FUZZY LOGIC APPLIED TO A TEXT RETRIEVAL APPLICATION

By

UMANG R. SHETH

Bachelor of Engineering

L. D. College of Engineering

Ahmedabad, India

1988

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 1991

FUZZY LOGIC APPLIED TO A TEXT RETRIEVAL APPLICATION

Thesis Approved:

Blayne E. Mayfield

Thesis Adviser

[illegible]

J Chandler

Thomas C. Collins

Dean of the Graduate College

ACKNOWLEDGMENTS

I wish to express my heartfelt appreciation to Dr. Blayne E. Mayfield for his encouragement and excellent guidance throughout my thesis. I also like to extend my thanks to Dr. G. E. Hedrick and Dr. W. D. Miller for their advice and for serving as members of my graduate committee.

I would like to show my sincere appreciation to Mr. Brent Phillips and Mr. Russ Teubner, at the Teubner & Associates, Inc., for providing the right atmosphere for my research.

Many thanks are due to my parents, Rajnikant Sheth and Niranjana Sheth, for their love, and moral as well as financial support.

Special thanks to my fiancee, Anu, for her moral support and encouragement.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

Much of human reasoning is approximate rather than precise in nature [25]. Humans reason in approximate terms on many occasions when they decide on things like 'which route to take to a desired destination' or 'what approach to use in proving a theorem' or 'who can be considered tall'. Indeed, only a small fraction of our thinking can be categorized as precise in either logical or quantitative terms.

Fuzzy logic is based on the concept that decision making is not always a matter of black and white, true or false; it often involves gray areas and the term 'might be'. The simplest way of characterizing fuzzy logic is to say that it is a logic of approximate reasoning [27].

In more specific terms, what is central about fuzzy logic is that, unlike classical logical systems, it aims at modelling the imprecise reasoning that plays an essential role in the remarkable human ability to make rational decisions in an environment of uncertainty and imprecision [28].

Fuzzy logic has found numerous applications in the real world, controlling processes and making decisions. Fuzzy logic is applied in kiln management during the process of making cement by F. L. Smith & Co. of Denmark in 1980 [1]. Japanese manufacturers have taken the lead in commercializing fuzzy logic technology. Hundreds of fuzzy system applications have been developed in Japan. A few of the commercial products that use fuzzy logic have been video cameras, with fuzzy controllers used in the focusing mechanism, washing machines that measures soap, water, and agitation to fit the dirtiness of the clothes in the machine, and air conditioners that can cool a room to preset temperature better than a standard thermostatically-controlled model. Fuzzy controllers also have been developed to operate a subway system in Japan automatically [1].

This research concentrates in the application of fuzzy logic in the text retrieval application. The value of fuzzy logic in this application is the ability to locate records quickly that not only exactly match search criteria, but also varying degrees of partial match criteria.

CHAPTER II

LITERATURE REVIEW

The first paper on fuzzy sets was published in 1965 by Professor Lotfi A. Zadeh [25], who pioneered the field of fuzzy logic. In this paper Professor Zadeh introduces the seminal idea and defines inclusion, union, intersection, complement, relation and convexity.

Zadeh [25] writes :

> The notion of a fuzzy set provides a convenient point of departure for the construction of a conceptual framework which parallels in many respects the framework used in the case of ordinary sets, but is more general than the latter and, potentially, may prove to have a much wider scope of applicability, particularly in the fields of pattern classification and information processing. Essentially, such a framework provides a natural way of dealing with problems in which the source of imprecision is the absence of sharply defined criteria of class membership rather than the presence of random variables.

Two-Valued and Multi-Valued Logical Systems

In classical two-valued systems, the two truth values are: true and false (or one and zero). All classes are assumed to have sharply defined boundaries. So either an object is a member of the class or it is not a member of a class. Some of the examples of the classes that have sharp boundaries may be - dead or alive, male or female, mortal or immortal.

But most classes in the real world do not have sharp boundaries. For example, characteristics or properties like tall, bald, intelligent, tired, sick, and so forth lack sharp boundaries. Classical two-valued logic is not designed to deal with properties that are a matter of degree [24].

One generalization of two-valued logic is called multi-valued logic. In multi-valued logical systems, a property can be possessed to a degree. In multi-valued logical systems, there are more than two truth values. There may be a finite or even an infinite number of degrees to which a property may be possessed. For instance, in a three-valued system, something can be true, false, or on the boundary. In terms of set membership, in a ternary logic, we might have

* something in the set
* something not in the set
* something might be in the set

An example of the situation that requires multi-valued logic is the statement "Tom is tall", for tall is a property that requires an infinite number of truth values to describe [24].

Difference between Multi-Valued Logic and Fuzzy Logic

Even though multi-valued logical systems have been available for some time, they have not been used to any significant extent in linguistics, in psychology, or in other fields where human cognition plays an important role. Multi-valued logic has not gone far enough in the problems and application areas in which the underlying information is imprecise. The reason is that the framework of two-valued and even multi-valued systems is restricted. It is difficult to deal with these problem areas, systematically, using such restricted framework. This is where fuzzy logic enters the picture.

The main difference between fuzzy logic and multi-valued logic is that fuzzy logic deals with quantifiers like 'most', 'few', 'many', 'several', 'more or less', 'slightly', 'quite a few', 'very many' etc. In multi-valued logic the only two quantifiers available are 'all' and 'some'.

Another key difference is that in fuzzy logic truth itself is allowed to be fuzzy. So it is acceptable to say that something is 'quite true' or 'it's more or less true'. It is possible to use fuzzy probability like 'not very likely' or 'almost impossible'.

The crux of the problem is the excessively wide gap between the precision of classical logic and the imprecision of the real world. By having fuzzy quantifiers, mentioned before, fuzzy logic is far better suited for dealing with real world problems than traditional logical systems.

Thus, fuzzy logic provides a better match with the imprecise real world than a sharply defined two-valued logic. In addition to serving as a basis for reasoning with common sense knowledge, it provides a reasoning system where the underlying information is imprecise. In this way fuzzy logic provides a system that is sufficiently flexible and expressive to serve as a natural framework for the semantics of natural language [24].

