

Web-Based BISC Decision Support System

Gamil Serag-Eldin, Masoud Nikravesh, *BISC Program, Computer Sciences Division, EECS Department, University of California, Berkeley, CA 94720, USA*

Abstract—Most of the existing search systems’ ‘software’ are modeled using crisp logic and queries. We introduce fuzzy querying and ranking as a flexible tool allowing approximation where the selected objects do not need to match exactly the decision criteria resembling natural human behavior. The model consists of five major modules: the Fuzzy Search Engine, the Application Templates, the User Interface, the Database and the Evolutionary Computing. The system is designed in a generic form to accommodate more diverse applications and to be delivered as stand-alone software to academia and businesses.

Index Terms— Crisp logic, fuzzy querying and ranking, evolutionary computation.

I. INTRODUCTION

Searching database records and ranking the results based on multi-criteria queries is central for many database applications used within organizations in finance, business, industrial and other fields. Most of the available systems’ ‘software’ are modeled using crisp logic and queries, which result in rigid systems with imprecise and subjective processes and results. In this chapter we introduce fuzzy querying and ranking as a flexible tool allowing approximation where the selected objects do not need to match exactly the decision criteria resembling natural human behavior.

The model consists of five major modules: the Fuzzy Search Engine (FSE), the Application Templates (AT), the User Interface (UI), the Database (DB) and the Evolutionary Computing (EC). We developed the software with many essential features. It is built as a web-based software system that users can access and use over the Internet. The system is designed to be generic so that it can run different application domains. To this end, the Application Template module provides information of a specific application as attributes and properties, and serves as a guideline structure for building a new application. The Fuzzy Search Engine (FSE) is the core module of the system. It has been developed to be generic so that it would fit any application. The main FSE component is the query structure, which utilizes membership functions, similarity functions and aggregators. Through the user interface a user can enter and save his profile, input criteria for a new query, run different queries and display results. The user can manually eliminate the results he disapproves of or change the ranking according to his preferences. The Evolutionary Computing (EC) module monitors ranking preferences of the user’s queries. It learns to adjust to the intended meaning of the users’ preferences.

We present our approach with three important applications: ranking (scoring), which has been used to make financing decisions concerning credit cards, cars and mortgage loans; the process of college admissions where hundreds of thousands of applications are processed yearly by U.S. universities; and date matching as one of the most popular internet programs. Even though we implemented three applications, the system is designed in a generic form to accommodate more diverse applications and to be delivered as stand-alone software to academia and businesses.

II. MODEL FRAMEWORK

The DSS system starts by loading the application template, which consists of various configuration files for a specific application (see section 4) and initializing the database for the application (see section 6), before handling a user’s requests, see figure 1.

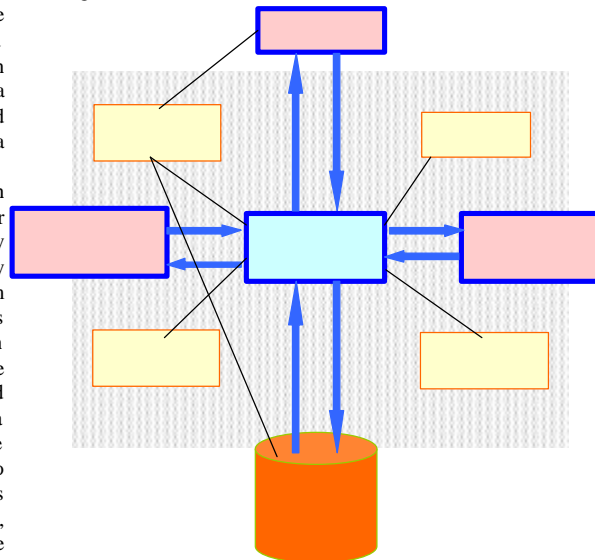


Figure 1 The BISC-DSS general framework

Once the DSS system is initialized, users can enter their own profiles in the user interface or make a search with their preferences. These requests are handled by the control unit of the system. The control unit converts user input into data objects that are recognized by the DSS system. Based on the request types, it forwards them to the appropriate modules. If

the user wants to create a profile, the control unit will send the profile data directly to the database module which stores the data in the database for the application. If the user wants to query the system, the control unit will direct the user's preferences to the Fuzzy Search Engine which queries the database (see section 3). The query results will be sent back to the control unit and displayed to the users.

III. FUZZY ENGINE

A. Fuzzy Query, search and Ranking

To support generic queries, the fuzzy engine has been designed to have a tree structure. There are two types of nodes in the tree, category nodes and attribute nodes, as depicted in figure 2. While multiple category levels are not necessary, they are designed to allow various refinements of the query through the use of the type of aggregation of the children. Categories only act to aggregate the lower levels. The attribute nodes contain all the important information about a query. They contain the membership functions for the fuzzy comparison as well as the use of the various aggregation methods to compare two values.

