1994

Knowledge based text indexing and retrieval utilizing case based reasoning

Alan Mick

Rochester Institute of Technology
Computer Science Department

Knowledge Based Text Indexing and Retrieval Utilizing Case Based Reasoning

By Alan A. Mick

A Thesis. Submitted To The Faculty Of The Computer Science Department In Partial Fulfillment Of The Requirements For The Degree Of Master Of Science In Computer Science

May 1, 1994

Approved By:

Professor Feredoun Kazemian

Professor Walter A. Wolf

Professor Peter G. Anderson

R·I·T

Rochester Institute of Technology
Sonja Wendt
Wallace Library
Post Office Box 9887
Rochester, New York 14623-0887
716-475-2562 Fax 716-475-6490

SAMPLE statements to reproduce an RIT thesis:

PERMISSION GRANTED
Title of thesis Knowledge Based Text Indexing and Retrieval Utilizing Case Based Reasoning

I _____ hereby **grant permission** to the Wallace Memorial Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Date: 5/1/94 **Signature of Author:** _____

PERMISSION FROM AUTHOR REQUIRED
Title of thesis _____

I _____ **prefer to be contacted** each time a request for reproduction is made. I can be reached at the following address:

_____
_____
_____

PHONE: _____

Date: _____ **Signature of Author:** _____

PERMISSION DENIED
Title of thesis _____

I _____ hereby **deny** permission to the Wallace Memorial Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part.

Date: _____ **Signature of Author:** _____

Acknowledgments

I would like to acknowledge the support received from my RIT thesis committee, especially Feredoun Kazemian and Walter Wolf.

I would also like to acknowledge the kind support received from the Inspec bibliographic service which provided bibliographic data essential to the completion of the project, and especially Jim Ashling, Director of the US Inspec office, who arranged it for me.

I dedicate this thesis to Rosemary Grace Mick, né Zimmermann, my mother, whose last wish was that it be finished.

Key Words and Phrases

- Information Search and Retrieval
- Knowledge Based Information Systems
- Case Based Reasoning

Computing Review Subject Codes

Using the computing classification codes established by the ACM, the primary classification is H.3.3 - Information Search and Retrieval, with a secondary code of H.3.1.

Abstract

Information retrieval systems for documents normally rely on the use of keywords that describe the text in some fashion or another, or are contained in the text itself, for indexing and searching. These keywords may be associated with standard boolean operators, where presence or absence in the text or text description is used as the truth value, or other operators indicating their proximity to one another in the text.

Another emerging approach is the use of content or knowledge based indexing and retrieval. In this approach the text is not represented or treated as a collection keywords, rather its meaning or semantic content is abstracted and the meaning is used to search for the text desired.

This approach may have several advantages over the standard keyword approach. Both precision and recall of the search may be improved, increasing the likelihood that relevant texts will be found while decreasing the probability of finding irrelevant ones. The knowledge based approach may also allow more sophisticated query techniques, for instance queries based on the purpose for which the text will be used.

This thesis will explore the possibility and usefulness of applying case based reasoning to the problem of text search and retrieval. An easy-to-use expert system for information retrieval that utilizes case-based reasoning to improve, over time, its capability to find those items that are relevant and useful, and only those items that are relevant and useful will be implemented. It will support formulation of a search in an intuitive manner that avoids complicated command syntax and occult operators. It will present retrieved documents to the user in a logical, useful way and will allow the user to easily refine his search criteria based on a selection of documents from his original results that he has judged to be good examples of what he is searching for.

Table of Contents

1.0 Introduction and Background

In contrast to typical database systems, information retrieval systems deal with the storage and retrieval of relatively large units of unstructured information. For instance, an information retrieval system may index and store journal articles that consist of thousands of words of text on a variety of topics, written in a variety of styles, and even in different languages. The internal structure of the articles may be generally similar, but variance would be quite common, and it would be hard to define in a rigorous enough way to make simple use of during processing. A retrieval item in a typical database system, however, might consist of no more than five information fields, would have a very restrictive content (such as name, social security number or address) and would have a well defined relationship to all other information in the database.

Keyword Retrieval

Information systems for document storage and retrieval normally rely on the use of keywords that describe the text in some fashion or another, or are contained in the text itself, for indexing and searching. In searching for information, these keywords may be used in conjunction with standard boolean operators, where presence or absence in the text or text description is used as the truth value, or with other operators indicating their order and proximity to one another in the text. Keywords most often describe a document's content, but may be used to describe other attributes such as its title, author, and publisher, or even what type of document it is and in what way it treats its subject.

A keyword query can be quite simple, or very complex. The simplest is a listing of keywords which select a document if all test true. For instance, the query "`Information Retrieval, Natural Language, Case Based Reasoning`" would retrieve any document that pertained to information retrieval utilizing natural language processing and case based reasoning, but would not retrieve a document that considered the usefulness of natural language processing in an information retrieval system without reference to case based reasoning.

A more complex search allows the addition of boolean operators to expand the differentiation power of a query. For instance, the query "`Information Retrieval AND (Natural Language OR Case Based Reasoning)`" would retrieve all documents that the previous example would, but would also retrieve any documents about information retrieval that considered only natural language processing or only case based reasoning, without requiring both topics to be covered.

If the keywords do not just describe the content of the document but are actually words contained in the document instead, then proximity operators may be used in addition to boolean operators. The Computer Select system allows operators that select

- Two words appearing in the same sentence or paragraph
- Two words appearing within a specified number of words within the same sentence or paragraph

These operators are used to search for documents that discuss the relationship of two topics. They may also be used to construct "complex keywords" that match phrases containing superfluous words or various orderings of significant words.

In addition to keywords that describe the content of a document, searches can be conducted on other attributes of a document. Frequently the title is treated as separate from the contents and searches may be used that operate only on it. Other common attributes used in searching are the document type and treatment of the subject matter. For instance, the INSPEC system allows a search on subject treatment that includes keywords such as "experimental," "practical," and "theoretical/mathematical."

Knowledge Based Retrieval

Another emerging approach is the use of content or knowledge based indexing and retrieval. In this approach the text is not represented or treated as a collection of keywords, rather its meaning or semantic content is abstracted and the meaning is used to search for the text desired.

This approach has several advantages over the standard keyword approach. Both precision and recall of the search may be improved, increasing the likelihood that relevant texts will be found while decreasing the probability of retrieving irrelevant texts. The knowledge based approach also allows for more sophisticated query techniques, such as queries based on an example or sample documents. It can also be used to link the text search and retrieval process into more sophisticated expert systems, such as one designed to help formulate litigation strategies in the domain of law, where text search and retrieval may be used to discover precedence, a key element in litigation strategy.

A key component of any knowledge based system is the data structures and processing techniques used to represent knowledge and to reason based on it. The major varieties of knowledge representation and reasoning techniques now being employed in expert systems are the rule-based approach, pattern recognition, neural networks and case-based reasoning.

A rule-based system stores rules that allow it to recognize or formulate a good plan, strategy or solution to a problem based on an input specification. The initial specification is used to match against any rules that are applicable. When a rule is applied (or "fired" in the terminology of rule-based systems), the state of the system is altered such that other rules may now be applicable, and they then are fired in turn. This process of "chaining" continues until a solution is arrived at. The system may also ask for further information based on its inferences in order to clarify some aspect of the problem and allow it to continue its inference process.

Pattern recognition is a technique used to classify complex instances into useful categories. In many cases, classification alone can determine a plan or course of action, but classification itself may be a complex problem. For instance, once a patient's disease has been identified, the treatment may be well known and straight forward, but identifying from a variety of symptoms exactly what the disease is may be quite difficult. Pattern recognition uses a large number of examples and counter examples and analyses them mathematically to create a classification function. The most popular analyses techniques are linear and quadratic discriminants and Bayes classifiers.Once a formula has been developed it may be applied in a system that processes the inputs in a straight forward manner and chooses a solution based on the resulting classification.

Neural networks rely on data structures that mathematically model the information processing features of biological nervous systems, such as the human brain [26]. The network defines connections between "neurons", which, when suffi ciently stimulated, "fire," sending a message to those neurons to which it is connected. Connections can be excitory or inhibitory, and are added and subtracted to form a weighted sum which must exceed a threshold value for the neuron to fire.The inputs to the network constitute the initial stimulus, and the outputs are the outputs of some subset of the network's nodes. The weights at each connection and the threshold values can be altered to affect the behavior of the network, and by systematically providing the network with inputs and adjusting the weights to achieve the desired output the network is "trained" to respond to a variety of inputs with the appropriate output.

Although each of these techniques has its own strengths, each also has some weaknesses. Some of the weaknesses of these techniques have been explained by Ralph Barletta in his introduction to case-based reasoning [2]. The rule based approach is time consuming and labor intensive to use because the rules that govern decisions in a domain must be discovered through an interview process with an expert. Experts generally do not consciously think in a "rule like" manner outside of the simpler problems they encounter. They thus have a difficult time explaining the "rules of the game" and the process of defining them may be as much a process of discovery for the expert as it is for the knowledge engineer. Rule-based systems also are not easy to maintain. Since rules are inherently dependent on a whole complex of other rules to work properly, they cannot be added piecemeal or ad hoc as needed. An understanding of the entire rule set is needed in order to effectively keep it up to date.

Pattern recognition techniques that use mathematical techniques such as linear and quadratic discriminants can only be applied to those problems that are open to numerical description. Many domains have important features that are non-numeric, such as medicine, where a variety of subjective descriptive information must be used to make a final judgement. Bayes classifiers require a complete probability estimate for each dependent variable in the example set, which is difficult to do in most real-word domains.

Neural networks require the feature of a problem to be defined in terms of a vector of either Boolean or numeric values. As in the case of pattern matching, this limits their applications to problems that can be defined in this way. Many problems are defined through complex internal relationships and the neural network is best suited to a simple list of features. Neural networks also require a lot of computational resources to arrive at an adequate network configuration. In complex cases, this can mean using weeks of CPU time to train a single network. In addition, no methodology exists for the process of defining a network so that it is at best an art and at worst a trial-and-error process. Since it takes so much computer time to perform the trials, the errors can be excessively expensive.

A knowledge representation that currently is generating interest is the case-based reasoning technique. Case based reasoning represents knowledge as cases or examples, and reasoning is based on the recall of cases similar to the current problem. While all techniques

have their own strengths and weaknesses, case-based reasoning has many advantages over the other techniques.

Case-Based Reasoning

When an expert makes a recommendation based on the facts of a particular case, he rarely reasons from "first principles" to solve the problems. More often, he recalls past instances where similar conditions prevailed and either uses the solution that worked then or modifies it to fit the new circumstances. When he does arrive at a particularly good solution to a unique problem, or when he finds a new way to approach an old one, it is remembered and serves as the bases for solving future problems. This is the basic model for case-based reasoning.

A case-based expert system consists of a store of past problems and solutions and a means to look for those that are similar to the problem at hand. Given a sufficiently diverse store, there is a probability that an exact match can be found and the problem can be solved simply by retrieving the stored solution.

When an exact match is not available, a solution to a similar problem is retrieved and then must be used as the bases for deriving a new solution. This is known as case adaptation. While the reasoning process used to derive a new solution from the old may be similar to that used in other techniques, it has a starting point much further along in the process than does the "blank slate" approach that would generate solutions from "first principles." The adapted case is then stored in memory for future use, either directly when the same problem arises, or as a basis for further adaptation when a new problems arises.

In designing a case-based system, several design considerations must be taken into account. These include case representation, case indexing, case storage and retrieval, case adaptation and the system's overall learning and generalization strategies [2].

Case Representation: A case representation is the data structure that stores the information about each case in the system. Simple structures would consist of a list of features from the description of the problem the system solves and the solution that fits each case. An example, from an actual system [43], is the description of a part and the associated manufacturing steps along with their costs. Such a system can be used to bid on the manufacture of new parts by finding similar parts that have been bid in the past and using the historical data to determine what the new part will cost to manufacture. More complicated structures might consist of a related set of subcases, each corresponding to some part of the complete problem solving task.

More important that just the data structure itself, however, is the exact features that should be used in characterizing the problem. This is one of the most important design tasks of the system, since the feature list will determine in exactly what terms the problem will be stated to the system. Discovering and validating the relevant features of the domain is the chief task of the knowledge engineer. In the case-based system, the knowledge engineer interviews the expert to understand and define the domain's terminology and to gather cases which can then be used in the system.

Case Indexing: Case indexing provides the ability to retrieve the appropriate cases from case memory quickly and efficiently given the problem description provided by the user. There are three primary techniques in general use: the nearest neighbor, inductive, and knowledge-guided approaches.

The nearest neighbor approach uses a weighted sum of features in the input case that match, to some degree, the values of features in a stored case to determine if that case should be retrieved. For each feature in the case, the value in the input case much match the value in the stored case to some degree, and the degree of the match may be used in the overall judgement of a case's applicability to a problem. The required match may be perfect, or their may be some degree of tolerance for less than exact matches. For instance, in one case-based system for bid preparation [43], the shape of a part is one feature used. The shape could have one of 10 possible values and for a match to occur on this feature the values have to match exactly. But in the same system a match occurs on the material of a part if the input material's name is a substring of the stored case's material. With numerical values, ranges representing degrees of similarity may be established.