Fuzzy Sets and Operations on Fuzzy Sets

Fuzzy sets can be defined without making any reference to measures of uncertainty by changing the usual definition of the characteristic function of a set in order to introduce *degrees* of membership.

A fuzzy set F is equivalent to giving a reference set Ω and a mapping, μ_F, of Ω into [0,1], the unit interval. $\mu_F(\delta)$ for $\delta \in \Omega$, is interpreted as the degree of membership of δ in the fuzzy set F [5]. Standard set-theoretic representation of F is as following.

$F_\alpha = \{ \delta \in \Omega \mid \mu_F(\delta) >= \alpha \}$

where $\alpha \in [0,1]$

Elementary fuzzy set operations:

The notions of inclusion and equality extend to fuzzy sets: the definitions most widely accepted are due to Zadeh:

Inclusion: $F \subseteq G$ $\quad \forall\ \delta,\ \mu_F(\delta) <= \mu_G(\delta)$

Equality: $F = G$ $\quad \forall\ \delta,\ \mu_F(\delta) = \mu_G(\delta)$

The principal set-theoretic operations (complementation, intersection, and union) have been defined by Zadeh for fuzzy sets as follows:

Complementation: The fuzzy set $\bar{F}$, the complement of F in Ω, is defined by

$$\forall\ \delta,\ \mu_{\bar{F}}(\delta) = 1 - \mu_F(\delta)$$

Intersection: The intersection $F \cap G$ of two fuzzy sets F and G on Ω is defined by

$$\forall\ \delta,\ \mu_{F \cap G} = \min(\mu_F(\delta),\ \mu_G(\delta))$$

Union: The union $F \cup G$ of two fuzzy sets F and G on Ω is defined by

$$\forall\ \delta,\ \mu_{F \cup G} = \max(\mu_F(\delta),\ \mu_G(\delta))$$

Probability Theory vs Possibility Theory

In science, it is traditional to deal with uncertainty through the use of *probability theory* and *interval analysis*. In recent years, however, it has become increasingly clear that there are some important facets of uncertainty which do not lend themselves to analysis by classical probability-based methods. One such facet is that of *lexical elasticity*, [5] which relates to the fuzziness of words in natural languages. As a case in point, even a simple relation between X, Y and Z, such as *'if X is small and Y is very large then Z is not very small'*, does not lend itself to a simple interpretation within the framework of probability theory by reason of the lexical elasticity of the predicates *small* and *large* [5].

The development of possibility theory as a branch of the theory of fuzzy sets was motivated in large measure by the need for a systematic way of dealing with lexical elasticity and other forms of uncertainty which are not probabilistic in nature. Basically, possibility theory--as its name applies--deals with the possible rather than probable values of a variable with possibility being a matter of degree [5].

Possibility theory, formulated by Zadeh in 1977 [5], offers a model for the quantification of judgment which also allows a canonical generalization of the interval analysis. In this framework, the uncertainty of an event is described both by the degree of possibility of the event itself and by the degree of possibility of the contrary event, these two measures related being only weakly. The complement (with respect to 1) of the possibility of the contrary event can be interpreted as the degree of necessity (certainty) of the event itself. i.e.,

$$\text{Nec}(x) = 1 - \text{Poss}(x)$$

where,

Poss(x) = possibility of the event contrary to the event x

Nec(x) = necessity of the event x

Thinking about uncertainty in terms of more or less possible events and more or less certain events appears natural, and commonly seems to be employed by the human mind [5].

Possibility theory emerged from the notion of fuzzy sets. This concept tries to take account of the fact that an object may more or less correspond to a certain category in which one attempts to place it. When the degree of possibility can assume only the black-or-white values of 0 or 1, the calculus of possibility exactly coincides with interval analysis, in which the imprecise information is represented in terms of sets of possible values (instead of exact values). In a calculus of degree of possibility, such sets become fuzzy sets [5].

The idea of adding individual degree of truths (DTs) and finding cumulative degree of truth, which is discussed in Chapter IV, is based on possibility theory.

Role of Fuzzy Logic in the Management of Imprecise Knowledge and Uncertainty in Expert Systems

The knowledge base of an expert system, which is a repository of a small subset of human knowledge, is a collection of rules and facts, which are neither totally certain nor totally consistent. The uncertainty of information in the knowledge base of any question-answering system induces some uncertainty in the validity of its conclusions. The transmission of uncertainty from the premises to the conclusion is made by associating a *certainty factor* with the conclusion. In existing expert systems, the computation of certainty factors is achieved through a combination of methods that are based on both two-valued logic and probability theory. A serious shortcoming of this method is that it is not capable of coming to grips with the pervasive fuzziness of information in the knowledge base [26].

In 1983, Zadeh [26] suggested an alternative approach to the management of uncertainty in expert systems is the use of fuzzy logic. According to Zadeh [26]:

> A feature of fuzzy logic which is of particular importance to the management of uncertainty in the expert systems is that it provides a systematic framework for dealing with fuzzy quantifiers, e.g. most, many, few, very many, almost all, infrequently, about 0.8 etc. In this way, fuzzy logic subsumes both predicate logic and probability theory, and makes it possible to deal with different types of uncertainty within a single conceptual framework.

A knowledge base of a typical expert system that uses fuzzy logic consists of fuzzy rather than nonfuzzy (crisp) propositions. The problem of inference from these fuzzy propositions then is reduced to the solution of a nonlinear program. Detailed description of this approach appears in [26].

Negoita [12] also discusses the problem of imprecise knowledge in expert systems. He deals with semantic manipulations versus symbolic ones. If inference procedures are augmented with mechanisms for plausible reasoning, then conclusions are drawn from facts that seem to be correct. If inference procedures are augmented with mechanisms for approximate reasoning, then conclusions are drawn from the facts considered to be fuzzy. In expert systems, based on semantic manipulation and approximate reasoning, the emphasis is on fuzziness viewed as an intrinsic property of natural language.

