(4)

To appear in Information Processing and Management.

An Expert System for Searching in Full-Text

Susan Gauch and John B. Smith
Department of Computer Science
University of North Carolina
Chapel Hill, North Carolina, 27599-3175

N00014-86-K-0680

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 automatically reformulates contextual Boolean queries to improve search results and presents 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. INFORMATION SCIENCE (JES)

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 requires the use of appropriate search strategies

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 information retrieval software, user-interface design, and artificial intelligence.

1.2.1 Information Retrieval Software

A bibliographic document retrieval system contains document surrogates, consisting of bibliographic information sometimes supplemented by an abstract. These surrogates are assigned index terms, either manually or automatically, by which they may be retrieved. In contrast, full-text retrieval systems contain the entire contents of their documents online, and the document contents themselves may be searched. Their main advantages are the elimination of the indexing task and the ability of the user to review the results of a search online, rather than having to locate a copy of the referenced documents. Some early full-text systems are the STAIRS system from IBM and the WESTLAW database of legal information distributed by the West Publishing Company. Other, more recent, full-text systems include CONTEXT [1] and IRX [2].

Both on types of retrieval systems share the two problems identified in the previous section. The mechanical problems are being addressed by research into improved user-interfaces. The intellectual difficulties in performing a search are being addressed by the investigations into the application of artificial intelligence techniques.

1.2.2 User-Interface Design

The CONIT system [3] was one of the first menu-driven interfaces to a bibliographic database. Its menus guide users through the stages of performing a search and provide online help information. The IIDA system [4] could provide online instruction as well as monitor the user's performance, making suggestions if errors were detected. Both 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.

More recently, the F-TAS system [5] makes use of high-resolution graphics displays common today to provide a menu/mouse/icon based interface to a full-text database. This system provides a very nice working environment, but does not address the problem of search strategy formation.

1.2.3 Natural Language Processing

Natural language processing is one area of artificial intelligence that has been applied to the problems in information retrieval. 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 [6] have produced a prototype of a transportable natural language interface to database management systems. This interface transforms ther users query into one that can be answered by the relations defined in the database. This system helps in the query formation, but does not assist the user in refining that query to improve search results.

An example of a natural language interface for a full-text database is the IOTA system of Chiaramella and Defude [7]. It parses natural language queries to select search terms and incorporates a simple query reformulation scheme.

Natural language processing has also been applied to the database contents, as well as the user queries. Question-answering systems build internal knowledge structures from documents in a given area and then synthesize answers to questions based on that structure. One example is the SCISOR system [8] which builds a knowledge base in the domain of corporate mergers and takeovers from which it answers natural language queries with natural language answers. 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" [9].

A very promising hybrid system is being developed by Croft and Lewis [10]. The ADRENAL system creates a frame to represent the meaning of the user's English query, incorporating the user's original wording. From the frame's contents, search terms are chosen and a traditional statistical search of a full-text database is performed. Further natural language processing is performed only on parts of the retrieved documents to rank them before presentation of the most probably relevant to the user.

1.2.4 Expert Systems

Expert systems is another branch of artificial intelligence that is being applied to information retrieval 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 [11] is a rule-based expert system written in PROLOG to search cancer therapy literature in the MEDLINE database. It retrieves indexed documents from the MEDLINE database.

The CODER system [12] is an ambitious project which incorporates basic natural language processing with expert systems techniques to produce a testbed for evaluating advanced information retrieval techniques. It is designed for the domain of electronic mail messages, and employs a distributed processing architecture. The expert system is used to identify the structure within a message, and to identify semantic relationships between messages.

RUBRIC [13] searches for word patterns in online text, rather than retrieving pre-indexed documents. However, RUBRIC requires users to tailor the rule base. While this allows for 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 [14] is an expert system to retrieve references on gardening. While PLEXUS contains a module of domain independent search strategies it differs from our project by retrieving from a database of indexed document abstracts. OCLC [15] is also in the early stages of developing an 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.

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 the OCLC system, however, we use an online thesaurus as the source of domain knowledge.

1.3 Functional Overview

The expert system we are developing serves 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 interacts with the expert system in a high-level query language. The expert system deals with the technical details of the textbase. It also works with the user to refine the query if the initial search is unsatisfactory. If the search produces too many passages, the expert system reformulates the query by tightening constraints. If it produces too few, it reformulates the query by loosening constraints and/or expanding the search terms. When an appropriate number of passages have been identified, the expert system rank orders them in terms of their probable 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.