In addition to the degree of match between features, some features may be more important than others. Thus features themselves may have a weight and this weight is applied to the degree of match in determining the overall score. For instance, in the bid example, the salient feature might be part shape. If this were so, the shape feature would have a large weighting factor to bias the selection process towards cases that have a match on it. However, the significance of a particular feature in a particular case might have a complex relation to all other features in the case and their values. In order to account for such complexity, rather than giving each feature a uniform weight in every case, an individual weight might be provided in each case.

The difficulty of utilizing nearest neighbor schemes lies in understanding the problem domain to the degree necessary to create complex similarity and weighting schemes. If there are enough good examples which relate to a set of well defined outcomes or solutions, induction may be used to determine those features which best discriminate the cases and the indexing scheme may be organized around them. Once the inductive analyses has been accomplished, cases may be organized in a hierarchical fashion that allows retrieval on the order of the log of the number of cases stored. Not only does induction allow the selection of those features which objectively best describe the domain, but also allows retrieval in less than the linear time required to calculate a weighted sum for each case stored.

In knowledge-based indexing, existing knowledge is used to determine which features, for each individual case, are the important ones and to match those against the input case. This is the best approach when such knowledge exists and can be codified. However, often such knowledge is hard to obtain and represent for a wide range of inputs. Thus this method is often used as a supplement to the other techniques, helping to make a final determination after an initial selection has been made.

Storage and Retrieval: Once case structure and indexing have been determined case storage and retrieval techniques need to be established. The cases must be organized into an

efficient structure and access methods must be supplied. Storage retrieval can be purely associative, with very minimal to no relational information between cases and a search method that individually considers each one; it may be highly organized into a strict hierarchical structure that provides access to a particular case with minimal searching; or some intermediary degree of structure between these two extremes may be appropriate.

The nearest neighbor indexing scheme lends itself well to purely associative storage and retrieval, whereas the inductive technique provides the information needed to create an efficient hierarchical structure. Discrimination nets can provide an intermediary level of organization between the two.

Adaptation: After a case that nearly matches the problem statement and its related solution has been retrieved, the solution needs to be adapted to fit the input problem specifically. There are several general approaches to case adaptation, but the process is more domain specific than is storage and retrieval.

One approach is to use domain specific rule-based reasoning to work from the retrieved cases to a new solution. The normal techniques of rule-based expert systems and domain modeling may be used, but in a case-based context, two differences result. First, the reasoning process does not start from scratch, but proceeds from one or more existing solutions. Thus the rule structure may be simpler and the process faster. Second, the new solution may be stored so that future adaptation can build upon it, increasing the knowledge of the system and its effectiveness.

Another approach is to combine parts of the various cases that exist to achieve a new solution. This may work best where a complex case structure exists that has subcases with well defined relations between them. An expert system for designing electronic circuits might be able to combine subsystems from several past designs for a new overall purpose.

The problem of case adaptation is not yet to the point where several different generic approaches exist for the system builder to choose from. It is generally approached in an ad hoc fashion, and in some systems it is not used at all. The mere retrieval of several past cases for the user to start from has enough value to justify construction of many systems. For instance, the SQUAD system, built by Hiroaki Kitano for NEC Corporation [23], is designed to index, store and retrieve instances of software defects and their cause, solution and prevention for the purpose of organizational *sharing of experience*, as opposed to an expert system which actually supplies solutions to a problem.

Learning and Generalization: Once a new problem has been solved the potential exists for expanding the knowledge of the system and increasing its potential and efficiency in solving other unique problems. How does the system incorporate new case knowledge in order to expand its knowledge base?

The simplest approach is to add the new case to the case memory by indexing it in the way that the original cases were. With an associative search strategy utilizing nearest neighbor, this is straight forward. However, as the case memory grows opportunities exist to apply more sophisticated techniques.

For instance, the new cases may be analyzed and the information used to improve the efficiency and reliability of the index and retrieval process. Inductive and explanation-based indexes could improve their feature analysis over time and restructure memory to take advantage of the knowledge gained. Another possibility is to perform case generalization that would develop prototypical cases from many existing cases. When a new case is sufficiently the same as the prototype, it is stored with it; but when a new type of case is encountered, the potential exists to create a new prototype and perhaps organize existing cases that did not quite fit the existing prototypes under it. Thus memory could undergo a radical reorganization that might add a significant quantum to the system's effectiveness, as opposed to a simple gradual increase with diminishing returns.

Case based reasoning has several strong advantages over other expert system techniques. It structures the knowledge engineering processes in a way that fits the expert's own approach to problem solving, making it easier to establish a system. It is easier to maintain a system by adding new cases than it is by adding new rules. Case-based techniques are better able to explain the solutions they derive than are the rule based and other approaches.

Case-based techniques work best where there already exists a large store of cases to draw upon. In some situations they do not exist, but must be created through interviewing and observing the domain expert. This can be a complicated and lengthily task. However, in some instances a ready made store of knowledge exists that can be exploited, such as in the case of a biding system that made extensive use of accounting and contract information that could be readily computerized.

1.1 Problem Statement

Typical keyword search systems for information retrieval have several limitations and draw backs that may be improved through the use of case-based reasoning. The major limitation is in retrieval effectiveness as measured by the performance factors of *recall* and *precision.*

When a search is made on an information system's database, the user wishes to obtain all relevant documents available, and does not want extraneous, irrelevant documents to be retrieved. In response to a query for information on a specific topic, the typical system will not find all of the relevant documents in the collection, and will inevitably retrieve some that do not pertain to the topic. Given the subset of documents in a collection that actually are relevant to the query, the proportion of this set retrieved by the query is the system's recall and the proportion of those retrieved that are relevant is the system's precision.

In Blair and Maron's study of the STAIRS retrieval system, word based retrieval systems were found to have poor performance as measured by the recall and precision metrics [5]. Recall was typically only at the 20 percent level, when used by lawyers in their area of expertise [6]. Users were often unaware of this poor performance. The lawyers in this study believed that the recall rate was around 75 percent. Their conclusion was that this resulted from the false assumption that "it is a simple matter

for users to foresee the exact words and phrases that will be used in those documents they will find useful, and *only* in those documents."

There are two linguistic characteristics of words that contribute to this problem: polysemy and synonymy. Polysemy is the trait of one word having multiple meanings, such as the word "stringer," which in the context of newspaper reporting refers to an employee, while in carpentry it refers to the support for a stairway. Synonymy is the trait of having multiple words or phrases to express the same concept or meaning, such as the words "canine" and "dog." Polysemy tends to degrade precision because the words that may correctly retrieve a relevant document will also retrieve those that are not. Synonymy tends to degrade recall because the use of a word that correctly refers to a concept in the query may not be the word used by the author in the sought after text.

In addition to the impotence of word based information retrieval, another problem with such systems is their user interface. The user must either accept the straight forward listing of words, each of which must be within the document, or must learn how to create parenthesized expressions using the system's logical and positional operators.

Due to the need to deal with polysemy and synonymy, search expressions may become very long and complex. In order to reduce the effects of synonymy, many terms relating to the same concept are strung together with the OR operator. If multiple topics are being searched for, each concept must be defined in this fashion. If documents with several different combinations of the same topics are desired, these combinations may have to be repeated over within the same expression.

For instance, consider the case where a user is interested in topics A, B and C, and wants any document relevant to A for background information, and documents that are relevant to both A and B as well as documents related to A and C, but is not interested in B and C alone or in combination. The expression he would use would be `A OR (A AND B) OR (A AND C)`, where A, B and C are long disjunctions of terms related to their respective concepts.

In order to reduce the effect of polysemy, the user may conjoin the basic concept terms with those of more general concepts in order to establish the context of the concept being searched for. In order to use the term "stringer" in the sense relevant to carpentry, as opposed to its use in the context of a newspaper organization, one could construct an expression such as `(STRINGER AND CARPENTRY AND NOT NEWSPAPER)`. Such expressions would then have to be strung together with OR's to handle synonymy and factored out where the general terms are repeated just to make the expressions manageable. Of course, there is no guarantee that the more general topic is mentioned in a relevant document, and in such cases the document would be excluded.

The user interfaces of most keyword based systems presents a clumsy search dialog and does not support easy query refinement. The way in which documents of interest are described, and the interaction that must take place to refine the description is typ-

ically inefficient. They typically give only the number of found documents and short descriptions such as titles, usually in some arbitrary, non-problem related order, to the user who is then on his own to refine the search further. The way in which a user generally interacts with such a system has been investigated by Blair [6]. A user will enter an initial keyword and use it to conduct a preliminary search. This will usually result in a large number of document descriptions retrieved, presented in a hodge-podge order. The user then adds terms to his query simply to reduce the number of documents that are retrieved, and repeats this process until the number of documents seems sufficient for his purpose and not unmanageably large. Thus the query refinement process is directed towards quickly reducing the number of documents retrieved, rather than towards the best formulation for finding the needed documents.

1.2 Previous Work

Case based reasoning has formed the basis for several recent information retrieval systems, and there have been a few theoretical discussions of the applicability of this technique to several domains within the information retrieval field.

CreANIMate

Daniel C. Edelson [13] has done work on the utilization of case-based reasoning in teaching systems. His system, called CreANIMate, uses stories to help teach the biological principles of animal morphology to elementary school students. The system engages the student in a dialogue about animals, and is reminded of stories by cues taken from the interaction. The "stories" are actually video clips illustrating the functions of various animal features.

In most knowledge based tutoring systems, the system must know as much about the instructional material as the student is to learn. In this system, however, much of the lesson comes from multimedia presentations the content of which would be very difficult to capture in full. Case-based reasoning is used in order to select appropriate stories for the student without having to have a full understanding of the stories' content. A full description of the contents of a library of video clips would be very large, if at all possible, and would tax the computational resources that are available for a practical implementation. But since the case-based system's knowledge representation is computationally manageable, more "expressive" information, such as video clips, can be presented than is usually found in such systems.

The kinds of reminding that the system supports are based on common pedagogical categories; the system is reminded of examples, of similarities, and of expectation violations. Examples are used to explain some feature that an animal has in terms of the capability it gives the animal and the survival value or purpose of the capability. Similarities are used to present examples of other animals with different features and capabilities that serve the same purpose, in order to lead the student to generalize the lesson learned from one case. Expectation violations are used to challenge the student with more exact knowledge, and are thought to have intrinsic value in themselves for educational purposes.

The case structure that supports this system is composed of an index of video clips. Each clip is related to an animal that appears in it, a feature of the animal illustrated, the capability the feature gives the animal, and the survival value of the capability. In addition, a more general expression of the capability and the survival value of it is also represented. Thus a video clip of a cheetah pursuing prey would be indexed thus:

- Specific Animal: Cheetah
- Specific Feature: Long Legs
- Specific Capability: Run Fast
- Specific Survival Value: Pursue Prey
- Generalized Capability: Move Fast
- Generalized Survival Value: Hunting

During the dialogue, mention of legs, running fast, or pursuit of prey will remind the system of the clip and it can be presented to the student. This is a case of "example reminding". The generalized capability and survival value can then be used to remind the system of similar cases. For instance, another video clip indexed with "Move Fast" and "Hunting" might be the fishing bat that flies in order to pounce on its prey.

In order to support exception remindings another index structure is used. A universal assertion about a kind of animal forms the expectation, and it is related to video clips illustrating an exception to it. For instance, the expectation that all birds fly to flee predators would be represented thus:

- Generalized Animal: Bird
- Capability: Fly
- Survival Value: Flee Predator
- Violation: Capability, Fly
- Clip: Ostrich Running Fast

When any animal clip that is related to birds flying to flee is activated, the system can be reminded of the exception and present it to the student in an appropriate manner.

FERRET

The FERRET system developed by Michael Mauldin at the Center for Machine Translation at Carnigie Mellon University utilizes natural language processing to understand and index texts and a case frame structure as its knowledge base [33]. The system has been used to experimentally measure the increase in recall and precision that might be gained by employing conceptual information retrieval.

The system consists of a text parser, a query parser, and a case frame matcher. The text parser reads the documents to be indexed and creates case frames that abstract their semantic content. The abstracts form the knowledge base for the system. A

query parser accepts a natural language query and constructs a case frame pattern from it. The case frame matcher then searches the text abstracts for relevant texts, which are retrieved.

The system was used to experimentally evaluate the increase in precision and recall that can be expected utilizing conceptual information retrieval over the standard boolean keyword retrieval. 22 sample user queries that had been expressed both in the standard keyword format and in natural language were used for retrieval on the same collection of over a thousand astronomy texts.The results were then compared.

Over the 22 test queries, the 35% of the documents retrieved by the keyword search were judged to be actually relevant (precision), whereas about 46% of the documents retrieved by FERRET were judged useful. The means of deciding a documents usefulness given the goal of the query were not discussed. To estimate recall a search was made by hand of the entire document base for all documents relevant to a random sample of 5 of the 44 queries. The keyword searches found on average 20% of the documents deemed relevant, while the FERRET system was able to find 52% of them.

The FERRET system is not a practical, fully functioning system. It was implemented to test the relative performance of conceptual information retrieval against boolean keyword searching techniques. The query parser of the system is not implemented and was simulated for the purpose of the study by using the text parser. Of 44 original natural language queries, 22 could not be parsed at all and 16 needed to be paraphrased by the author before successful parsing. Only 6 of the original 44 could be parsed as submitted, and of the 44 only 22 were able to be used in the final study. Although the author carefully documents the time required to index documents, no performance figures are given for retrieval, thus the performance a user could expect is unknown.