The paper by Ralescu et al.[15] shows that methods of probability theory and of fuzzy set theory can be combined to model different sources of inexactness. It is shown that this combination makes possible the assessment of credibility in expert systems.

In 1986, Togai and Watanabe [22] designed and fabricated a chip for real-time approximate reasoning based on the "max-min operation" of fuzzy set theory. Togai Infralogic, Inc. is the first company to develop a fuzzy logic chip [1]. Potential applications of this inference engine for real-time include decision making in command and control areas, intelligent controllers, and intelligent robot systems.

Document Retrieval using a Fuzzy Knowledge Based Systems

The rapid advances in the fields of science and technology have resulted in an abundance of bibliographic material available in the libraries. This makes it difficult for libraries and other information centers to provide users with up-to-date reference material on topics of their interest.

Subramanian et al.[19] present the design and development of a prototype document retrieval system using a knowledge based systems approach. A query by a user may include fuzzy terms and fuzzy relational operators. Using a natural language interface, this query is converted to an unambiguous intermediate form. The retrieval mechanism consists of two steps. The first step produces a pruned list of documents pertinent to the query. In the second step, the inferencing mechanism computes a measure of relevance between a document characterization and intermediate query form.

The fuzzy knowledge based approach for document retrieval by Subramanian et al. has two main advantages over conventional approaches.

(1) Partial matches can be handled.

(2) Systems capability is enhanced by defining a fuzzy relational operator 'BASED_ON'.

The fuzzy relational operator 'BASED_ON' is used in addition to conventional boolean operators AND, OR and NOT. The study by Subramanian et al. revealed that a number of user queries in natural language containing connectives such as "in context of" and "related to" best translate to the 'BASED_ON' operator.

Selective Dissemination of Information (SDI)

The SDI system was introduced in the early 1960's. The concept of SDI system was introduced by Luhn [9] in 1958.

The basic idea of such a system is to keep the researchers continually informed of new documents published in his or her area of specialization so that he or she can keep abreast of the latest developments [11].

While SDI was not intended to be the ultimate achievement in the scientific communication process, nonetheless it was presented in a dazzling format promising immediate reduction in intellectual effort. At a time when there was a shortage of trained scientists and technicians and an urgent need to communicate a staggering accumulation of newly recorded information, SDI was embraced as a viable solution to a problem of overwhelming magnitude and claimed by its architects as an invaluable time-saver for both the user and the librarian [11].

However, later studies by Cole [3] indicated that SDI has been adopted by a surprisingly small percentage of the community it was designed to serve. The main reason given by Cole is that the critical design assumptions underlying SDI are no longer valid. Detailed reasoning follows.

From the literature on SDI, the critical design assumptions underlying SDI were identified as following.
1. Information in scientific and technical publications is applied directly to people's activities.
2. The activities aided by SDI are those involving the reward system of science.
3. Relevant documents are those supporting "normal science" content, i.e., provide a continuous base of knowledge.

These design assumptions are behavioral rather than logical. Thus, the validity of the assumption rests in its ability to be descriptive of a subpopulation under study. The assumption predicts or stipulates behaviorally constant or uniform modes. What is variable for a population is a constant for a subpopulation in question. The assumption lacks validity if subpopulation characteristics are variable rather than constant.

There is a similarity, to a certain extent, between SDI and this research for text retrieval applications, in a way that both work on the idea of full-text retrieval. However, the biggest difference between the two systems is that this research involves application of fuzzy logic to the problem of text retrieval. The value of fuzzy logic in this application is the ability to locate records quickly that not only exactly match search criteria , but also varying degrees of partial match criteria.

Summary

Fuzzy logic is a logic of approximate reasoning. Fuzzy logic framework provides a natural way of dealing with problems in which the underlying information is imprecise. It is far better suited for dealing with real world problems than the traditional two-valued or multi-valued logical systems. With fuzzy quantifiers like 'most', 'few', 'many', 'more or less', fuzzy logic makes it possible to deal with different types of uncertainty within a single conceptual framework.

CHAPTER III

FULL-TEXT DOCUMENT RETRIEVAL SYSTEMS

Basic Idea of Full-Text Document Retrieval

Document retrieval is the problem of finding stored documents that contain useful information [2]. There exists a set of documents on a range of topics, written by different authors, at different times, and at varying level of depth, detail, clarity, and precision. There also exists a set of individuals who, at different times and for different reasons, search for recorded information that may be contained in some of these documents. In each instance in which an individual seeks information, he or she will find some documents of the set useful and other documents not useful; the documents found useful are called *relevant*; the others, not relevant [2].

Ideally, a method should be found to organize the collection of documents such that a person seeking the information can find all and only the relevant items. Let us consider how automatic full-text retrieval method attempts to achieve this goal.

In a full-text retrieval method, full text of all documents in the collection is stored in the computer so that every character of every word in every sentence of every document can be located by the machine. Then when a person wants information from that stored collection, the computer is instructed to search for all documents containing certain specified words or word combinations, which the user has specified [2].

There are two main elements that make the idea of automatic full-text retrieval even more effective. On the one hand, digital technology continues to provide computers that are larger, faster, cheaper, more reliable, and easier to use, and, on the other hand, full-text retrieval avoids the need for human indexers whose employment is increasingly costly and whose work often appears inconsistent and less than fully effective [2].

A pioneering test to evaluate the feasibility of full-text search and retrieval was conducted by Don Swanson and reported in *Science* in 1960 [17]. Swanson concluded that text searching by computer was significantly better than conventional retrieval using human subject indexing. Ten years later, in 1970, Salton, also in *Science*, reported optimistically on a series of experiments on automatic full-text searching [20].