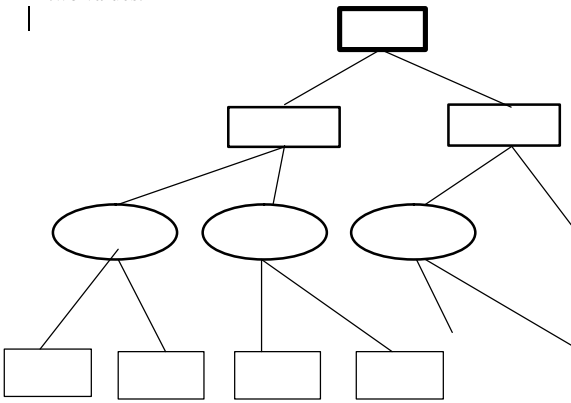


Figure 2 The Fuzzy search engine tree structure.

The attribute nodes handle the compare method slightly differently than the category nodes. There are two different ways attributes may be compared. The attribute nodes contain a list of membership functions comprising the fuzzy set. The degrees of membership for this set are passed to the similarity comparator object, which currently has a variety of different methods to calculate the similarity between the two membership vectors. In the other method, the membership vector created by having full membership to a single membership function specified in the fuzzy data object, but no membership value for the other functions.

B. Membership function

Currently there are three membership functions implemented for the Fuzzy Engine. A generic interface has been created to

allow several different types of membership functions to be added to the system. The three types of membership functions in the system are: Gaussian, Triangular and Trapezoidal. These functions have three main points, for the lower bound, upper bound and the point of maximum membership. For other functions, optional extra points may be used to define the shape (an extra point is required for the trapezoidal form).

IV. APPLICATION TEMPLATE

The DSS system is designed to work with different application domains. The application template is a format for any new application we build; it contains data of different categories, attributes and membership functions of that application. The application template module consists of two parts: 1) the application template data file specifies all the membership functions, attributes and categories of an application. We can consider it as a configuration data file for an application. It contains the definition of membership functions, attributes and the relationship between them; 2) The application template logic parses and caches data from the data file so that other modules in the system can have faster access to definitions of membership functions, attributes and categories. It also creates a tree data structure for the fuzzy search engine to transverse.

V. USER INTERFACE

It is difficult to design a generic user interface that suits different kind of applications for all the fields. For example, we may want to have different layouts for user interfaces for different applications. To make the DSS system generic while preserving the user friendliness of the interfaces for different applications, we developed the user interfaces into two parts (see figure 4).

First, we designed a specific HTML interface for each application we developed. Users can input their own profiles; make queries by specifying preferences for different attributes. Details for the DSS system are encapsulated from the HTML interface so that the HTML interface design would not be constrained by the DSS system.

The second part of our user interface module is a mapping between the parameters in the HTML files and the attributes in the application template module for the application. The input mapping specifies the attribute names to which each parameter in the HTML interface corresponds.. With this input mapping, a user interface designer can use input methods and parameter names freely.

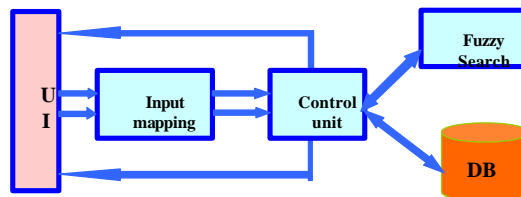


Figure 4 User interface data flow

VI. DATABASE (DB)

The database module is responsible for all the transactions between the DSS system and the database. This module handles all queries or user profile creations from the Fuzzy Engine and the Control Unit respectively. For queries from the Fuzzy Search Engine, it retrieves data from the database and returns it in a data object form. Usually queries are sets of attribute values and their associated weights. The database module returns the matching records in a format that can be

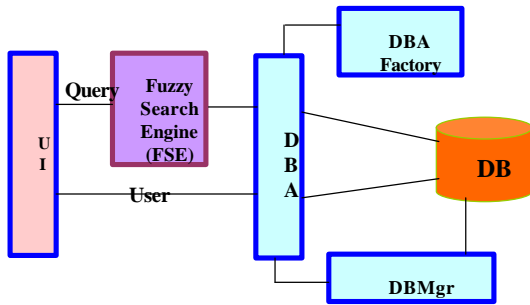


Figure 5 Database module components

manipulated by the user, such as eliminating one or more record or changing their order. To create a user profile, it takes data objects from the Control Unit and stores it in the database. There are three components in the DB module (see figure 5). 1) The DB Manager is accountable for two things: 1) setting up database connections and allocating database connections to DB Accessor objects when needed. It also supplies information to the database for authentication purposes (e.g. username, password, path to the database etc); 2) The DB Accessor Factory creates DB Accessor objects for a specific application. For example, if the system is running the date matching application, DB Accessor Factory will create DB Accessor objects for the date matching application. The existence of this class serves the purpose of using a generic Fuzzy Search Engine; 3) the DB Accessor is responsible for storing and getting user profiles to and from the database. It is the component that queries the database and wrap result from the database into data objects that are recognized by our application framework.