Although the FERRET system uses case frames as its knowledge base and case matching as its retrieval mechanism, the author did not elaborate on the techniques used to determine similarity between the query case frame patterns and the text abstracts The texts that are indexed are all narrative scripts from a radio show, chosen in part because the narrative structure makes them easy to parse compared to journal articles, which may contain references to other articles, charts, figures, etc. The texts are also fairly short, only about 300 words each. The system requires a domain description based on a survey of the texts to be indexed, and in extending it from the limited domain description adequate to cover radio show scripts to that required for the full science of the astronomy problems would likely be encountered.

The proposed system will differ from FERRET in many ways. It will be a working system with a fully functional user interface. It will base queries on a "prototype" or "example" of the document created by the user in a simple and stright-forward way. The found documents themselves, after being graded by the user for their relevance and usefulness, will be used as examples for further searches. The system will learn, through user feedback, about the concepts of interest to the users and improve its

document index, and thus over time make it easier to retrieve documents as well as improve the system's precision and recall.

Litigation Support Systems: Flexicon

Case based reasoning has been frequently applied to the area of litigation, since reasoning from precedent and past cases is fundamental to the legal process. Although there are many novel knowledge based systems emerging, in most practical, working systems research into past cases is fundamentally treated as a text search and retrieval problem. The legal profession makes use of more information than any other professional group, and lawyers access electronic databases that encompass tens of millions of documents on a daily bases. A text retrieval system in the legal domain, called Flexicon, that incorporates case-based reasoning techniques has been constructed by Daphne Gelbart and J. C. Smith [14]. This system shares some features with the proposed system.

The Flexicon user interface avoids the use of the typical boolean search query and instead uses four input lists under the categories of concepts, cases, statutes and facts. The significance of each entry may be specified as high, medium or low. Legal concepts or phrases are keywords standing for applicable law and the resolutions obtained when the law is applied to the facts. Examples given include "zoning," "duty of care" and "negligence." Case citations list those previous cases that the user already knows to have a bearing on the legal issues in question. Statute citations list those laws that the user judges to have relevance. Facts are keywords that collectively express the relevant factual aspects of a particular case. The example given of a fact list includes the names of the litigants, geographical information such as cities and street names, types of businesses and institutions such as "restaurant," relationships that obtain within the case such as "employee," and legal features that are relevant such as "building permit." Each list is simple in that it does not attempt to express relations between elements.

The system indexes cases automatically according to the four input categories. In order to glean legal concepts and facts of interest from the case text, an intelligent text parser is used that has an ability to recognize complex legal phrases based on approximate word matches and orderings, without, however, doing actual natural language parsing and understanding. The legal concepts and facts are weighted proportional to the frequency within the document and inversely proportional to the number of documents that contain the word, which tends to give lower weights to those words which have less ability to differentiate and define the document's content.

Case citations are converted to a "canonical form" and are weighted according to frequency within the document. In the legal domain case citations are good indicators of document content, since each cited case serves as a succinct expression for a particular legal issue or concept, and, unlike words or phrases, do not have synonyms or homographs. Statute citations are also converted to a canonical form and

are also weighted proportional to their document frequency. They serve much the same usefulness as do the case citations.

When the user enters a query, FLEXICON searches its document profile database and computes a similarity score for each document based on a variation of the cosine formula given by G. S. Salton (discussed in more detail in the section below on the SMART system). Citation cross reference information is used in matching on cases and statutes and a synonym thesaurus is used to match on legal concepts and factual terms. This allows the inclusion of documents that cover the same or similar cases, statutes, concepts and facts regardless of whether the user's query contained the exact terminology in the document.

After concluding a search, the system presents the retrieved cases to the user in order of their similarity score, and indicates the score for each case both numerically and in the form of a bar graph. All documents over a threshold value of similarity are shown, but since they are stack ranked the user does not perceive the number of cases retrieved to be a problem, thus avoiding query refinement based solely on the need to reduce the number of documents to avoid "information overload."

There are a number of difference between FLEXICON and the proposed system. First and foremost, the query specification and indexing techniques used are specific to the legal domain. The proposed system will function in any domain as a general purpose text retrieval system. The FLEXICON system does not learn from the user how better to evaluate and classify documents. The proposed system will provide the user a means of evaluating the relevance of documents and will not only use this information in place of query refinement, but also to learn about the documents' content and thus be able to improve performance for other users over time.

Litigation Support Systems: Purposive Search

One class of case-based reasoning system generally starts with a problem statement that involves the specification of a purpose or other functional description. These are system that are intended to help engineers find and reuse designs. The designs are indexed according to the purpose, or functionality that they implement, as opposed to the implementation specifics. Case matching is used to find designs for devices or components that serve similar functions or purposes to the one the engineer is currently trying to implements. Mital [30] has applied the purposive approach to text retrieval in the legal domain.

In litigation support systems the documents to be indexed and retrieved are all related to a specific case that the litigation team is working on. Documents may consist of contracts, telephone logs, receipts, and other variety of evidence that may be used to support or contend against the claims being made is a specific case. It is not unusual find cases that involve 50,000 or more documents. The ability to find the right document at the right time may make the difference in a case.

Rather than index the document collection on the basis of its contents alone, the indexing scheme is based on the fundamental purposes that the documents are used

for in litigation. A certain type of legal action has certain specific legal and factual issues that are in contention, and the litigation team's purpose is to prove or disprove them. For instance, in a case involving negligent misrepresentation by a financial adviser that causes his client to suffer a loss the broad issues may be defined as

- Whether the client possessed information from other sources
- Whether the client acted according to the adviser's advice
- Whether or not the loss was caused by reasons other than just taking the adviser's advice.

For a particular case, and indeed, for entire classes of cases, these issues may be well defined and may be decomposed into a hierarchy of issues. However, just labeling the documents with the issues and sub-issues is too imprecise and coarse. The users should be able to retrieve those document's that relate to a specific issue in a particular way, and there are other types of properties of documents, other than the issues they relate to, that determine their usefulness in a court of law.

One type of property that is need to be known about documents is the kind of documents they are. But in this case, the classification system is peculiar to the legal field. At the highest level documents may be classified into those of "disputable provenance" and those whose contents may be "assumed known to sender." For instance, a telephone log is of disputable province since the sender (caller) does not necessarily know what is in it, or even that it exists. However the contents of a postal communication is assumed known to the sender. In the case of a postal communication, the type may be further decomposed into those that are registered, those that have a signed receipt, etc.

Another way of classifying the documents in a case is by the situational facts that are covered in them. Once again, there is a specific legal interpretation and classification, typical to each type of case, as to what situational facts are and are not important. In the example we started with, loss due to bad advice, the root facts are the sources of "independent knowledge" and the loss itself. Facts under the topic of independent knowledge may be related to "Third Party Advisors" and each type of such advisor, such as lawyer or accountant. Under the loss, relevant facts are the "loss accrued" and "estimated future loss." In any particular type of litigation the fundamental facts to be proved or disproved are a mater of previous legal precedent, so a system of this type could start with much of the higher level categories already established.

The issues a document is concerned with, the type of document it is, and the facts that it tends to establish are all atomic properties of the document itself. Other types of relational information is stored with the document. These are explanatory links, reference links and relevance functions.

An explanatory link relates an atomic property to another in the document in order to represent the extent of the validity of the interpretation the atomic property represents. The types of relations are "Implied By," "Alternative Interpretation," and

"Excludes." As an example, a document may support the situational fact that the plaintiff "attended seminars on trading futures." But if the indexer thinks that the document can show that he did so not as a passive listener, but intended to get personal advise form the lecturer, an alternative interpretation may be "advised by third parties." The indexer includes the explanatory link to indicate that the atomic property "advised by third parties" is an alternative interpretation of the more directly supported property "attended seminars."

A reference link relates the document containing it to others in the system. The types of relations are "Refers To," "Incorporates," and "Rebuts." If a letter is sent by the plaintiff in reply to an accusatory letter from the defendant, it may contain information that would go towards disproving the contentions in the accusal, and thus would be related by the "Rebuts" link.

A relevance function relates one (and only one) of a document's relevant issues to one or more of the other atomic properties of the document. For instance, the relevant issue "Loss by Extraneous Factors" may be related to the properties "Phone Log," "Accountant," "Buy Put Options," and "Acceptance of Advice." Remember that the atomic properties exist inside of legal domain hierarchies such as "Phone Log" being an instance of a document with disputable providence and "Accountant" being a "Third Party." This gives these basic qualities more significance than they would have as simple phrases or qualities.

While there is no universal theory of relevance of concepts to issues in the legal domain, it is possible given the facts of a case for a lawyer or paralegal to determine that a document is likely to be relevant to a particular issue and for the reason that the document contains references to certain concepts. The reference functions apply only inside a specific document, but it is inevitable that many of the functions from many of the documents will be similar, if not exactly the same. The reference functions from the documents may be organized into a subsumption hierarchy, discriminated first by the issue and then by the other properties in order of importance as indicated by the indexer.

In order to retrieve documents, the user specifies a query that matches a reference function in form, that is, it consists of one issue and a set of other atomic properties. A document is retrieved when at least one reference function in it matches the query. A query can match exactly either simply or through the agency of an explanatory function, and it can match partially.

For a query to match simply, the issue and the atomic properties must either match identically or be closely related in the various hierarchies to which they belong. A query can also match if an explanatory link may be used to alter a reference function to match. For instance, if the issue in the query is "Advised by Third Party" and in the reference function the issue is "Loss by Extraneous Factors," a simple match will not be made. But if a "Alternative Interpretation" explanatory link exists between the two issues, then a match may be made. A partial match exists when an issue

matches but not all other properties match. The system orders results by degree of match for easy perusing.

This Mital's system is very different from the proposed system. In Mital's system a document is retrieved according to its purpose as determined by, and only by, a specific legal context created by a specific legal action. The context and the action so constrain the use and interpretation of the documents, that the purposes that a document may serve actually become an *intrinsic* property of the document, as much as any other attribute we might normally ascribe to one, and thus are able to be used in indexing. In contrast, the purposes that documents serve in the context of a general purpose literature search system are extrinsic to them, and cannot be determined *a priori* by an indexer. Certain attributes of a document might make it statistically more useful to more researchers with a common stated purpose, and this information may be used to present the results in a useful way to the user, but the purpose cannot be attached to the document as an inherent attribute of it. The extrinsic notion of purpose would have to be used in a general purpose litereture search system.

The SMART System

The SMART system is an information retrieval system that pioneered many of the experimental retrieval techniques developed in the last three decades. It contains many innovative features that distinguish it from more conventional systems: fully automatic indexing, subject class indexing; queries based on similarity matching rather than boolean keywords and automatic query improvement based on user feedback [37].

Automatic indexing in the SMART system is accomplished by using words and phrases extracted from the text using fairly simple language processing techniques. The individual words in a text are first parsed and filtered using a stop list of ordinary, less useful words such as "and," "or," and "but." This leaves an ordered list of words that have the ability to help differentiate the document from others.

Each word is then broadened by reduction to a word stem form. The word stem can be thought to contain a word's root semantic content, so this operation has the effect of abstracting the content or significance of the word, as distinguished by its more syntactical characteristics. Thus a sentence such as "People in need of information require effective retrieval services" would be reduced to the list "people inform effect reriev service." The number of occurrences of a single stem can be used to characterize the degree to which a document treats of a specific subject, which becomes the stem's weight in the document.

In order to include phrases, as well as just words, into the index, a combinatorial process is used. From the ordered stem list a distance factor is used to combine all words that occur within a specific proximity into two word phrases. The distance factor is chosen to produce no more than a manageable number of phrases. Phrases in which both words match are eliminated. A phrase is weighted based on the weight of each stem in it, not the frequency of the phrase itself in the document. Using a distance factor of four, the words in the previous example would form the following

phrases: people-inform, inform-effect, inform-retriev, inform-service, effect-retriev, effect-service, and retriev-service. It can be seen how these combinations can approach the effect of using compound key words such as "information retrieval" assigned by a reviewer during a manual indexing process.

The SMART System is also innovative in that it eliminates the use of boolean keyword queries and uses instead a weighted list of terms that is then matched against a documents index using a similarity function. The query and the document is conceived of as a vector in a multidimensional space, and the cosine of the angle between the two is used to characterize their similarity.

A particular document, D_i, is represented by the collection of terms, $T_{i1}, T_{i2}, \ldots, T_{it}$, where T_{ij} is the weight in D_i of term j found in the document during the automatic indexing process. A term that is not in the document is given a weight of zero. Similarly, a query, Q_k, is a vector of terms, $T_{k1}, \ldots T_{kt}$, where T_{kj} is a weight indicating the importance of term j to the content of the documents desired by the user. The retrieval of a stored item then is determined not by the inclusion of all desired terms, but by the degree of similarity between the stored vector and the query. The similarity is measured by the cosine of the angle between the two vectors in the multidimensional space of dimension t, where t is the total number of distinct terms in the system's index. This is given by the equation:

$$SIM(D_i, Q_j) = \frac{\sum_{k=1}^{t} (T_{ik} \cdot T_{jk})}{\sqrt{\sum_{k=1}^{t} (T_{ik})^2 \cdot \sum_{k=1}^{t} (T_{jk})^2}} \qquad (1)$$

The representation of queries and documents in essentially the same way and the use of a similarity function between them makes possible several unique features in an information retrieval system. It is no longer necessary to retrieve all the documents that contain all of the specified terms; rather, retrieval is dependent on the degree of similarity in relation to some threshold value above which a document will be included. This could have the desired effect of eliminating many irrelevant documents that would otherwise be retrieved. Documents that do not have all the desired terms represented, and yet are better matches than some that do, may be included, increasing the likelihood of a more relevant document being included over a less relevant one.