Automatic Indexing versus Manual Indexing

Retrieving documents by subject content occupies a special place in the field of information retrieval because, unlike data retrieval, the richness and flexibility of natural language have a significant impact on the conduct of the search. The indexer chooses subject terms that will describe the information content of the documents included in the database, and the user describes his or her information needs in terms of the subject descriptors actually assigned to the documents. However, there are no clear and precise rules to govern the indexer's choice of appropriate subject terms, so that even trained indexers may be inconsistent in their application of subject terms. Experimental studies have demonstrated that different indexers will generally index the same document differently [21], and even the same individual will not always select the identical index terms if asked at a later time to index a document he or she has already indexed [2].

The problems associated with manual assignment of subject descriptors make computerized, full-text document retrieval extremely appealing. By entering the entire, or the most significant part of, a document text onto the database, one is freed from the inherent evils of manually creating document records reflecting the subject content of particular document; among these, the construction of an indexing vocabulary, the training of indexers, and the time consumed in scanning/reading documents and assigning context and subject terms [2].

Measures of document retrieval effectiveness

In a retrieval environment, it is possible to use as a criterion of system effectiveness the ability of the system to satisfy the user's need for information by retrieving wanted material and rejecting unwanted items. Two measures have been widely used for this purpose, known as "Recall" and "Precision" [17].

A document is considered as relevant if it is judged useful by the user who initiated the search. "Recall" and "Precision" respectively represent the proportion of relevant material actually retrieved and the proportion of retrieved material actually relevant. Ideally, all relevant items should be retrieved and all nonrelevant items should be rejected; such a situation is reflected in perfect Recall and Precision values, equal to 1 [17].

Blair [2] defines "Recall" and "Precision" as follows.

Recall = Number of Relevant and Retrieved / Total Number Relevant

Precision = Number of Relevant and Retrieved / Total Number Retrieved

Drawbacks of present full-text retrieval systems

Blair and Maron [2], in their paper, describe a full-text search and retrieval experiment aimed at evaluating the effectiveness of full-text retrieval. For the purpose of their study, they examined IBM's full-text retrieval system, STAIRS (**ST**orage **A**nd **I**nformation **R**etrieval **S**ystem). The empirical study of STAIRS in a litigation support situation showed its retrieval effectiveness to be surprisingly poor. It was observed that the system retrieved less than 20 percent of the documents relevant to a particular search. Blair and Maron offer theoretical reasons to explain why this poor performance should not be surprising. The study by Blair and Maron, regarding retrieval problems, should be seen as a critique of the principles on which full-text document retrieval systems are based.

The database examined in the study by Blair and Maron consisted about 40,000 documents, representing roughly 350,000 pages of hard-copy text, which were to be used in the defense of a large corporate law suit. Access to the documents was provided by IBM's STAIRS. STAIRS provides facilities for retrieving text where specified words appear either singly or in complex boolean combinations. Retrieval can also be performed on fields such as author, date and document number. For the test, two lawyers and two paralegals participated in the experiment. A total of 51 different information requests, which were translated into formal queries by either of the two paralegals, were generated by the lawyers. The two paralegals, who were familiar with the case and experienced with STAIRS, searched on the database until a set of documents they believed would satisfy one of the initial requests. The original hard copies of these documents were retrieved from files, and copies were sent to the lawyer who originated the request. The lawyer then made an overall judgement concerning the set of documents received, stating whether he or she wanted further refinement of the query and further searching. The information-request and query-formulation procedures were considered complete only when the lawyer stated in writing that he or she was satisfied with the search results for that particular query. The lawyers and paralegals were permitted as much interaction as they thought necessary to ensure highly effective retrieval. The values of the evaluation parameters "Precision" and "Recall" were then calculated for these retrieval requests.

It was observed by Blair and Maron [2] that the values of precision ranged from a maximum of 100% to a minimum of 19.6%. The average value of Precision turned out to be 79%. This meant that, on average, 79 out of every 100 documents were retrieved using STAIRS were judged to be relevant.

Recall ranged from a maximum of 78.7% to a minimum of 2.8%. The average value of Recall was 20%. This meant that, on average, STAIRS could be used to retrieve only 20% of the relevant documents, whereas the lawyers using the system believed that they were retrieving a much higher percentage (i.e. over 75%).

The realization that STAIRS may be retrieving only one out of five relevant documents in response to an information request was surprising. Blair and Maron believe that the following reasons can explain the poor performance of the full-text retrieval systems.

(1) The value of Recall decreases as the size of the database increases, or, from a different point of view, the amount of search effort required to obtain the same Recall level increases as the database increases, at a faster rate than the increase in database size.

(2) On the database under study, there were many search terms that, used by themselves, would retrieve over 10,000 documents. Such *output overload* is a frequent problem of full-text retrieval systems.

(3) In a full-text retrieval system, the users need to specify the exact words/phrases to use the system effectively. It is difficult for users to predict the exact words, word combinations and phrases that are used by all (or most) relevant documents and only (or primarily) by those documents [2].

For example, in the legal case in question, the formal queries were constructed with the search word "accident" along with several relevant proper nouns. Later, it was found that the accident was not always referred to as an "accident" but as an "event", "incident", "situation", "problem" or "difficulty".

(4) Misspellings can be another obstacle. Misspellings, which are tolerable in normal everyday correspondence, when included in a computerized database become literal traps for users.

To summarize the discussion, it can be said that the full-text retrieval system did not work well in the environment in which it was tested. The optimism of early studies of full-text retrieval systems by two pioneers, Swanson [20] and Salton [17], was based on the small size of the database used. However, the theoretical reasons mentioned before explain why full-text retrieval systems applied to large databases are unlikely to perform well.

CHAPTER IV

FUZZY LOGIC APPLIED TO A TEXT RETRIEVAL APPLICATION

Advantages of using Fuzzy Logic Approach for Text Retrieval Application

Fuzzy logic appears to be a good technology to solve the problem of storing and retrieving technical support incidents. A technical support incident consists of some fixed fields (customer name, date etc.) and a free-format abstract of the problem. The fuzzy logic application will allow each individual in the organization to leverage previous experiences with similar support incidents while thoroughly tracking the support history of individual customers [13].