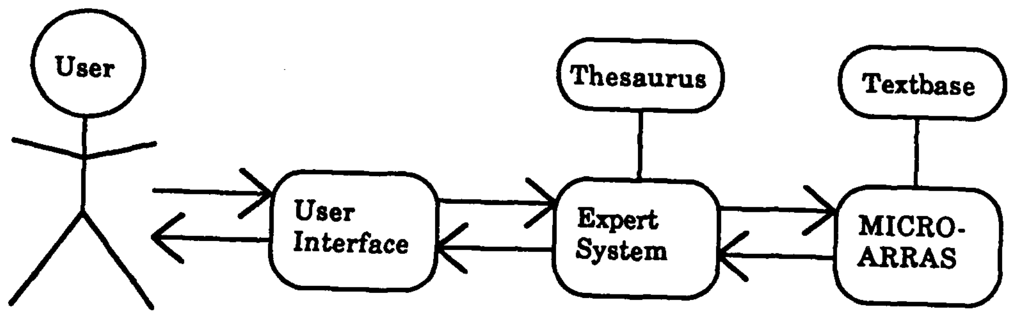

Figure 2.1 System Architecture

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 [16]. 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 uses three different techniques, 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 terms from more general, more specific, or other related thesaurus classes. 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 [17]. 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.

To be inserted into MICROARRAS' textbase, documents must first be inverted. 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.

3.2 FLANGE

FLANGE is a two-way command language that was developed as part of the MICROARRAS system. Consequently, 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.

4 Textbase

The manuscript by Brooks and Blaauw [16] we are using for our current project consists 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 the search is a collection of passages from one large text rather than a collection of documents from a database of many documents. As discussed by O'Connor [18] passage retrieval "speeds both user access to wanted information and the screening out of false retrievals."

Texts to be used as MICROARRAS textbases require initial processing. First, format marks of interest to users must be inserted in the text. For this text, we included format marks which are 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*, or *nodes*, contain stemgroups which are synonyms for each other. Stemgroups may appear in zero, one, or more than one thesaurus classes. 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 together in 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. 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 we combine high frequency technical stemgroups, for example 'data' and 'structure', to form lower frequency multiple word phrases, e.g. 'data structure' which are included in the thesaurus. However, we also include the high frequency stemgroups because the user is likely to use these words, and the expert system uses the thesaurus to direct them to narrower terms. The expert system can exclude these high-frequency 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. 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 children nodes. 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 multi-word phrases have as parents the nodes containing each of the component stemgroups. For example, consider the thesaurus entry for the phrase Data_Structure.

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

The thesaurus was constructed manually from the 7313 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 936 technical stemgroups were selected from the 1993 to be arranged hierarchically in the thesaurus.

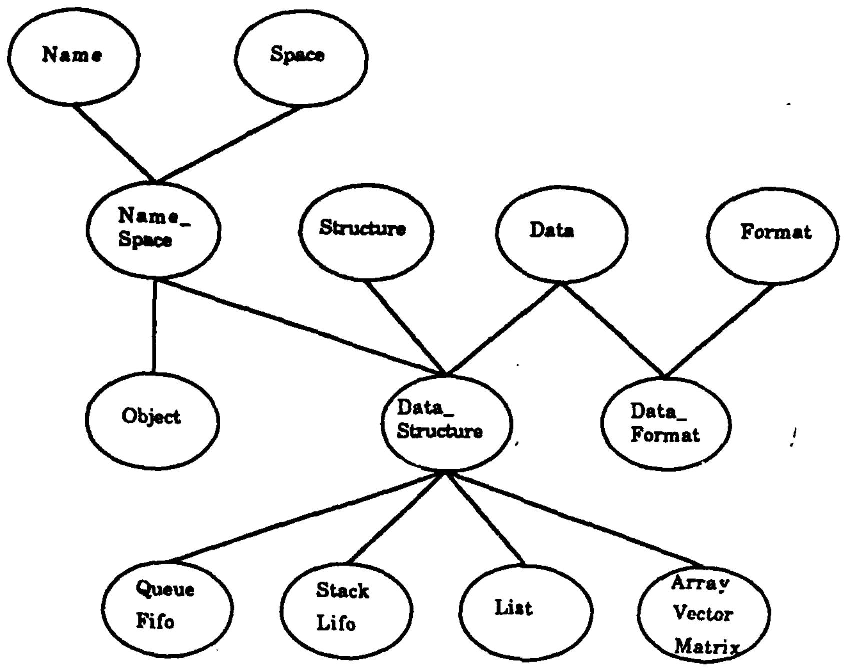

Figure 5.1 A Sample Thesaurus.

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 together.