One other important property is that it becomes possible to order the retrieved documents by their relevance, and thus present them to the user in a useful, rather than arbitrary, way. By ranking the documents a large number may be more manageable than if they were presented in a random order and this allows the user to make reasonable use of large retrievals.

Since the document representation and the query representation are essentially the same, it becomes easy to use documents to modify queries in order to improve their retrieval performance, and even to use documents themselves as queries. By utilizing user feedback as to the relevance of retrieved documents, and using the relevant documents to modify the query, query refinement becomes a straight forward and simple process for the user. By rationalizing the refinement process, more effective queries may be arrived at faster than with the boolean keyword approach. In addition to query refinement, a document's vector may be systematically altered based on relevance feedback and the user's modified query. This then makes the system smarter by improving its document memory. This is the technique of dynamic memory that is the core foundation of case-based reasoning.

Another advantage of representing documents in the vector format is that the similarity function may be evaluated between them during indexing and storage processing and similar documents may be hierarchically indexed and grouped physically together on the storage media. This subject class indexing makes more efficient retrieval possible than when documents are stored in a less cohesive manner.

The proposed system will use a vector representation of documents and of queries, as well as the word steaming and phrase building strategies of the SMART system. However, in the SMART system each term in the query or document is assumed to be orthogonal to all other terms in the system. Thus each term represents a different and distinct dimension in the t-dimensional vector space. Each term "pulls away" from all other terms, regardless of whether it is or isn't conceptually related to some subset of them. This assumption is not realistic. In the proposed system, the definition of concepts by the user gives a bases on which to group terms so that terms related to a given concept may "pull together." Rather than building a t-dimensional term space, a c-dimensional concept space is constructed. Using the cosine similarity function this produces a unique effect, as described below under theoretical and conceptual development.

1.3 Theoretical and Conceptual Development

The overall goal of the project is to implement an easy-to-use expert system for information retrieval that utilizes case-based reasoning to improve, over time, the system's capability to find those items that are relevant and useful, and only those items that are relevant and useful. It should support formulation of a search in an intuitive manner that avoids complicated command syntax and occult operators. It will present retrieved documents to the user in a logical, useful way and will allow the user to easily refine his search criteria based on a selection of documents from his original results that he has judged to be good examples of what he is searching for.

Concepts and Document Prototypes

In order to construct a search, the user will be helped by a catalog of previously used topics and concepts. He can use concepts as they exist in the catalog, or he can create new ones, either in whole or by borrowing from the catalog. Structurally, a concept is a named list of words or phrases, the presence of which in a document indicates

that it may be relevant. Rather than using cryptic operators to create a query that resembles a mathematical formula, the user builds document prototypes by associating concepts together into an example or prototype resembeling in abstract form the documents desired.

The concepts in the catalog and in the user's search workspace are arranged in hierarchies to help define them and limit their meaning within a more general concept, area of application, or topic. In the user's search workspace, the concept hierarchies can be constructed on an ad hoc basis, without the imposition of a general all encompassing taxonomy. This is necessary since the same concepts may be categorized and arranged in many different ways, given different interests and purposes.

The user may search the system catalog of concepts based on partially defined concepts and borrow from it. The search will be implemented using case-based similarity matching. Individual words and phrases are reduced to their stems, utilizing the technique that is employed in the SMART system. Concepts are matched on the number of individual word and phrase matches, utilizing the cosine similarity function, also used in the SMART system.

Concepts that are defined for a search or are refined in the course of one are retained in the system and the documents are reevaluated against the new and changed concepts. By allowing users acess to previously defined search concepts in this way, they get the benefit of past experience in defining their search, and they focuses on a good definition of the concepts involved, instead of refining the search in the context of an excessive number of largely irrelevant documents, as in typical information retrieval systems. Thus the intellegence of the system is increased not only by altering the documents' index, but also by storing examples of previous searches and allowing thier reuse and refinement. This is a major inovation over existing systems.

Document Search

After formulating a query the system will search its document catalog for relevant documents. Documents are represented by a frame structure containing slots for several attributes, such as the document's authors, publication, document type, treatment, language, etc. The document's content is represented by a list of words and phrases, each weighted to represent its importance in characterizing the document, and a list of concepts from the system concept catalog scored according to the frequency of its constituent words and phrases in the document. The content list is used to match against the words and phrases from new concepts defined by the user and the concept list is used for concepts already defined in the system. Scoring is based on the cosine measure used in the SMART system, but rather than use each term as an individual dimension in the vector space, terms are aggregated into a concept space based on the user's concept hierarchies.

Individual words and phrases in both the query and from the documents are reduced to their stems, utilizing the technique that is employed in the SMART system, rather than having a match be dependent on the accidents of the syntax of a sentence or the individual expressiveness of an author. By utilizing a word's stem, rather than the

full grammatical form that it takes in the sentence, words tend to be matched on the basis of their root meaning. The stemming algorithm used is a straight forward implementation of that described by Lovins [29], which is cited by Salton in describing the SMART system. It is not an inovation of the current system.

When a concept is included in a query, the entire path in the concept hierarchy to which it belongs is implicitly included. Each concept in the path is matched as part of the search process, and while they influence the results less than a match on the search concept itself, matches on the associated concepts will increase a document's relevance score. Each concept contained in those paths from the root to a leaf and that pass through the included concept will be included in the query with a weight inversely proportional to the square of its distance from the included concepts.

Documents that score above a specific relevance threshold will be scored according to their usefulness and presented to the user in an order that classifies and ranks them in a meaningful and useful way. The user will then be given the chance to evaluate the relevance of the document based on the information that the system has stored about it. This may (in some configurations) include the full text of the document or an abstract of it.

The user is then given the opportunity to add phrases and terms from the high scoreing documents to the query's concepts, and to add concepts to the query from the documents. This is done through a user-friendly interface that analyzes the good documents to allow the user simple and effective control over the query refinement process. Modifications to concepts are saved in the concept catalog. On subsequent searches that include the concept, this will affect the document's over all score, either raising it or lowering it. As the system is utilized, the catalog of concepts grows and is refined, increasing the power of the system.

After refining the query, the user may then run the search again. In the initial search, the query's concepts were given target values based on the average score for the concept accross all the documents in the document catalog. In subsequent searches, however, the system updates the query's concepts' scores based on the documents that the user has judged most relevent. This makes the query a better representation of the documents the user desires than it orginally was in the initial search. A detailed explination and rational for this refinement process is given under "conceptual simularity" below.

Presentation of Results

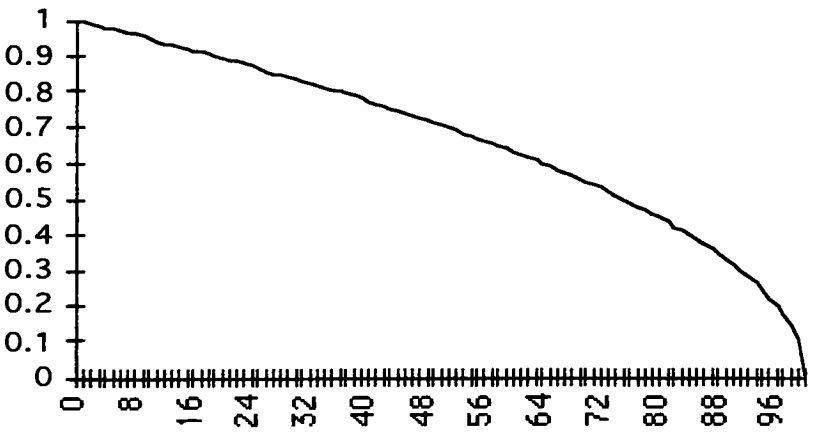
The system will order and arrange retrieved material its relevance. When examining a particular document, the user may grade it according to his judgement of its relevance and usefulness, or he may dismiss it from the results altogether. By ordering and presenting the documents in this way, the user does not perceive the results to be an unmanageable hodgepodge of relevant and irrelevant material. He is less likely to focus on simply reducing the bulk of retrieved material and more likely to pay attention to the information presented about each document in order to judge its relevance. The user is then able to conduct further searches based on his input about the

documents and thus is more likely to find a greater number of relevant and useful documents.

Conceptual similarity

To judge the relevance of a document to a user's query, a variation on the cosine measure employed by the SMART system will be used. This measure treats both the query and a document as vectors in a t-dimensional space, where each possible term has its own dimension. The similarity of a document to the query is inversely proportional to the angle between the two vectors. A document that scores identically to the query vector on each term has an angle of zero and a similarity of one, indicating an exact match. As the angle between them increases, the cosine of the angle decreases, and the score drops toward zero. The cosine has the further property that it decreases more rapidly as the angle increases. Figure 1 shows the similarity score for a document and query as their similarity decreases.

FIGURE 1. Cosine Similarity Measure



This similarity measure, however, makes the assumption that each term used in a document or query vector in a given system is unrelated to all other terms. That is, each term is indeed "another dimension" in the term space, and each dimension is orthogonal to the others, "at right angles" to them. Thus terms that are related to the same root concept or topic of discourse are unrealistically represented.

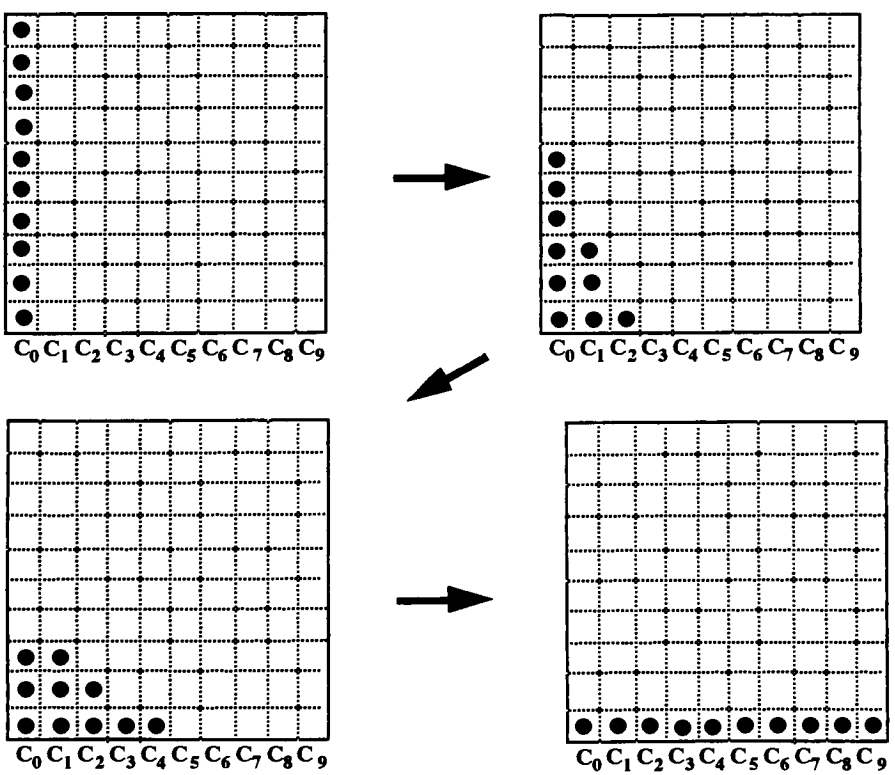
In order to overcome this assumption, the proposed system will use concepts as the bases for similarity matching, where concepts are conceived as a group of terms and phrases that each relate to or indicate that a particular concept or topic of discourse is the subject of a text. Thus when searching for works that relate to case based reasoning, one may search not only for the phrase "case based reasoning," but also for

phrases such as "dynamic memory" and "episodic memory." Each of these phrases indicates the single topic of interest, not (in this context) separate concepts.

In order to "conceptualize" the t-dimensional term space into the c-dimensional concept space, the user's concept catalog of related terms and phrases is used. A query is a vector of concepts, each defined by the terms and phrases related to it in the concept catalog, and a document is a vector of concepts, each of which aggregates the scores for the same set of individual words and phrases.