An application of fuzzy technology will allow fast and flexible access to free-format text (such as an abstract). A technical support database is a particularly good example of such an application. When customer or user problems are reported, a new record is opened, with some formatted fields (customer name, date etc.), and a free-format abstract of the problem. When the problem is resolved the solution is also entered and the problem is closed. Abstracts describing similar problems and their solutions can then quickly be selected using natural language queries [13].

The value of fuzzy logic in this application is the ability to locate records quickly that not only exactly match search criteria, but also varying degrees of partial match criteria. After a problem is opened (problem abstract entered into the database) and/or closed (solution abstract entered into the database), a fuzzy logic system allows users great flexibility in specifying search criteria to produce a list of support incidents with relevant information. Only conceptual words need to be specified in the search criteria rather than the exact word patterns that were entered in the previous support incident records. Because of the structure chosen for the database, search could complete very quickly [13].

Phillips [13] writes :

> A database utilizing this fuzzy technology has two primary advantages over conventional databases: 1) Search queries do not have to be given in well defined formats giving the database a 'natural language' search interface, and 2) Fields of the records do not have to be pre-defined, consequently databases do not have to be restructured because of field additions or modifications and you do not have to know in advance all the categories of words you want to search on.

Design approach to Expert Support Program

The design of such an expert support program consists of three main modules.

(1) User interface

This module is responsible for allowing the user to enter the data records.

(2) Indexing module

The indexing software parses the text in the record title and problem and solution abstract section and stores the information in a master word index database. The index database keeps track of each record number, where the word is used, and each position in the record at which the word occurs. A separate database of exception words is also kept so that trivial words such as 'a', 'and', 'the' etc. are not indexed. Appendix A shows the expert support program database structure.

(3) Fuzzy search module

This module is what gives an expert support program its fuzzy capability and gives it an edge over conventional database systems.

While a number of commercial C libraries can be used in the design of the user interface and the indexing module, the proposed research concentrates on the design and implementation of an algorithm that performs fuzzy search on records.

Fuzzy Search Algorithm

A step-by-step description of the fuzzy search algorithm is given below. Reference to the example in Appendix B would help in understanding these steps.

The framework for the design and development of the fuzzy search algorithm was provided by Phillips [13] in his specifications for the expert support program.

Note: Vagueness of the algorithm is due to the fact that certain essential details have been withheld. The reason for this is that the algorithm is proprietary of Teubner & Associates, Inc.

Step 1 : Parse the search string into non-trivial words. Store these words into an array.

For example, if the search string is "industry program means", then after parsing, search words "industry", "program" and "means" are stored into an array.

Step 2 : For each combination of the search words, compute the relative positions of the search words in the search string and store in the *search word array*. The relative positions of the non-trivial words in the search string matter.

Step 3 : Load the *database word array* from the word occurrence database. (See Appendix A for file layout.)

Step 4a : Select the next record to process.

Step 4b : For each combination of search word pairs compute the *degree of truth* (DT) according to certain specified rules.

Add the DTs, calculated using the rules, for each combination, and find the cumulative degree of truth for that record.

Step 4c : Make an entry of record number and corresponding degree of truth in a *blackboard array*, which is
a two-dimensional array that holds the cumulative degree of truth for each record.

Repeat steps 4a, 4b and 4c until no more records remain.

Step 5 : Output the contents of blackboard array. The blackboard array elements are sorted in descending order before displaying.

CHAPTER V

PROOF OF CONCEPT MODEL AND PRELIMINARY EVALUATION

Simulation is an accepted practice in the evaluation of the performance of proposed designs. Simulation can estimate performance as well as test proposed modifications. This chapter describes the implementation of Indexing and Fuzzy Search modules of the Expert Support Program. These modules were written in a modular, procedural style in C language on IBM PS/2 compatible machine. The modular, procedural style was selected to improve readability, maintainability and modifiability. The copy of the source code listing for the Expert Support Program can be obtained from the Computer Science Department, Oklahoma State University.

Indexing Module

The Indexing module runs automatically whenever a new record is entered in the database. The editing module allows new records to be entered into the system, but for the prototype version the data records were entered into the record database separately.

'Essential B-Tree' library by South Mountain Software, Inc., NJ was used to develop the indexing module.

Word Occurrence database (refer to Appendix A) was implemented using following declaration.

```
struct WO_DB{
          char word[20];          /* word */
          int num_app;            /* number of appearances */
          struct l_list *ptr;     /* link to next element */
             };

struct l_list{
          int rec_no;             /* record number */
          int rel_pos;            /* relative position */
          struct l_list *ptr;     /* link to next element */
              };
```

A linked list with the information regarding each word's record number and relative position was maintained. The reason of using a linked list (dynamic memory allocation) is to save memory.

An index on word occurrence database by word was created so that search for a particular word can be faster.

Fuzzy Search Module

The Fuzzy Search module is responsible for making a fuzzy search of the free-format text of the records in the database, in order to bring up an index of database records that most closely match the search criteria.

Main Driver

The main driver functions as the controller for the Expert Support Program.

The driver first calls the Indexing module to create various database and index files. It then calls the Fuzzy Search module. The code that interactively asks the user to input a search string for which he/she wants to search, has been included with the Fuzzy Search module.

Preliminary Evaluation

In an experiment that was conducted as a preliminary evaluation of the Expert Support Program, a group of sample records were entered into the database and the Fuzzy Search module was checked by specifying different search strings.

The 20 sample records that were entered into the database for the preliminary evaluation of the Expert Support Program appear in Appendix C.

Following observations were made during the test runs.
1. Once the database and index files are created, then the software takes just few seconds to do a fuzzy search and come up with the index of the records that most closely match the search string.

2. If all the words specified in the search string are trivial words (e.g. a, an, the etc.) or if none of the search words were found in any of the records, the software gives an appropriate message.

3. Following are few examples showing the results of the program with different search strings.

Note: A 100% indicates an exact match and lesser number indicate less of a match.

