved, ranked, and displayed. A passage wme is created for each passage, and information stored on it as indicated above.

3.2.2 Rule Base Based on the presence, and contents, of the wmes described above, rules in the knowledge base fire. For each of the reformulation techniques described in section 3.1 there are a set of rules to recognize, based on the current contents of working memory, that the technique is applicable and to carry out the required actions.

For example, consider the set of rules to add new stemgroups. There are several rules to identify sets of candidate stemgroups for a given concept. Synonyms, parents, etc. are all possible candidate sets. Based on the reformulation history one of these sets is selected.

Next, a rule to remove the high frequency candidates from the set is fired. This rule is triggered by the subgoal field containing 'frequencyfilter'. Normally thesauri do not contain high frequency terms, but they appear in this thesaurus as parents of multi-word phrases and need to be filtered out by the expert system. Then, the subgoal field is set to 'duplicatefilter' and another rule fires which removes stemgroups that already appear in the query.

Finally, rules fire to add the stemgroups that pass through the previous two filters to the concept one at a time. As each is added, the resulting number of passages is determined. If the effects of adding the stemgroup are too drastic, the system backtracks by removing the stemgroup. When all of the stemgroups for a given concept have been processed, the expert system processes the next concept. When all concepts have been processed, the number of passages retrieved is compared to the target number and the decision is made to broaden, narrow, or rank and display the passages.

4 Sample Scenario

Since our current textbase concerns the domain of computer architecture the following scenario describes the interactions of the system and a user searching for information on computers with specialized architectures for array processing.

The user might enter a query 'array_processor', which indicates that the user wishes to retrieve passages containing the multi-word phrase 'array processor'. This would retrieve only one passage, although 'array' appears 102 times in the textbase and 'processor' appears 179 times. The first step would be to replace the word types 'array' and 'processor' with their stemgroups. The resulting query would be '(array or arrays)_ (processor or processors)'. Still, only one passage would now be retrieved.

The next step would be to broaden the query by including synonym stemgroups for each of the search terms in turn. Since 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)_(processor or processors)'. This improves the number of passages retrieved to three. Adding processor's synonym stemgroups (computer, machine) doesn't increase the number of passages retrieved at all. In fact, 'computer' would not be added at all since it would be filtered out for being too high frequency a word.

The next technique employed by the expert system would be to increase the allowable context between the phrase terms from strict adjacency to within 3 words. This new query would retrieve eight passages, and the user might stop the reformulation. If the user continues to broaden the query, stemgroups from the search terms parent, sibling, and children nodes in the thesaurus would be added. Context would also be increased to look for the phrase terms within the same sentence rather than 3 words apart.

When an adequate number of passages have been retrieved, the expert system would rank the retrieved passages and present them to the user in decreasing rank-order.

5 Conclusions

5.1 Current Status

The text retrieval software, textbase, and thesaurus are complete; and the second version of the strategic rule base for the expert system has been implemented.

As well as the independent operation mode described here the expert system can operate in an interactive mode. The user is asked to filter candidate stems before they are added to a concept, and direct the system to continue reformulating the query or to return control to the user so they can enter a new starting query. The choice of which reformulation strategy to employ next is left to the expert system.

An experiment involving 12 users has been run to evaluate the search results of users with different systems. The systems compared are: users searching on their own with no online thesaurus; users searching with an online thesaurus; users searching with the interactive expert system; and finally the stand-alone expert system searching on its own. The results of these experiments have not yet been statistically analysed, but they do seem to indicate that users can retrieve relevant information with less effort (measured by the number of queries entered) with the interactive expert system.

5.2 Future Work

In the near term, the data from the experiment needs to be analysed in more detail. Also, observing users search has suggested several refinements to the system. Specifically, more sophisticated narrowing techniques are necessary. When a user enters a very broad single term as a query, e.g. 'computer', the expert system should present the children of 'computer' from thesaurus as alternative queries.

Similarly, when too many concepts are *anded* together requiring extensive broadening subsets of the original concepts should be tried as alternate starting queries. This should be done after the context has been broadened to maximum but before the *ands* are replaced with *ors*.

The issue of how much, and what type of, interaction is best is a human factors issue that should be explored. Other long term goals include extending the search and analysis operations that are leveraged by the expert system. In particular, we would like to look at the effect of using confidence factors to guide the search.

Different reformulation techniques have different levels of confidence associated with them. Candidate search terms also have levels of confidence based on the closeness of their relationship to the user's original word. The query-concept-stem hierarchy suggests that an expert system shell incorporating frames as well as rules may be appropriate for this application. We would like to do more sophisticated content analysis to determine probable relevance, hopefully incorporating some natural language processing techniques. We also plan to develop an informal graphical query language in which to specify the initial search request.

6 Bibliography

[1] Borgman, Christine L., "Individual Differences in the Use of Information Retrieval Systems: Some Issues and Some Data", *Proceedings of the Tenth Annual International ACMSIGIR Conference on Research & Development in Information Retrieval*, C. J. von Rijsbergen and C. T. Yu (ed.), ACM Press, 1987. pp. 61-69.

[2] Fenichel, Carol Hansen, "The Process of Searching Online Bibliographic Databases: A Review of Research", *Library Research*, Vol. 2, No. 2, 1980. pp. 107-127.

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

[4] O'Connor, John, "Retrieval of Answer-Sentences and Answer-Figures from Papers by Text Searching", *Information Processing & Management*, Vol. 11, No. 5, 1975. pp. 155-164.

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

[6] O'Connor, John, "Answer-Passage Retrieval by Text Searching", *Journal of the ASIS*, Vol. 31, No. 4, July 1980. pp. 227-238.

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

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

[9] Tong, Richard M., Applebaum, Lee A., Askmann, Victor N., and Cunningham, James F., "Conceptual Information Retrieval using RUBRIC", *Proceedings of the Tenth Annual International ACMSIGIR Conference on Research & Development in Information Retrieval*, C. J. von Rijsbergen and C. T. Yu (ed.), ACM Press, 1987. pp. 247-253.

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

[11] Teskey, Niall, "Extensions to the Advanced Interface Management Project", OCLC Research Review, July 1987. pp. 1-3.

[12] Smith, John B., Weiss, Stephen F., and Ferguson, Gordon J., "MICROARRAS: An Advanced Full-Text Retrieval and Analysis System", Proceedings of the Tenth Annual International ACMSIGIR Conference on Research & Development in Information Retrieval, C. J. von Rijsbergen and C. T. Yu (ed.), ACM Press, 1987. pp. 187-195.

[13] Brooks, F. P., Jr. and Blaauw, G. A., "Computer Architecture, Volume 1 - Design Decisions", Draft, Spring 1987.

[14] Gauch, Susan, and Smith, John B., "Intelligent Search of Full-Text Databases", *Proceedings of RIAO 88*, Vol. 1, March 1988. pp. 162-171.