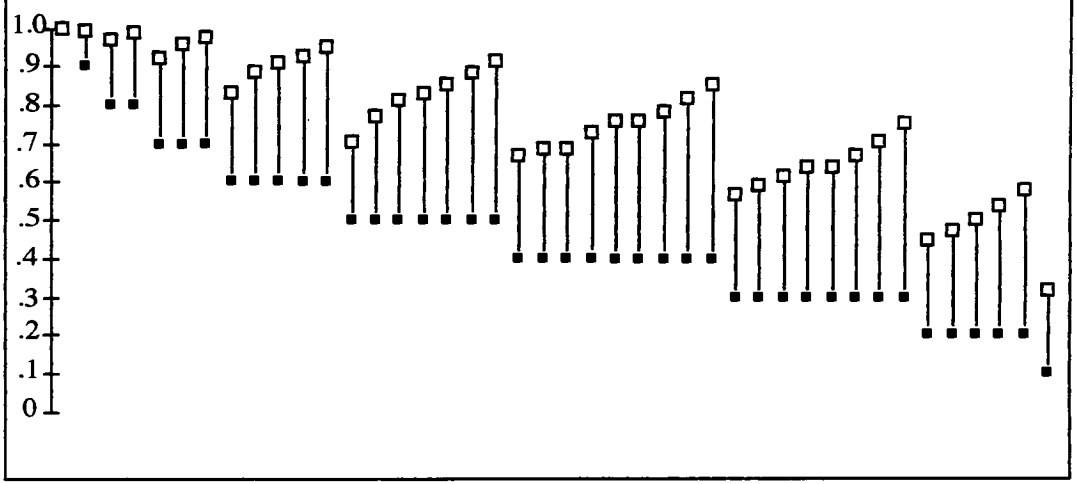
Hypothetically, one would expect the conceptual representation to produce much stronger similarity scores for relevant documents than would the orthogonal term representation. In order to give credence to this expectation a priori, a mathematical test was conducted. A universe of 10 concepts, each represented by 10 unique terms was postulated. A subset of all possible document vectors was chosen which smoothly distributed ten terms through the concepts from a state where all ten terms match a single concept to the state where the ten terms each match one of the ten concepts. Figure 2 illustrates this progression.

FIGURE 2. Progression of Terms Through Concepts



Each of the 42 different document vectors was then matched against a query vector representing a request for a document covering the first topic. The resulting similarity score was compared against the analogous score for the non-conceptual orthogonal term similarity measure. Figure 3 displays the results.

FIGURE 3. Conceptual Vs. Term Similarity Scores



Several interesting properties of the conceptual similarity technique are revealed. The graph in Figure 3 is arranged in what was intuitively thought to be an increasingly dissimilar order of documents. By the criteria of the term vector technique, the sequence was increasingly dissimilar, but the score could not distinguish each and every document. The conceptual vector technique was able to distinguish each document, but a saw-tooth pattern emerged, showing that the intuitively expected ordering was not in accordance with the score. In all cases the conceptual vector scored higher than the corresponding term vector, which was generally in accordance with the hypothesis that the conceptual technique is better at evaluating the similarity when the underlying terms "pull together."

In order to understand the counter intuitive "saw tooth" pattern that resulted, examine the values from one sequence as displayed in Table 1. Note that the intuitive ordering starts with a concept vector that has terms concentrated in two concepts and a few terms in a third. As the ordering progresses the terms diffuse through the vec-

tor to the point where the terms are concentrated in one concept with scattered terms throughout six others.

Table 1: Conceptual Scores for One Sequence

Document Vector	"Intuitive" Order	Term Score	Conceptual Score
4,4,2,0,0,0,0,0,0,0	20	400	667
4,4,1,1,0,0,0,0,0,0	21	400	686
4,3,3,0,0,0,0,0,0,0	22	400	686
4,3,2,1,0,0,0,0,0,0	23	400	730
4,3,1,1,1,0,0,0,0,0	24	400	756
4,2,2,2,0,0,0,0,0,0	25	400	756
4,2,2,1,1,0,0,0,0,0	26	400	784
4,2,1,1,1,1,0,0,0,0	27	400	816
4,1,1,1,1,1,1,0,0,0	28	400	853

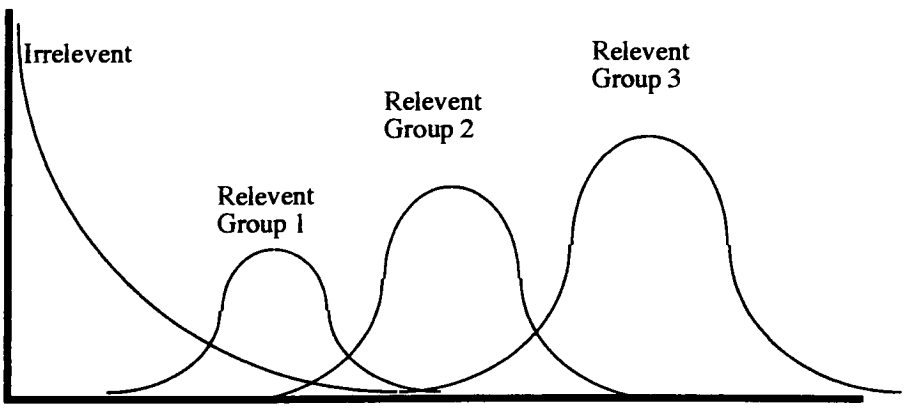
These vectors were matched against the query vector (10,0,0,0,0,0,0,0,0,0), which represents a request for a document that discuses the first topic. The document (4,1,1,1,1,1,1,0,0,0) scores better than (4,4,2,0,0,0,0,0,0,0) because it is more definitely about the topic of interest and less likely to be about any of the other possible concepts, while the other document is likely to be about the topic of interest *and* another topic. The query vector should be interpreted as requesting documents about the first topic and *only* the first topic, since no other topics are marked in it. The term Scoring technique does not differentiate the various documents in this sequence, treating the query in a boolean key word fashion: if the document is about the topic of interest and any other topic, it is selected. The conceptual technique, however, differentiates the documents in the sequence, and scores consistently with the interpretation of the query as an "example document." This indicates that the ordering produced by the conceptual technique is superior to both the intuitive ordering and that produced by the term technique.Since, however, the investigation has been purely a priori using a mathematical model, the hypothesis that the conceptual technique is superior needs empirical verification. One goal of the proposed thesis will be to test this hypothesis experimentally.

For the purpose of the mathematical investigation above, each concept was based on ten terms, and each term was given a weight of one in the document vector to indicate its presence in the document. The query gave each term in the target concept a weight of one, for a total concept weight of ten. This is a simplification of the scoring

procedure that will be used in the proposed system. In order to construct a document vector, the frequency of the term stems from a concept will be used to derive the weight of the concept in the document. This is a fairly straight forward procedure. However, in constructing a query vector, the weight to give a concept is more problematic. If a simple count of terms were to be used (as in the simplified model), concepts that have a few terms would have low scores compared to those that have many. If the expected weight of a concept for a document in which it is relevant were proportional to the number of terms that defined the concept, this would perhaps be adequate. But, of course, there is no reason to expect such a relationship to hold. What then, would be a reasonable weight to put on a concept in a query?

It is expected that for a particular concept, the frequency distribution of weights over an entire data base may appear as in Figure 4. It is expected to have two fundamental components; an inverse exponential component that represents the scores of documents that are not relevant to the concept but happen to contain some of the terms that make it up, and one or more bell curve components that represents the scores of documents that are relevant in differing types of documents. Some documents are more or less principly about a specific concpet. These documents would have a strong score for the document. Other documents may have several different topics of discussion, including the one under consideration. These different document groups would have lesser scores for that same concept, but would still be relevent to the concept, and would be especially relevent if the other concepts were the other concepts the user is searching for.

FIGURE 4. Expected Concept Weight Frequency Diagram



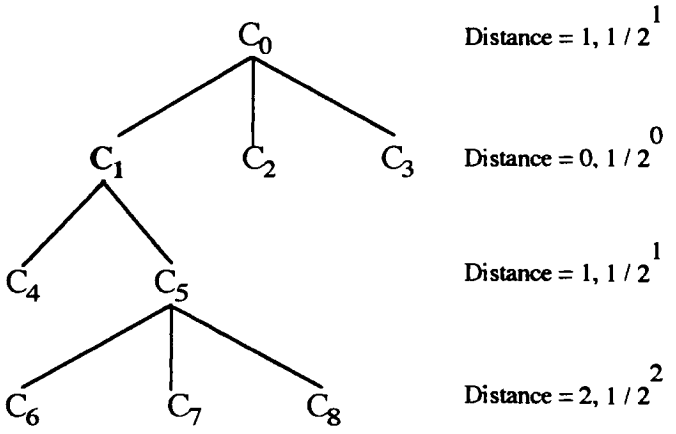
The best value to represent a concept in a query would be the mean of the weights of the *relevant* documents in the *relevent* document group. Given only the the distribution for a particular concept, however, it is imposible to predict which group represents the relevent group and thus which concept value in the query will optimize the

selection of relevent documents. This provides the motivation for the query refinement process described above. In the initial query, the average score accross all non-zero scoring documents is used. This gives broad coverage to select a good representation of documents containing the concept. After the initial search the user selects documents that are good examples of those he is looking for. On the subsequent search, the query value for the concept is the average accross those selected documents, which would then more exactly represent the concepts' scores in the *kind* of document the user is looking for.

In constructing a query vector, not only the concepts that the user explicitly includes in a document prototype, but also those concepts to which it is related in the concept hierarchy will be included. This is to help counteract the effect of synonymity, in which a particular term or phrase may relate to several different concepts in several different domains of discourse. By including concepts from the hierarchy which relate to the desired concept, a tendency to exclude documents from domains of discourse other than that of the terminology selected will be established, thus leading to a richer yield of relevant documents. However, simply including those concepts into the vector as if selected explicitly by the user would biase the query in favor of concepts other than the desired focus. In order to include these concepts, but not overwhelm the query in their favor, they are given diminished weight in the query.

A concept hierarchy establishes a tree relationship amongst the concepts in it. A concept selected by the user through inclusion in a document prototype occupies a position relative to the root in one or more paths through the tree. Each concept in a path that includes the selected concept will be included in the vector at a weight biased inversely proportional to the distance from the selected concept, using the formula $1 / 2^{D}$, where D is the distance. Given the concept hierarchy depicted in Figure 5, the

FIGURE 5. Related Concept Weighting



selected concept C_1 will receive its full weight, W_1, when included in the query vector. The related concept C_0, at a distance of 1 from it, will receive a weight of 0.5 * W_0, similarly C4, 0.5 * W_4, and C5, 0.5 * W_5. At a distance of 2, C_6, C_7 and C_8 will be biased by 0.25. Concepts C2 and C3 will be given a score of 0, unless explicitly selected. When a related concept is included in the paths of several selected concepts, it takes its highest value from amongst those it may be assigned; the values are not additive.

1.4 Glossary

- Cased Based Reasoning: An artificial intelligence technique in which knowledge is stored as previous examples or cases. The cases are indexed by attributes of the problem that was previously solved or the purpose that was previously served. The system retrieves those cases which are most similar to the problem at hand and uses them for further reasoning. The case store is usually updated based on the problem at hand, to aid reasoning in the future.
- Conceptual Information Retrieval: The application of knowledge-based techniques from the field of artificial intelligence to the problem of retrieving information.
- Precision: The measure of an information retrieval system's ability to find *only* those documents that are relevant to a given topic or subject as expressed by a query. Given the subset of documents in a collection that actually are relevant to the query, the proportion of those retrieved that are in this set is the system's precision.See *recall* and *relevance*.
- Recall: The measure of an information retrieval system's ability to find those documents that are relevant to a given topic or subject as expressed by a query. Given the subset of documents in a collection that actually are relevant to the query, the proportion of this set retrieved by the system is its recall. See *precision* and *relevance*.
- Relevance: The relevancy of a document to a user's interests is an expert human judgement based on an understanding of the user's interests, the field or domain of the interest and the content of the document. It is based on the fit between subject or content of the document and the user's interest, not necessarily the *usefulness* of the document to the user, which may take into account factors such as its treatment of the subject, its availability and the language in which it is written.
- Usefulness: The usefulness of a document to a user is his judgement as to the value he will receive from it, as opposed to the mere relevance of the document to the topic he is researching. While relevancy is necessary for usefulness, other factors, such as the document's treatment of the subject, its availability in local libraries, and the language in which it is written, may make it more or less useful for a particular person and purpose than another document equally relevant.

2.0 Software Project Description

The overall goal of the project is to implement an easy-to-use expert system for information retrieval that utilizes case-based reasoning to improve, over time, the system's capability to find those items that are relevant and useful, and only those items that are relevant and useful. It should support formulation of a search in an intuitive manner, that avoids complicated command syntax and occult operators. It should present retrieved documents to the user in a logical, useful way and should allow the user to easily refine his search criteria based on a selection of documents from his original results that he judges to be good examples of what he is searching for.

One of the key elements of the system is the means for entering search criteria. The entry should be straight forward and uncomplicated for the user, and should support the case reasoning techniques that lie at the heart of the system. Information retrieval systems are in general either too simple in this area to support sophisticated retrieval, or they are overly complicated. In the proposed system the process is divided into two phases, the formulation of concepts and their use to form document prototypes.

Concept Formation

At the core of the system's query formulation is the definition of a concept or topic. The concept is then used as the building block for a document prototype. Concepts are defined in two ways; first, by specifying words and phrases that relate to the concept, and second, by relating concepts hierarchically by topic and subtopics. In doing so, graphic user interface techniques are employed in order to make the process simple and straight forward.

Query refinement not only updates the query that the user is currently working on, but also the concept catalog. Thus the system improves, over time, its catalog of concepts that users may draw on to create queries. As more concepts are defined and as the terms and phrases associated with them are refined, the system becomes easier to use and is able to increase its ability to find relevant documents and only relevant documents. The ability to store and reuse query elements in subsequent searches is a unique to this system and represents a major innovation.