(1) Search string : 'Installing under rscs'

Output :

```
Record Number  Closeness
*************  *********

        12        100%

        14        33.3%
```

(2) Search string : 'request form at the back of one'

Output :

```
Record Number  Closeness
*************  *********

        18        100%

        20        100%

        19        80%

        18        20%

        12        20%

        16        20%
```

(3) Search string : 'request form back one'

Output :

```
Record Number  Closeness
*************  *********

      18         73.3%

      20         73.3%

      19         69.7%

      16         25%

      12         25%
```

(4) Search string : 'manuals reference sna'

Output :

```
Record Number  Closeness
*************  *********

      11         69.3%

      18         33.3%

      19         33.3%

      20         33.3%
```

(5) Search string : 'sna reference manuals'

Output :

```
Record Number  Closeness
*************  *********

      11         100%

      18         33.3%

      19         33.3%

      20         33.3%
```

(6) Search string : 'error message'

Output :

```
Record Number  Closeness
*************  *********

      14         100%
```

```
          20       94%

          2        50%

          19       50%

          10       50%

          18       40%
```

(7) Search string : 'cannot execute command'

Output :

```
Record Number  Closeness
*************  *********

          14       60%

          9        33.3%

          5        33.3%

          16       33.3%
```

(8) Search string : 'can't use sends and rcves'

Output :

```
Record Number  Closeness
*************  *********

          3        100%

          5        25%

          7        25%

          9        25%

          19       25%
```

(9) Search string : 'control unit FAXGate connected to went inop'

Output :

```
Record Number  Closeness
*************  *********

          7        96.4%
          6          23.3%

          12         19.3%

          1          16.7%

          2          16.7%

          4          16.7%
```

(10) Search string : 'address in invalid form'

Output :

```
Record Number  Closeness
*************  *********

      16        96%

      18        33.3%

      19        33.3%

      20        33.3%
```

CHAPTER VI

SUMMARY AND RECOMMENDATIONS FOR FURTHER STUDY

A fuzzy logic application approach to the problem of text retrieval was proposed. Fuzzy logic approach allows users great flexibility in specifying search criteria. The fuzzy search on records will result in locating the records that not only exactly match search criteria, but also varying degrees of partial match criteria. A step-by-step algorithm for fuzzy search has been presented.

The preliminary evaluation shows that an application of fuzzy technology will allow fast and flexible access to free-format text. Regardless, there are several directions which future research endeavors might pursue.

First, thesaurus capability can be incorporated into the Expert Support Program. For each word in a search string, the user can select or specify alternate words with the same meaning.

Second, the Expert Support Program can be modified to take care of plurals, alternate forms of same root words, misspellings and abbreviations of the search words.

Third, we must investigate ways to permit users to assign more weight to certain words than to others. For example, a record can be characterized by a short description or profile. A widely used representation scheme is a *vector model* [19], where each record is represented by a set of *concept-weight* pairs. Concept weights can be derived from statistical measure such as frequency of occurrence or it can be decided by experts who are familiar concepts used in domain [19].

Fourth, a fuzzy logic application approach to a text retrieval application can be empirically compared to existing approaches.

REFERENCES

1. Beard, P., "Fuzzy Logic: Buzzword Or Breakthrough?", AI Week 7 (1990), 1-7.

2. Blair, D. C., "An Evaluation Of Retrieval Effectiveness For A Full-Text Document-Retrieval System", Comm. of ACM 28 (1985), 289-299.

3. Cole, E., "Examining Design Assumptions for an Information Retrieval Service: SDI Use for Scientific and Technical Databases", J.Am.Soc.Info.Sc. 40 (1981), 444-450.

4. Croft, W. B., "Document Representation in Probabilistic Models of Information Retrieval", J.Am.Soc.Info.Sc. 32 (1981), 451.

5. Dubois, D. and Prade, H. Possibility Theory-An Approach to Computerized Processing of Uncertainty, Plenum Press, NY and London, 1986.

6. Karr, C., "Genetic Algorithms For Fuzzy Controllers", AI Expert (1991), 26-33.

7. Kaufmann, A. and Gupta, M. M. Introduction To Fuzzy Arithmetic : Theory And Applications. Van Nostrand Co. Ltd., NY, 1985.

8. Klir, G. J. and Folger, T. A. Fuzzy Sets, Uncertainty, And Information, Englewood Cliffs, NJ, 1988.

9. Luhn, H. P., "A Business Intelligent System", IBM Journal of Research and Development 2 (1958), 314-319.

10. Mamdani, E. H. and Gaines, B. R. Fuzzy Reasoning And Its Applications. Academic Press, NY, 1981.

11. Mauerhoff, G. R., "Selective Dissemination Of Information", Advances in Librarianship 4 (1974), 25-62.

12. Negoita, C. V.,Expert Systems And Fuzzy Systems, Benjamin/Cummings Publishing, Calif., 1985.

13. Phillips, B., "Expert Support Program : Fuzzy Logic Applied To A Text Retrieval Application",

Specifications for ESP, Jan. 1991, 1-9.

14. Raju, K. V. S. V. N. and Majumdar, A. K., "Fuzzy Functional Dependencies And Lossless Join Decomposition Of Fuzzy Relational Database Systems", ACM Trans. On Database Sys. 13 (1988) 129-166.

15. Ralescu, A. L. and Ralescu, D. A., "Probability And Fuzziness", Information Sciences 34 (1984), 85-92.

16. Richards, B. L., "When Facts Gets Fuzzy", BYTE 13 (1988), 285-290.

17. Salton, G., "Automatic Text Analysis", Science 168, 3929 (1970), 335-343.

18. Schwartz, T. A., "Fuzzy Tools For Expert Systems", AI Expert (1991) 34-41.

19. Subramanian, V., Biswas G. and Bezdek J. C., "Document Retrieval Using A Fuzzy Knowledge-Based System", Optical Engineering 25 (1986), 445-455.

20. Swanson, D. G., "Searching Natural Language Text By Computer", Science 132, 3434 (1960), 1099-1104.

21. Swanson, D. R., "Information Retrieval As A Trial And Error Process", Libr. Q. 47, 2 (1978), 128-148.

22. Togai M. and Watanabe H., "Expert Systems On A Chip : An Engine For Real-Time Approximate Reasoning", IEEE Expert, Fall 1986, 55-62.