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

The expert system distinguishes between the parts of the query which indicate information desired by the user, the *positive* search terms and operators, and those parts which describe information to be excluded, the *negative* search terms and operators. Each search term is assumed to represent a distinct concept desired, or not desired, by the user. For example, the query 'boundary and word andnot page' contains two positive concepts, 'boundary' and 'word', and one negative concept, 'page'. The 'and' operator is considered positive, while the 'andnot' is negative.

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 FLANGE, and sends the formatted message 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 the target number, the number of passages retrieved, and the history of broadening and narrowing techniques already applied.

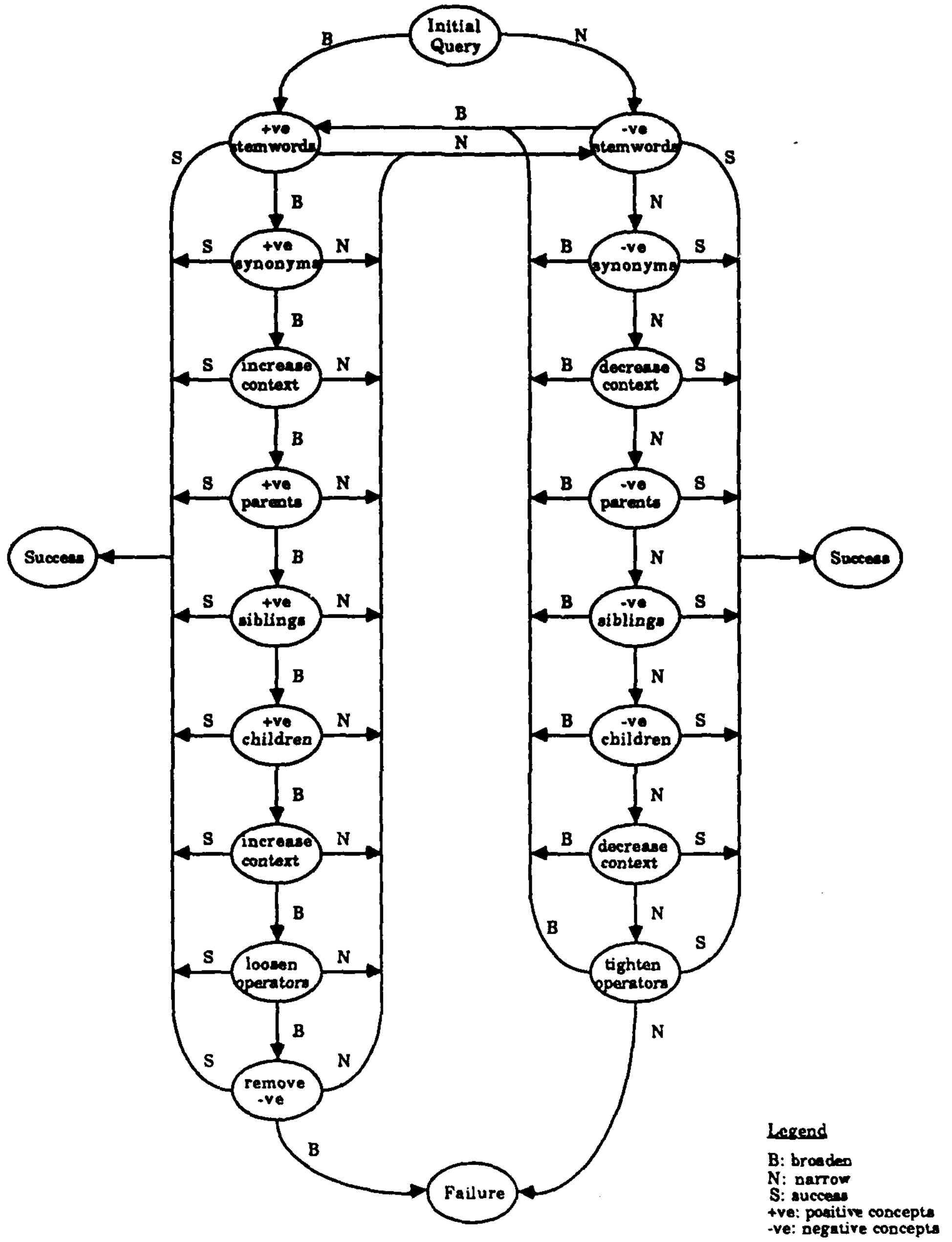

Figure 6.1 Reformulation Techniques

Figure 6.1 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 6.1 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 6.1 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.

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

Although the system is still being refined, this example describes the actions of a working prototype. The expert system currently broadens and narrows under the direction of the user, rather than iterating towards a pre-specified target number. Since our current textbase [16] 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.