To formulate a concept, the user will create a list of words and phrases related to it. This list is roughly analogous to a parenthesized list of words and phrases (perhaps using positional operators) utilizing the OR conjunction in a traditional information retrieval system. However, in our system the list represents a query case to be matched against cases representing documents. The case matching algorithm, explained above, inherently utilizes a notion of similarity, so it is not necessary to supply positional operators or wild card characters to avoid selecting only documents that exactly match. The system will employ such strategies of its own accord. Figure 6 illustrates such a list. The list may be collapsed and represented by its most succinct name on a single line.

Several concepts may be related together hierarchically in an outline format, representing more specific concepts falling under the rubric of more general concepts. Figure 7 shows the appearance of such a hierarchy. In a traditional information retrieval system, this could be represented by an AND conjunction between the more general concept and the more

specific, which would help to limit and define the specific concept less ambiguously. However, in our system the more general concept does not need to be represented explicitly in the search formulation, it exists as part of the concept definition and provides a helpful

FIGURE 6. CONCEPT REPRESENTATION, OPEN AND CLOSED

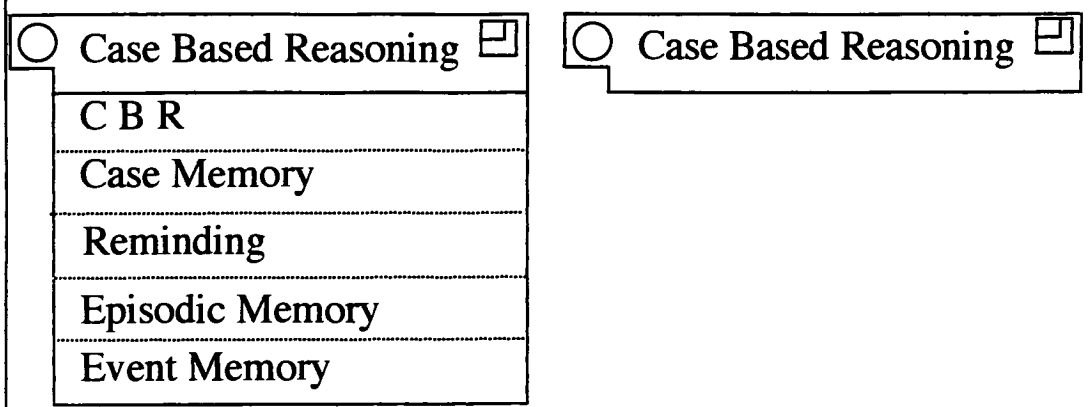
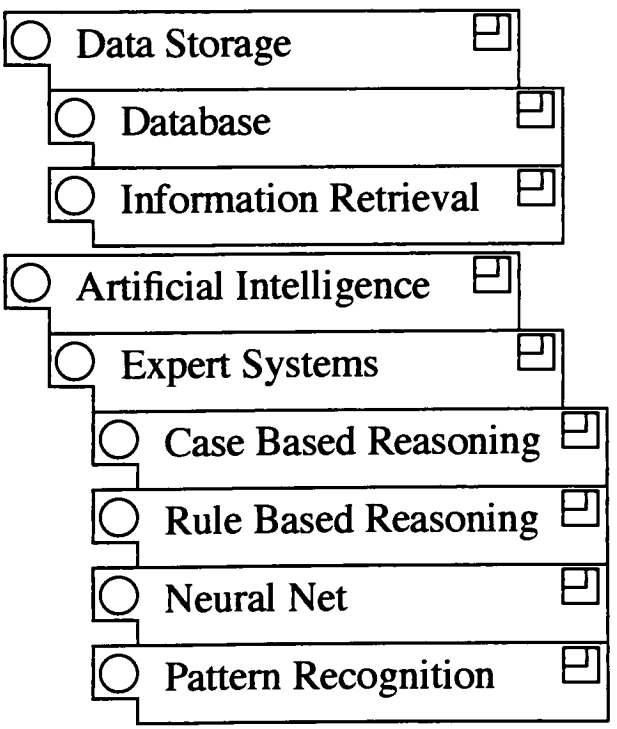


FIGURE 7. REPRESENTATION OF CONCEPT HIERARCHIES



hint to the case matching mechanism, not an absolute directive that the words and phases of the more general concept *must* be present. Concept hierarchies may exist separate from one another, and do not have to be combined into one over-arching structure that defines a complete universal taxonomy of the search topics.

Document Prototypes

After defining, finding and refining the concepts that interest him, the user is ready to formulate a search. In the typical information retrieval system, concept and search formulation are not generally separated; the query defines the concepts (keywords and phrases) as well as relates them together, utilizing the AND and OR conjunctions. In the proposed system concepts are formulated as building blocks and then used to create "document prototypes" which describe the documents desired by the user. One or more document prototypes are then used as a query to the system.

The system represents a document prototype as a document icon that contains concepts the user has selected for inclusion in it. By placing a concept onto a document icon, the user is telling the system that he desires it to find documents that are relevant to that concept. When several concepts are listed within a single document prototype, the documents that are most likely to satisfy it will be those that are relevant to ALL those concepts. This is analogous to an AND conjunction between keywords in a traditional information retrieval system, but since case matching is used, all concepts listed need not be present for a match to occur. If a document that is missing a particular concept nonetheless is a better match than another, it will be preferred.

A user may have several document prototypes in his work space, each with its own list of concepts, and a single concept may be used in more than one prototype. When several prototypes are so defined, the result is a search that retrieves documents that are relevant to all the concepts in one OR all the concepts in the other. In this way the traditional OR conjunction is represented. Figure 8 illustrates the document prototypes used for a typical search.

After building one or more prototypes, the user selects those that he wants to utilize in a search, and then issues the search command. The results of the search are then made available to the user.

Viewing and Evaluating Search Results

In order to initiate a search, the user will select one or more document prototypes and then issue the search command. The system will then match the search case descriptions against those of the documents stored in the system in order to select those that pertain to the topics defined for a particular document prototype and to give them a relevancy weight

When the initial search is complete, the user will be presented with a list of the found documents ordered by decreasing relevancy. This aids in perusing the list and in concentrating on what constitutes a good document, rather than having the user focus on the shear number of documents found. The user will then evaluate the actual usefulness and relevance of the documents and mark each one as being an excellent example of the desired documents, a document to keep in the results, or a document to be discarded from the results.

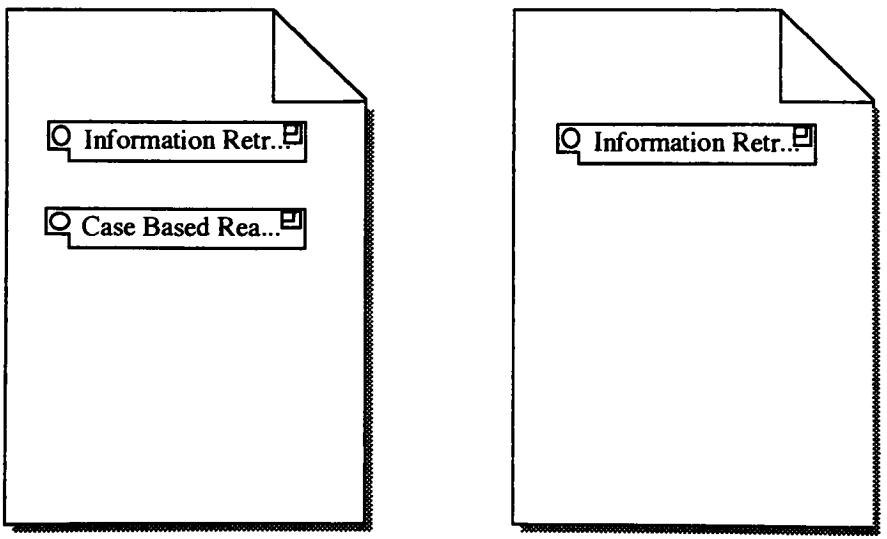
Once the documents have been rated, the user may update the concepts or may update the query, based on the documents rated as excellent. In updating the concepts, the user is presented with a list of concepts from the query and a list of terms and phrases from the documents that were rated excellent but were not included in any of the concepts. The user may pick a concept and then pick terms and phrases to add to it. This makes it easy for the user to improve the definition of concepts based on phrases from documents that are relevant to the concepts. In updating the query, the user is presented with a list of concepts from the query and a list of concepts from the documents that were rated excellent but were not included in the query. This user may pick a concept to add to the query or a concept in the query to remove from it. This makes it easy for the user to make the query example more like the documents he desires.

After rating the documents and improving the query and its concepts, the results may be regenerated. The underlying value assigned to each concept in the query is recalculated based on the documents rated excellent. Any documents marked for discarding are eliminated and the results are regenerated based on the more refined understanding provided by the evaluation.

2.1 Functional Specification

This section provides a detailed functional definition of the system. Each function that the system performs is described at the level of user interaction. All of the user provided inputs, the outputs to the user and the data stores that are visible at the system level are defined. The design and implementation of the system are described in section 2.2.

FIGURE 8. REPRESENTATION OF A DOCUMENT PROTOTYPE



In addition to the search and retrieval functions used by a researcher, a utility application that creates the document index from bibliographic data in an interchange format is provided.

User inputs, outputs and files are defined in terms of the objects that a researcher will create and manipulate in order to conduct a search. System inputs, outputs and files are defined in terms of the objects created and manipulated for the purpose of document indexing. Each function is defined in terms of the object or objects that it operates upon and the context in which it is used for a particular purpose by a particular type of user.

Each function corresponds to a particular selection of an object depicted on the user's terminal screen and a command from the user's menu. Thus the operation of the system is also defined by the functional description.

2.1.1 System Objects, Inputs, Outputs and Files

Each object that a user may create and manipulate in order to accomplish some task is defined in this section. Objects are data structures that contain information that users have entered or results generated by the system. Objects generally have a visual representation on the computer screen that aids users in entering information, performing functions and understanding the results.

- Workspace: The system representation of a set of operations that pertain to a general function and the objects to which they may be applied. The following workspaces exist in the system:
  - Search Workspace: The workspace used to conduct searches and view their results.
  - Indexing Workspace: The workspace used to add new documents to the system.
- Phrase: One or more words that express or are associated with an idea, topic, or theme that the researcher wishes to use as part of a concept.
- Concept: A collection of phrases that all relate to a particular idea, topic theme or concept that the researcher wishes to use as part of a document prototype.
- Concept Hierarchy: A collection of related concepts. A more general concept is related in a tree structure to more specific concepts which are subtopics within its domain.
- Document Prototype: A collection of concepts which represent the kind of documents a researcher desires to find.
- Search Result: A collection of document descriptions which match one or more document prototypes. The descriptions are ordered according to their relevance to the concepts defined for a search.

- Document: Bibliographic and content information about a document, including title, author, publication, key words, abstract, etc.
- Document Template: A description of the constituent parts of a document used to guide the automatic extraction of information from a document for a document case. Document templates are selected by an indexer, but are not created by users. They are established as part of system installation.
- Stop List: A list of words and phrases that should *not* be included in the document cases. These would be common words that generally have little or no relation to any particular idea, topic or theme, such as "the," "but," "or," and the like.
- Document Case: The description of a document that the system uses to match against a document prototype to determine if it is relevant to the user's request and that is evaluated against the user's characteristics and the search purpose to determine it's probable usefulness.
- Catalog: A collection of objects of the same type and information about them. A catalog provides some access method for retrieving objects from it and it may relate its objects to other objects. The following catalogs exist in the system:
  - Document Catalog: A collection of documents and document cases, indexed to support matching with document prototypes.
  - Concept Catalog: A catalog of concept hierarchies describing topics that are frequently used in searches on the system.

2.1.2 Functions Performed

Each function that a user may perform is defined. Functions that are provided as part of normal operating system and user interface system facilities are not described.

Each function is specified as an operation that may be performed on one or more objects in a particular context by a particular user.

2.1.2.1 Open, Close, Save or Delete Search Workspace

The search workspace is the system representation of the objects and operations the user may perform. The normal system functions applicable to user application documents apply. Save operations update the system's concept and document catalogs.

2.1.2.2 Create or Delete Concept

The typical object creation and duplication operations may be performed on a concept.

2.1.2.3 Open or Close Concept

A concept may be represented in one of two states: opened, in which the phrase list associated with it may be viewed and edited;

and closed, in which only the concept's name is displayed. For an illustration, see Figure 6. The open and close function switches the concept between these two states.

2.1.2.4 Create, Delete, Modify, Cnt, Copy or Paste Phrase

Typical text editing operations may be used to create or change phrases within concepts. These operations are used by the user in the search workplace to create his concept hierarchies and search purposes.

2.1.2.5 Collapse or Expand Hierarchy

A concept hierarchy, or portion thereof, may be collapsed so that subtopics of a concept are not visible. Likewise, it may be expanded in order to view those concepts that have been hidden.

This function is used by the user in constructing his concept hierarchies.

2.1.2.6 Create or Delete Document Prototype

The typical object creation operations may be performed on document prototypes.

These operations are used by the user in the search workplace to define a search.