23. Williams, M. H. and Nicholson, K. A., "An Approach To Handling Incomplete Information In Databases", The Computer Journal 31 (1988), 133-140.

24. Yager, R. R., Ovchinnikov, S., Tong, R. M., and Nguyen, H. T. Fuzzy Sets And Applications : Selected Papers By L. A. Zadeh. John Wiley & Sons, NY, 1987.

25. Zadeh, L. A., "Fuzzy Sets", Information and Control 8 (1965), 338-353.

26. Zadeh, L. A., "The Role Of Fuzzy Logic In The Management Of Uncertainty In Expert Systems", Fuzzy Sets and Systems 11 (1983), 199-227.

27. Zadeh, L. A.,"Fuzzy Logic And Approximate Reasoning", Synthese 30 (1975), 407-428.

28. Zadeh, L. A., "Fuzzy Logic", Computer 21 (1988) 83-93.

29. Zadeh, L.A., "Commonsense Knowledge Representation Based On Fuzzy Logic", Computer 16 (1983), 61-65.

30. Zimmermann, H. J. Fuzzy Set Theory - And Its Applications, Kluwer-Nijhoff Publishing, MA, 1985.

APPENDICES

APPENDIX A

EXPERT SUPPORT PROGRAM DATABASE STRUCTURE

WORD INDEX DATABASE
(Fixed fields)

Word # 1
offset in WO DB
Word # 2
offset in WO DB
.
.
.
.
.
Word # n
offset in WO DB

WORD OCCURRENCE DATABASE
(Variable fields)

# of times used
rec#, relative pos.
rec#, relative pos.
.
.
# of times used
rec#, relative pos.
rec#, relative pos.
rec#, relative pos.
.
.

WORD EXCEPTION DATABASE
(Fixed fields)

Exception 1
Exception 2
Exception 3
Exception 4
.
.
.
.

Exception n

RECORD DATABASE
(Variable fields)

Fixed field # 1
Fixed field # 2
.
.
Length of prob. abstract
.
.
Length of soln. abstract
.
.

RECORD INDEX
(Fixed fields)

Record 1 index
Cust name
Cust #
Record 2 index
Cust name
Cust #
.
.
.
Record n index
Cust name
Cust #

APPENDIX B

EXAMPLE SHOWING APPLICATION OF FUZZY SEARCH ALGORITHM TO SAMPLE RECORDS

Sample Records to Verify the Correctness of the Algorithms for Fuzzy Search

Record # 1

This module is responsible for making a fuzzy search to the records in the database, in order to bring up an index of records in the database which most closely match the search criteria in industry program.

Record # 2

Getting the most out of Viewlink requires a sizable time investment. Besides an initially steep learning curve, the very nature of the program means the installation is time-consuming.

Record # 3

GC3 brings together three learning interrelated industry segments in concurrently running exhibits. When you register for TYPE-X, and ART-X you will receive admission to PRINTING EXPO.

WORD EXCEPTION DATABASE
(See Appendix A for file layout)

```
This
is
for
a
to
the
in
with
an
of
which
out
or
and
will
when
```

WORD INDEX DATABASE
(See Appendix A for file layout)

module	00	steep	56
responsible	02	learning	58
making	04	curve	61
fuzzy	06	very	63
search	08	nature	65
records	11	means	67
database	14	installation	69
order	17	time-consuming	71
bring	19	GC3	73
up	21	brings	75
index	23	together	77
most	25	three	79
closely	28	interrelated	81
match	30	segments	83
criteria	32	concurrently	85
industry	34	running	87

program	37	exhibits	89
getting	40	you	91
Viewlink	42	register	94
requires	44	TYPE-X	96
sizable	46	ART-X	98
time	48	receive	100
investment	50	admission	102
Besides	52	printing	104
initially	54	expo	106

WORD OCCURRENCE DATABASE
(See Appendix A for file layout)

00	1	34	2	71	1	104	1
	1 2		1 36		2 71		3 25
			3 7				
02	1	37	2	73	1	106	1
	1 4		1 37		3 1		3 26
			2 23				
04	1	40	1	75	1		
	1 6		2 1		3 2		
06	1	42	1	77	1		
	1 8		2 6		3 3		
08	2	44	1	79	1		
	1 9		2 7		3 4		
	1 33						
11	2	46	1	81	1		
	1 12		2 9		3 6		
	1 33						
14	2	48	1	83	1		
	1 15		2 10		3 8		
	1 27						
17	1	50	1	85	1		
	1 16		2 11		3 10		
19	1	52	1	87	1		
	1 19		2 12		3 11		
21	1	54	1	89	1		
	1 20		2 14		3 12		
23	1	56	1	91	2		
	1 22		2 15		3 14		
					3 20		
25	2	58	2	94	1		
	1 29		2 16		3 15		
	2 3		3 5				
28	1	61	1	96	1		
	1 30		2 17		3 17		
30	1	63	1	98	1		
	1 31		2 18		3 19		

32	1	65	1	100	1
	1 34		2 19		3 22
		67	1	102	1
			2 24		3 23
		69	1		
			2 26		

Search String ---> **"industry program means"**

Parse()

```
num_of_w = 3
words    : industry
           program
           means
```

Load_SW_Array()

combination	offset of word1	offset of word2	difference in rel_pos
industry-program	34	37	1
industry-means	34	67	2
program-means	37	67	1

Search Word Array

Load_DBW_Array()

industry	1	36	3	7
program	1	37	2	23
means	2	24		

Database Word Array

Find_First_Rec()