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. The ranking algorithm has not been implemented yet, but basically those passages retrieved earlier in the cycle more closely match the user's initial query, and thus would tend to rank higher than passages identified later. The number of occurrences of the search terms also factor affects the ranking of a passage.

7 Conclusion

7.1 Current Status

The text retrieval software and textbase are complete; roughly half of the 936 technical stemgroups have been placed in the hierarchical thesaurus; and the first version of the strategic rule base for the expert system has been implemented. The present expert system front-end presently contains 77 production rules which access 5500 lines of 'C' code. MICROARRAS itself consists of approximately 33,000 lines of commented 'C' code.

We are currently adding production rules used by the expert system to rank the retrieved passages and broaden or narrow the query automatically based on the target number rather than on user responses after each iteration. We expect to complete the working prototype shortly and to run experiments to evaluate the system's effectiveness and efficiency during the fall.

7.2 Future Work

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.

8 Acknowledgements

This work was supported in part by ONR contract No. N00014-86-K-0680. We would like to thank Dr. Frederick P. Brooks, Jr. for the use of his manuscript as the experimental textbase.

9 Bibliography

1. Menkes, Z. CONTEXT: Natural Language Full-Text Retrieval System. Proceedings of RIAO 88: 478–490; 1988.
2. Harman, D.; Benson, D.; Fitzpatrick, L.; Huntzinger, C. G. IRX: An Information Retrieval System for Experimentation and User Applications. Proceedings of RIAO 88: 840–848; 1988.

3. Marcus, R. S. An Automated Assistant for Information Retrieval. Proceedings of the 44th ASIS Annual Meeting, 18: 270–273; 1981.

4. Meadow, C. T.; Hewitt, T. T.; Aversa, E. S. A Computer Intermediary for Interactive Database Searching. I. Design. Journal of the ASIS, 33(4): 325–332; November 1982.

5. Prasse, M. J.; Dillon, M.; Gordon, M. J.; Mortland, B.; Repka, A. F-TAS: A Full-Text Access System. National Online Meeting; 1988 May 10–12; New York City, USA..

6. Euzenat, B.; Normier, B.; Ogonowski, A.; Zarri, G. P. SAPHIR + RESEDA: A New Approach to Intelligent Data Base Access. Proceedings of the 9th IJCAI: 855–857; 1985.

7. Chiaramella, Y.; Defude, B. A Prototype of and Intelligent System for Information Retrieval: IOTA. Information Processing & Management, 23(4): 285–303; 1987.

8. Rau, L. F. Conceptual Information Extraction and Retrieval from Natural Language Input. Proceedings of RIAO 88: 424–437; 1988.

9. Sparck Jones, K. Intelligent Retrieval. In: Jones, K. P., editor. Informatics 7: Intelligent Information Retrieval. 1983; 136–143.

10. Croft, W. B.; Lewis, D. D. Proceedings of the Tenth Annual International ACMSIGIR Conference on Research & Development in Information Retrieval: 26–32; 1987.

11. Pollitt, A. S. CANSEARCH: An Expert Systems Approach to Document Retrieval. Information Processing & Management, 23(2): 119–136; 1987.

12. Fox, E, A.; Weaver, M. T.; Chen, Q. F.; France, R. K. Implementing a Distributed Expert-Based Information Retrieval System. Proceedings of RIAO 88: 708–726; 1988.

13. Tong, R. M.; Applebaum, L. A.; Askmann, V. N.; Cunningham, J. F. Conceptual Information Retrieval Using RUBRIC. Proceedings of the Tenth Annual International ACMSIGIR Conference on Research & Development in Information Retrieval: 247–253; 1987.

14. Vickery, A.; Brooks, H. M. PLEXUS – The Expert System for Referral. Information Processing & Management, 23(2): 99-117; 1987.

15. Teskey, N. Extensions to the Advanced Interface Management Project. OCLC Research Review, July: 1–3; 1987.

16. Brooks, F. P.; Blaauw G. A. Computer Architecture, Volume 1 – Design Decisions. Draft; University of North Carolina, Chapel Hill; Spring 1987.

17. Smith, J. B.; Weiss, S. F.; Ferguson, G. J. MICROARRAS: An Advanced Full-Text Retrieval and Analysis System. Proceedings of the Tenth Annual International ACMSIGIR Conference on Research & Development in Information Retrieval: 187–195; 1987.

18. O'Connor, J. Retrieval of Answer-Sentences and Answer-Figures from Papers by Text Searching. Information Processing & Management, 11: 155-164; 1975.