2.1.2.7 Attach Concept to (or Remove from) Document Prototype

A concept may be selected from a concept hierarchy and attached to one or more document prototypes. By so doing the user requests that documents be found that are relevant to that concept. A concept may be removed from the document prototype.

This operation is performed by the user in the search workplace to define a search.

2.1.2.8 Search based on Document Prototype

The user may select one or more document prototypes that he has constructed in the search workplace and perform a search based on them. The system searches the system document catalog utilizing case matching for documents that are relevant to each selected document prototype

2.1.2.9 Evaluate Document based on Description

The user may evaluate each document in the result set as being an excellent example of the documents he desires, as being a document to keep in the results even though it is not an excellent example, or as a document to discard from the results. The user's judgement may then be used by the system to improve the results of the search and to improve the presentation of the results.

2.1.2.10 Update Concept based on excellent Results

The system will provide the user with a pick list of concepts from his query and words and phrases from the excellent documents that are not already in one of those concepts. He may then transfer words and phrases to a chosen concept.

2.1.2.11 Update Query based on Excellent Results

The system will provide the user with a pick list of concepts from his query and from the excellent documents that are not already in it. He may then transfer concepts to or from the query.

2.1.2.12 Regenerate Search Results based on Evaluation

After evaluating the search results, the user may have the system regenerate them. The system will conduct another search of the document catalog using case matching to find any documents that may be good matches to those the user has indicated as being most relevant and useful. The search results are then recreated, eliminating documents found to be irrelevant and not useful, and adding any new documents found.

2.1.2.13 Open, Close or Save Indexing Workspace

The Indexing workspace is the system representation of the objects and operations the indexer may perform. The normal system functions applicable to user application documents apply.

2.1.2.14 Extract Document Case from Bibliographic Data

One or more document cases may be extracted from a properly formatted bibliographic interchange file.

This operation is performed by the indexer in the indexing workspace.

2.1.2.15 Add Term to (or Remove from) Stop List

The indexer may add terms to the system's stop list, or remove terms from it. The stop list is a simple text file and this facility is provided through the use of a text editor.

2.1.3 Limitations and Restrictions

The system will be single session only. While multiple users may perform searches and administrative functions, only one may be doing so at a time.

The system will support searches based on only one document prototype at a time.

The system will be implemented on a single CPU and not make extensive use of network services. Users may run the system only from the CPU on which it is located.

Only one document template for indexing will be provided. It will be set up to allow case extraction from the document source information. The source information will be a sample from the INSPEC bibliographic database. The INSPEC database is distributed in the ISO 2709 interchange format, so this facility may be applicable to other databases.

For testing and evaluation purposes, only the "natural language" portions of the INSPEC bibliographic record are used for indexing the documents. This is so that the "key words and phrases," which also are part of the record, may be used to verify retrievals against the independent judgement of a document's content that they represent. This is not an inherent limitation of the system, the keywords and phrases may be used for indexing under normal circumstances, and would improve system performance.

2.2 Verification and Validation

The functioning of the system was verified and its performance evaluated in a series of tests. The tests were performed on a document catalog consisting of 50 documents indexed from the INSPEC bibliographic database. This number of documents is considered to be large enough to create interesting search patterns, but small enough to allow comprehensive understanding of each so that the relevance of each to a particular concept may be assessed.

Each INSPEC bibliographic record contains "natural language" information from the document, specifically its title and its abstract. In addition, it also contains a list of keywords and phrases which are deemed to characterize the document by a human indexer. When indexing the documents, only the title and abstract were used; the keywords and phrases were not used to index the document so that they could be used as an independent assessment of the relevancy of a concept to the document.

2.2.1 Single Concept Search Test

The simple search test verifies the basic functioning of the system, and evaluates the automatic indexing and initial search performance. Ten key words and phrases were randomly chosen from the keyword lists of the documents. A concept catalog was created containing them, and a simple single concept search was performed for each one. In order to obtain the best indication of over-all results, a search threshold of "greater than zero" was used. Doing this assured that all documents with any score at all would be retrieved. This has the effect of increasing recall at the expense of precision.

Results: In all but one case, the documents that contained the concept keyword were retrieved. With the exception of this one case, the documents containing the keyword scored at the top of the list and all relevant documents (including those that did not contain the keyword) were retrieved.

Since every search save one retrieved all relevant documents, the average recall performance for the searches was 90%. The average precision was 43%. Using the threshold to screen lower scoring documents would have

dramatically improved precision with little or no decrease in recall, since all relevant documents were high scoring.

2.2.2 Multi-Concept Search

In order to validate the use of multiple concepts in a prototype, the same randomly chosen 10 concepts were used to conduct 2 and 3 concept searches. The combinations used were suggested by an examination of the documents to find groups of keywords that corollate well. Thus the searches were not constructed randomly, but represented combinations that might actually be used by researches.

In order to test the affect of automatic query refinement based on selected documents from the results, the resulting documents of each search were graded as excellent examples when they were relevant and the search was conducted again. The effect on each individual document's score was evaluated. In one case, a group of documents below the highest scoring group was selected as containing the "best examples" to test the effect of using a lower scoring subset to reformulate the query scores.

Since the overall number of documents was small, and the number of concepts used was small, it was not easy to find examples where a document represented two or more of the concepts well. Thus, while the reasonableness and coverage of the search results was examined, exact calculations of recall and precision were not deemed significant. This test was used more to test the functioning of the system as designed, rather than to evaluate its search performance.

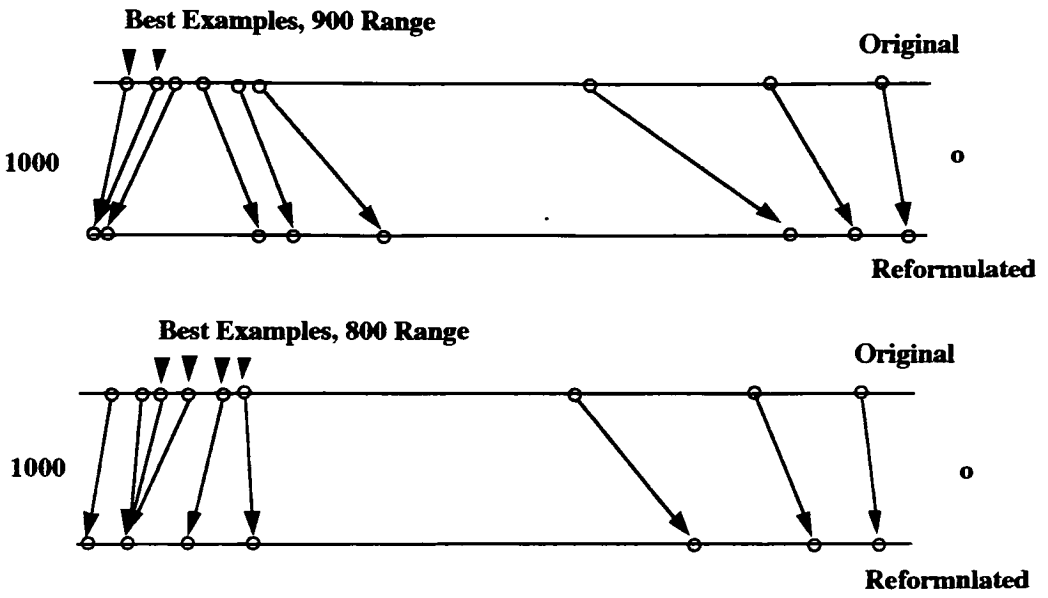
Results: Each document's score was calculated correctly when compared to the hand calculations. The recall was excellent, and the precision, while less than that of the single concept searches, was good.

By selecting the high scoring document (or documents) as a best example in the "result workplace" and running the search again, the document scores were altered as expected, that is, the selected documents' score went up, those close to the selected documents became closer still, and those further from the selected documents moved further away.

In one case a group of documents that did not score the highest was chosen as containing the "best examples." The results for this particular query are typical of the others and are illustrated in Figure 9. The query resulted in 21 documents, with a range of scores from 997 down to a low of 50. As can be seen from the figure, selecting a group of documents as containing the best examples and conducting the search again tends to create a "divide" around the selected documents, where those that are close and greater in score to the group are increased in score to varying degree and those that are less in score are decreased. The further from the selected documents, the more extreme the increase or decrease, except at the end points of the score range (0 and

1000), where there is just no room to push the score further. This creates a tendency to emphasize documents similar to the selected group, and de-emphasize those less so.

FIGURE 9. Query Refinement Based on Best Example Docnments' Scores'



2.2.3 Conceptualization and Query Refinement

Although the randomly selected keywords produced excellent search results, examination of the keywords revealed an excellent example of a concept that could not be simply defined through the use of the keyword alone. This concept was used as the basis for a "case study" of conceptualizing with multiple terms and phrases developed through the query refinement process.

The keyword that was most often used to characterize documents in the sample was "teaching." Apparently, many physics journal articles are written to explain novel approaches for teaching difficult concepts. In only a few cases does the abstract associated with this keyword utilize the word "teaching" or one of its syntactic variants. The texts more often concentrate on the concept that is to be taught, and use oblique language, such as "the concept of entropy is often misunderstood" to convey the notion that the discussion is motivated by the need to teach the concept. In some cases, there is not even this hint to differentiate the abstract from others that are not meant for teaching. The only real clue is the fact that the subject mater contains an element

of novelty or simplicity that would not be present in a "serious" physics paper. This is the case with one abstract that described the creation of a square wheel that rolls on a catenary track. If it were not for our knowledge that this is an entertaining demonstration of mechanics and not a "serious research subject" there would be no way to differentiate this abstract from the pure research variety. This then, is an ideal concept or topic to develop as a test of query refinement and conceptualization.

The simple concept "teaching" was defined and a simple search was conducted looking for it. The results were then used to repeatedly refine the concept by selecting those documents at each cycle that concern themselves with teaching various topics. At each cycle the recall and precision was tracked to the point of diminishing returns.

The first search used only the phrase "teaching." No phrases were added to the concept except those suggested by the concept refinement option of the results menu. This option scans those documents that were judged by the user to be the best examples and presents the user with a pick list of terms and phrases that can be added to the concepts in the query. Only those terms that are not already included in the query's concepts are suggested. After choosing those terms that would help to expand the concept of teaching the search is performed again.

Results: The first search, using the simple concept "teaching" resulted in only two documents found, both relevant. This gave a recall of only 10%, typical of simple keyword searches. The two documents were both selected as good examples and the system suggested terms and phrases with which to expand the concept. In all, 25 phrases were added to the concept, including such phrases as "Undergraduate Student," "Lecture Demonstration," "Difficult Theory" and "Inexpensive and Simple."

Upon repeating the search, 31 documents were found, of which 14 were relevant and 17 irrelevant. With a total of 20 relevant documents in the database, this gives a recall of 70% and a precision of 45%. Although many new phrases were suggested by the system that could be deemed typical of abstracts relevant to teaching, the 6 unfound documents could not be retrieved using them, and the point of diminishing returns had been reached.

The remaining documents were of the sort mentioned above, that is, they contained no direct vocabulary that indicated that they were meant for teaching purposes. Most of them I would not have judged relevant to teaching and simply relied on the keyword designation provided by INSPEC. One was obviously meant for teaching, but could only be judged so based on the simplistic and entertaining nature of the subject matter and one's expectations for serious research topics. This would be an excellent test case for a much more extensive knowledge based retrieval system, but cannot be handled by this system.

One characteristic of this retrieval bears mentioning. Due to the inclusion of a large number of phrases, the precision of the retrieval suffered. This would not be a problem if the irrelevant documents tended to have low scores while the relevant documents tended to have high. Then the sorted results coupled with a threshold capability would increase the usefulness and precision of the results. However, there were quite a few high scoring irrelevant documents, and this tends to negate the advantages of ordering and using a threshold. Since the user interface allows such documents to be deleted from the results, however, the user still ends up with a "pure" list of relevant documents. Nonetheless, a more sophisticated technique for scoring a concept based on its terms could use the information provided by the concept catalog to discover which concepts have overlapping terms, and adjust for this during scoring to increase precision.

3.0 Conclusions

The test results from the evaluation showed that the system has very good performance characteristics compared to other information retrieval systems. The typical keyword information retrieval system has a recall of about 20% and a precision of about 35%. One knowledge based system achieved a recall of 52% and a precision of 46%. By contrast, the current system's initial testing showed a recall of about 90% and a precision of 43%. The system's performance improves over time as more concepts are added to the concept catalog and as those concepts are refined through user feedback. The conceptual arrangement of the terms and phrases used in the search makes it simple for users to utilize the queries developed by other users for their own purposes. The user interface provides a simple, intuitive way of developing and refining searches.

Since the tests were conducted on a sample database of only 50 documents, the performance figures do not necessarily predict the performance of a full size system, which may contain as many as 260,000 documents. However, they do provide an indication that such a system would have superior performance compared to traditional information retrieval systems.

3.1 Problems Encountered and Solved

The largest problem encountered during implementation of the system was simply the storage and retrieval of data objects on disk. The system was coded utilizing C++ and object oriented techniques. Writing and reading objects on secondary storage was designed to fit into the object-oriented paradigm where an object contains methods that allow it to be saved to disk and later initialized from storage. This is known as "persistent objects." For typical information objects this is a straight-forward programing task. However, the objects in this system are highly inter-related, and the relations were implemented through the use of pointers. This caused the difficulty of having to model the computer's memory space on disk so that the relationships could be maintained during storage and retrieval. This is, indeed, a central problem in the implementation of "persistent objects" and object-oriented databases. Unfortunately, while an interesting problem in and of itself, it is not central to the topic of this thesis. The implementation of the memory model allowing persistent objects took an inordinate amount of time in relation to its significance for the project. Other aspects of the system were quite simple and easy to implement, once they were thought out. The use of a commercially available object oriented data base for disk storage would have been in order, but one was not available.