First record to work on : record # 1

```
Calc_DT()

Record #1
---------

Combination                            Calculation for DT
-----------                            ------------------
   1-2                                     DT = 0.33

   1-3                                     DT = 0.33

   2-3                                     DT = 0

Cumulative Degree_Of_Truth (DT) for Record #1 = 0.66


Record #2
---------

Combination                            Calculation for DT
-----------                            ------------------

  1-2                                      DT = 0.33

  1-3                                      DT = 0

  2-3                                      DT = 0.33

Cumulative Degree_Of_Truth (DT) for Record #2 = 0.66
Record #3
---------

Combination                            Calculation for DT
-----------                            ------------------

  1-2                                      DT = 0.33

  1-3                                      DT = 0

  2-3                                      DT = 0

Cumulative Degree_Of_Truth (DT) for Record #3 = 0.33
```

Out_List()

Record #	Degree_Of_Truth (DT)
1	0.66
2	0.66
3	0.33

Black Board Array

APPENDIX C

20 SAMPLE RECORDS TO PRELIMINARILY EVALUATE THE PERFORMANCE OF THE EXPERT SUPPORT PROGRAM

Record # 1

They are sending to a Toshiba xxx fax machine and sometimes the characters come out distorted looking like barcode characters. On the faxgate machine they are getting some ERR disagreement on bit rate messages. They are sending to KATC tv station. Contact is Cindy Menard 318-236-6393.

Record # 2

We sent some faxes from our faxgate and were sent fine. Call Intel to see if they have a record of the problem. Intel said to adjust the line compensation factor in the setup. However this will increase the amount of error checking done, and consequently the time for each fax to be sent.

Record # 3

Wants to transfer binary files (received faxes) to the host and down to other pc users. Can't use sends and rcves from rje unless you have supertrax (the sendb and rcveb corrupt original files).

Record # 4

I recommended having faxgate automatically receive faxes and copy them over to a directory on a LAN server. Then they could ft up to host. DCA support said that FT/3270 will support binary file transfer using ind$file. E78 has it in version1 and above (upgrade for $95 or buy FT/3270 for $115).

Record # 5

Problem is getting the PDIR FMH2 from power. Temporary fix is to change the receive command in gateway\rje\autord.fg to use the # instead of the & (also drop of the D=...)

Record # 6

June Mengwasser from ABB said that it started working for her ok on logon1. I theorized that the problem was the order of things when the system came up. If the control unit is activated before the VTAM-JES link is started then the logon will be issued, but fail.

Record # 7

The control unit FaxGate is connected to went inop, then when came active the FaxGate unit keeps getting a sense code 0805. (session limit exceeded). He did a VTAM trace, saw that it does init itself, sends bind image, does another init self.

Record # 8

For FileGate they need to get a file down that does not have carriage returns or line feeds but rather is padded to the full FB record length. The current implementation truncates fixed block records at the last printable character.

Record # 9

Use the Rcvep RJE command. The ramification is that filtering out embedded codes will work only on files sent down with 133 fb record length and must use exchange stream not printer stream. Alternative is to create an embedded code called #pad#len e.g. #pad#132 will pad each record to 132 bytes and write it out without putting a carriage return or line feed.

Record # 10

When they send a large file down (500 lines) to the printer definition the RJE software locks up. The PC file open message is given then the pr1=f:9 is the last status.

Record # 11

NSA said they should set the Pacing parameter = 7, (not Vpacing), that it was probably a buffer overrun. 6/20 that did not work, he is going to try to set his buffer up to 4 or 5k (F5 option on rjecfg program). The pacing parameter was shown to be set to 0 in the monitor.log file we took. byte 12 in the Bind indicates pacing, see the SNA Reference manuals.

Record # 12

Installing under RSCS VM. The got the unit to logon okay. But when they route files to FaxGate only the first one comes down then it stops. If they reboot or get out and then in the RJE software then the next one comes down.

Record # 13

The logmode published from NSA was not working, they had to change secprot from A3 to A1 to get it working. To disable the CACHE and the bios functions and to put ARAM=CE00-CFFF on the QEMM statement in config.sys

Record # 14

The CodeView debugger displays an error message whenever it detects a command it cannot execute. Most errors (start-up errors are the exception) terminate the CodeView command under which the error occurred, but do not terminate the debugger. You may see any of the following messages.

Record # 15

Argument to IMAG/DIMAG must be simple type. You specified an argument to an IMAG or DIMAG function that is not permitted, such as an array with no subscripts.

Record # 16

Bad address. You specified an address in an invalid form. For instance, you may have entered an address containing hexadecimal characters when the radix is decimal. You typed an invalid breakpoint number with the Breakpoint Clear, Breakpoint Disable, or Breakpoint Enable command.

Record # 17

Bad format string. Expressions used with the Display expression, Watch, Watchpoint, and Tracepoint commands can have format specifiers set off from the expression by a comma. The valid format specifiers are d, i, u, o, x, X, f, e, E, G, c and s.

Record # 18

Badly formed type. The type information in the symbol table of the file you are debugging is incorrect. If this message occurs, please note the circumstances of the error and inform Microsoft Corporation using the Microsoft Product Assistance Request form at the back of your manuals.

Record # 19

EMM error. The debugger is failing to use EMM correctly. Please contact Microsoft Corporation using the Microsoft Product Assistance Request form at the back of your manuals.

Record # 20

Floating point error. This message should not occur, but if it does, please note the circumstances of the error and inform Microsoft Corporation, using the Microsoft Product Assistance Request form at the back of one of your manuals.

VITA

Umang R Sheth

Candidate for the Degree of

Master of Science

Thesis: FUZZY LOGIC APPLIED TO A TEXT RETRIEVAL APPLICATION

Major Field: Computer Science

Biographical:

Personal Data: Born in Ahmedabad, India, June 17, 1966, the son of Rajnikant Sheth and Niranjana Sheth.

Education: Graduated from St. Xavier's High School, Ahmedabad, India, in May, 1983; received Bachelor of Engineering Degree in Electrical Engineering from L. D. College of Engineering, Ahmedabad, India, in June, 1988; completed requirements for the Master of Science degree at Oklahoma State University in December, 1991.

Professional Experience: Assistant Lecturer, L. D. College of Engineering, India, July, 1988 to June, 1989; Research Assistant, Office of Business and Economic Research, Oklahoma State University, June, 1990 to Jan, 1991.