3.2 Discrepancies and Shortcomings of the System

One shortcoming of the system bears mention. The storage and retrieval of objects on disk is rather crude. An entire collection of similar objects (a "catalog") is stored and retrieved in one operation as a whole. This imposes a performance and capacity liability on the system. Since the data must be retrieved all at once, a long "start up" response time is required and only as many objects as can be held in memory at once may occupy the system. This is not adequate for anything other than an experimental

system. An "on demand" storage and retrieval facility including asynchronous input-output operations with read-ahead and buffering should be employed.

3.3 Lessons Learned

During implementation and evaluation two improvements or alternative approaches suggested themselves for the system.

3.3.1 Improvements and Alternative Approaches

1. The system contains an object, called a concept vector, which represents both a query and a document's index. This object utilizes a concept's unique identifier as the means of "including" a concept in the vector. This makes matching the concepts from two vectors easy. The system also contains a term vector which represents both a concept's terms and phrases and the terms and phrases in a document. In order to simplify the stemming operations and combinatorial phrase generation process, the terms and phrases in the term vector are represented by the text that constitutes them. This makes matching terms and phrases more complex than matching concepts, and it makes it hard to present a "natural language" representation of a term or phrase to the user because they have been put into a "canonical form."

Rather than representing terms and phrases in this manner, it would be better to represent them using the same technique used for concepts, utilizing a unique identifier from a term catalog. This would allow easy presentation of the term or phrase to the user, and it would allow the collection of comprehensive occurrence data in the catalog where it would be easy to access.

With a term catalog, statistics concerning term usage as a whole could be used to create a more sophisticated term scoring system, based on the frequency of a term in the database as a whole and its distribution characteristics across documents and redundancies between concepts. It would also allow an easy way to display the full text version of the term to the user, and it would allow the term and concept vector to share a single implementation of many methods.

2. During the query refinement phase of a search, the user has the ability to select terms and phrases from the excellent documents and add them to the query concepts. However, I found that sometimes I wanted to add a term or phrase to a concept which not only didn't exist in the query, but which may not have existed in the system and needed to be created. It would make a great improvement in the user's ability to refine a query to provide a "create concept" function for the concept refinement dialog under the results menu.

3.3.2 Suggestions For Further Work

1. A problem with typical information retrieval systems which was not explored is that while retrieved information may be relevant, it may not be useful. For instance, the document's treatment of its topic may be too theoretical for one user or much too practical for another. A person with experience

in a particular area does not need introductory articles, but that is exactly what a novice is looking for. Information retrieval systems do contain information about documents that can be used to judge usefulness, such as the treatment that a document gives its subject or the kind of journal in which it is published. This information is used by the expert research assistant in finding not only the most relevant documents, but also those documents that are most useful for his client. The typical information retrieval system cannot judge usefulness because it does not have any information about a user's background or purpose in conducting the search. Some systems provide access to attributes of documents such as their treatment of subject matter and the publication in which they appear as part of the search query, but in order to use these attributes the user must have an appreciation for how they indicate usefulness. The professional research assistant can use them profitably, but the casual, rather than expert, user will find it more difficult to narrow his search on this basis.

Like the expert who helps conduct a literature search, the system could base its search and retrieval not only on the subject areas that the user is interested in, but also on the purpose for which the user intends to use the retrieved information as well as on the user's characteristics, such as his academic background and the languages he reads. An undergraduate student who is interested in writing a class paper on some subject such as X-Windows, a graduate student interested in the same subject as the general topic of a thesis, and a system administrator looking for X-Windows products for possible purchase should all receive different results from their search, just as they would if they each had consulted an expert, who has an intuitive feel for what types of documents fit various purposes and users.

Each user should have his own search workspace. When the user creates it, he would be asked to give certain characteristics about himself, such as academic background, occupation, and languages that he reads. As part of his query, he would be able to specify a purpose for which he wants the retrieved documents. Each document indexed in the system would have a number of characteristics and attributes, such as the type of document, the magazine or journal in which it is published, the type of magazine or journal, and the language it is written in. These characteristics will be matched against the user's characteristics and purpose during the search process to indicate the document's usefulness.

The correlation of user characteristics and purpose with document attributes should be based on the user's judgement of usefulness made after the results of a preliminary search have been presented. Over time the system's ability to predict usefulness should increase based on user feedback.

2. When concepts contain many terms and phrases, there is a good chance that many of them are relevant to several concepts. When this is the case, the

precision of the retrieval suffers. This would not be a problem if the irrelevant documents tended to have low scores while the relevant documents tended to have high. Then the sorted results coupled with a threshold capability would increase the usefulness and precision of the results. However, there may be high scoring irrelevant documents, and this would tend to negate the advantages of ordering and using a threshold. This phenomenon was observed during testing. A more sophisticated technique for scoring a concept based on its terms might use the information provided by the concept and term catalog to discover which concepts have overlapping terms, and adjust for this during scoring to increase precision.

4.0 Bibliography

1] American Association for Artifieial Intelligence.
Proceedings: Tenth National Conference on Artificial Intelligence.
AAAI Press, 445 Bugess Drive, Menlo Park, CA 94025. 1992.

2] R. Barletta.
An introduction to case-based reasoning.
AI Expert 6 (8): 42-9.

3] D. H. Berman
Developer's choice in the legal domain. The Sisyphean journey with CBR or down hill with rules.
In Richard Suskind [44].

4] D. C. Blair and M. E. Maron.
An evaluation of retrieval effectiveness for a full-text document-retrieval system.
The Communications of the ACM, 28 (3), March 1985, 289-299.

5] Quoted in Mauldin [33].

6] Quoted in Gelbart [14].

7] A. Bookstein, Y. Chiaramella, G. Salton, and V. V. Raghavan.
Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval.
Association for Computing Machinery Press, New York, 1991.
Published as a special issue of the *SIGIR Forum*.

8] A. Celentano, M. G. Fugini and S. Pozzi.
Querying office systems about document roles.
In A. Bookstein [7].

9] H. Chen.
Knowledge-based document retrieval: framework and design.
Journal of Information Science, Principles & Practice 18 (4): 293-314, 1992.

10] A. Julian Craddock.
Common sense retrieval.
In AAAI [41].

11] S. Cazalens and R. Demolombe.
Intelligent access to data and knowledge bases via user's topics of interest.
IFIP Transactions A: Computer Science and Technology. A (14): 245-251, 1992.

12] J. P. Dick.
Representation of legal text for conceptual retrieval.
In Richard Suskind [44].

13] Daniel C. Edelson.
When should a cheetah remind you of a bat?
In AAAI [41].

14] Daphne Gelbart and J. C. Smith
FLEXICON, a legal text-based intelligent system.
In Richard Suskind [44].

15] A. K. Goel.
Representation of design functions in experience-based design.
IFIP Transaction B, Applications in Technology B (4): 283-308, 1992.
The Journal of the IFIP Technical Committee 5 on Computer Applications and Technology.

16] P. Harmon.
Case-based reasoning I
Intelligent Software Strategies 7 (11): 1-8, 1991.

17] P. Harmon.
Case-based reasoning II
Intelligent Software Strategies 7 (12): 1-9, 1991.

18] M. H. Ibrahim.
TRIBE: a knowledge acquisition and representation model for learning text recognition rules from examples.
Proceedings of the Second International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Volume 2.
Association for Computing Machinery, New York, 1989.

19] Syu Inien and Lang Sheau-Dong.
A knowledge-based approach to conceptual information retrieval from full text.
Managing Information Technology in a Global Society: Proceedings of the 1991 Information Resources Management Association International Conference.
Idea Group Publishing, Harrisburg, PA. 1991.

20] Eric K. Jones.
Model-based case adaption.
In AAAI [41].

21] James Janicki.
Retrieval from an image knowledge base.
Master's thesis, Rochester Institute of Technology, 1993.

22] Young Whan Kim, Won Gyu Cho and J. H. Kim.
Deducing conceptual distance from a hierarchical thesaurus in a knowledge based information retrieval model.
InfoJapan '90: Information Technology Harmonizing with Society, Volume 2.
North-Holland, Amsterdam, Netherlands. 1990.

23] Hiroaki Kitano et al.
Building large-scale and corporate-wide case-based systems: Integration of organizational and machine executable algorithms.
In AAAI [41].

24] Janet L. Kolodner.
An introduction to case-based reasoning.
Artificial Intelligence Review 6 (1): 3-34, 1992.

25] Janet L. Kolodner.
Retrieving events from a case memory: a parallel implementation.
Proceeding from Case-Based Reasoning Workshop, 1988.

26] Granino A. Korn.
Neural Network Experiments on Personal Computers and Workstations.
M.I.T. Press, Cambridge, MS, 1991.

27] Srinivas Krovvidy, William G. Wee and C. Y. Han.
Case based reasoning approach for heuristic search.
Proceeding of the International Society for Optical Engineering - Applications of Artificial Intelligence IX, 1991.

28] A. Lichnerowicz.
Manipulable inter-medium encoding for information retrieval.
Proceedings of the RIAO Conference on Intelligent Text and Image Handling.
Elsevier, Amsterdam, Netherlands. 1991.

29] Julie Beth Lovins.
Developent of a stemming algorithm.
Mechanical Translation and Computational Linguistics 11 (1-2): 11-21, 1968.

30] V. Mital, A. Stylianou and L. Johnson.
A system that takes a 'purposive' view of conceptual information retrieval.
British Computer Society 13th Information Retrieval Colloquium, 1991.
University of Lancaster, Lancaster, United Kingdom. 1991.

31] V. Mital, A. Stylianou and L. Johnson
Conceptual information retrieval in litigation support systems.
In Richard Suskind [44].

32] L. Mohan and R. L. Kashyap.
A dialog based interface to a design knowledge base that understands user design-intentions.
The Second International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Volume 2.
Association for Computing Machinery, 1989.

33] M. L. Mauldin.
Retrieval performance in FERRET: a conceptual information retrieval system.
In A. Bookstein [7].

34] H. P. Ohly.
Conceptual information retrieval by knowledge-based programming techniques.
International Classification 18 (3): 148-152, 1991.

35] D. E. Rose and R. K. Belew.
A connectionist and symbolic hybrid for improving legal research.
International Journal of Man-Machine Studies 35 (1): 1-33, July 1991.

36] Gerard Salton and Michael J. McGill.
Introduction to Modern Information Retrieval.
McGraw-Hill Book Company, New York. 1983.

37] Page 120.

38] G. G. Shin and K. B. Irani.
Fragmenting relations horizontally using a knowledge-based approach.
IEEE Transactions on Software Engineering 17 (9): 872-883, Sept. 1991.

39] K. P. Sycara and D. Navinchandra.
Representing and indexing design cases.
The Second International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Volume 2.
Association for Computing Machinery, 1989.

40] Roger C. Schank.
Dynamic Memory: A Theory of Reminding and Learning in Computers and People.
Cambridge University Press, 1982.

41] Stephen Slade.
Case-based reasoning: a research paradigm.
AI Magazine: 12 (1), 1991. Spring, 1991.

42] E. Simoudis and J. Miller
Validated retrieval in case-based reasoning.
Eighth National Conference on Artificial Intelligence, Volume I.
MIT Press, Cambridge, Massachusetts, 1990.

43] Richard H. Stottler.
Case-based reasoning for bid preparation.
AI Expert, March 1992.

44] Richard Suskind, editor.
The Third International Conference on Artificial Intelligence and Law: Proceedings of the Conference.
Association for Computing Machinery, P.O. Box 64145, Baltimore, MD 21264; 1991.

45] K. P. Sycara and D. Navinchandra.
Representing and indexing design cases.
Proceedings of the Second International Conference on Industrial and Engineering Application of Artificial Intelligence and Expert Systems, Volume I.
Association for Computing Machinery, New York, 1989.

46] A. F. Smeaton.
Progress in the application of natural language processing to information retrieval tasks.
Computer Journal 35 (3): 268-278, June 1992.

47] I. Solvberg, I. Nordbo and A. Aamodt.
Kowledge-based information retrieval.
Future Generation Computer Systems 7 (4): 379-390, May 1992.

48] P. Sheridan and A. F. Smeaton.
The application of morpho-syntactic language processing to effective phrase matching.
Information Processing & Management 28 (3): 349-369, 1992.

49] Anne Tißen.
A case-based architecture for a dialogue manager for information-seeking processes.
In A. Bookstein [7].

50] R. M. Tong, L. A. Appelbaum and V. N. Askman.
A knowledge representation for conceptual information retrieval.
International Journal of Intelligent Systems 4 (3):259-283, Fall 1989.

51] B. Vickery.
Classificatory principles in intelligent interfaces.
Proceedings of the 1st International ISKO-Conference on Tools for Knowledge Organization and the Human Interface.
Indeks Verlag, Frankfurt / Main, Germany. 1990.