VII. MEASURE OF ASSOCIATION AND FUZZY SIMILARITY

As in crisp query and ranking, an important concept in fuzzy query and ranking applications is the measure of association or similarity between two objects in consideration. For example, in a fuzzy query application, a measure of similarity between two queries and a document, or between two documents, provides a basis for determining the optimal response from the

system. In fuzzy ranking applications, a measure of similarity between a new object and a known preferred (or non-preferred) object can be used to define the relative goodness of the new object. Most of the measures of fuzzy association and similarity are simply extensions from their crisp counterparts. However, because of the use of perception based and fuzzy information, the computation in the fuzzy domain can be more powerful and more complex.

Various definitions of similarity exist in the classical, crisp domain, and many of them can be easily extended to the fuzzy domain. However, unlike in the crisp case, in the fuzzy case the similarity is defined on two fuzzy sets. Suppose we have two fuzzy sets A and B with membership functions $\mu_A(x)$ and $\mu_B(x)$, respectively. The arithmetic operators involved in the fuzzy similarity measures can be treated using their usual definitions while the union and the intersection operators need to be treated specially. It is important for these operator pairs to have the following properties: (1) conservation, (2) monotonicity, (3) commutativity, and (4) associativity. It can be verified that the triangular norm (T-norm) and triangular conorm (T-conorm) (Nikravesh, 2001a; Bonissone and Decker, 1986; Mizumoto, 1989; Fagin, 1998 and 1999) conform to these properties and can be applied here. A detailed survey of some commonly used T-norm and T-conorm pairs along with other aggregation operators can be found at Nikravesh and Azvine (2002).

In many situations, the controlling parameters, including the similarity metric, the type of T-norm/conorm, the type of aggregation operator and associated weights, can all be specified based on the domain knowledge of a particular application. However, in some other cases, it may be difficult to specify a priori an optimal set of parameters. In those cases, various machine learning methods can be employed to automatically “discover” a suitable set of parameters using a supervised or unsupervised approach. For example, the Genetic Algorithm (GA) and DNA-based computing, as described in later sections, can be quite effective.

VIII. EVOLUTIONARY COMPUTING

In the Evolutionary Computing (EC) module of the BISC Decision Support System, our purpose is to use an evolutionary method to allow automatic adjusting of the user's preferences. These preferences can be seen as parameters of the fuzzy logic model in form of weighting of the used variables. These preferences are then represented by a weight vector and genetic algorithms will be used to fix them.

In the Evolutionary Computation approach, Genetic Programming, which is an extension of Genetic Algorithms, is the closest technique to our purpose. It allows us to learn a tree structure, which represents the combination of aggregators. The selection of these aggregators is included to the learning process using the Genetic Programming.

In this section, we describe the GA (Holland, 1992) and GP (Koza, 1992) application to our problem. Our aim is learning fuzzy-DSS parameters which are the weight vectors representing the user preferences associated to the variables

that have to be aggregated on the one hand, and the adequate decision tree representing the combination of the aggregation operators that have to be used on the other hand.

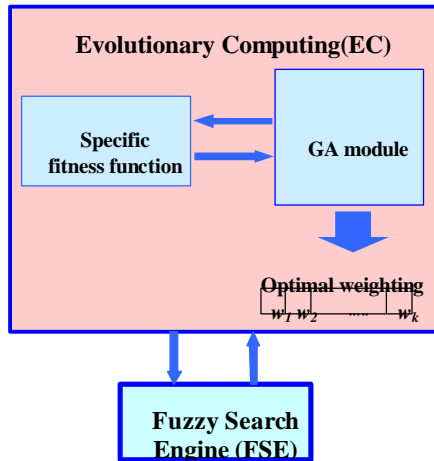


Figure 6. Evolutionary Computing Module: preferences learning.

Weight vector being a linear structure, can be represented by a binary string in which weight values are converted to binary numbers. This binary string corresponds to the individual's DNA in the GA learning process. The goal is to find the optimal weighting of the variables. A general GA module can be used by defining a specific fitness function for each application as shown in Figure 6.

Aggregators can be combined in the form of a tree structure which can be built using a Genetic Programming learning module. It consists in evolving a population of individuals represented by tree structures. The evolution principle remains the same as in a conventional GP module but the DNA encoding needs to be defined according to the considered problem. We propose to define an encoding for aggregation trees which is more complex than for classical trees and which is common to all considered applications. As shown in Figure 8, we need to define a specific encoding, in addition to the fitness function specification.

Tree structures are generated randomly as in the conventional GP. But, since these trees are augmented according to the properties defined above, the generation process has to be updated. So, we decided to randomly generate the number of arguments when choosing an aggregator as a node in the tree structure. And for the weights, we chose to generate them randomly for each node during its creation.

Concerning the fitness function, it is based on performing the aggregation operation and the root node of the tree that has to be evaluated. For the university admissions application, the result of the root execution corresponds to the score that has to

be computed for each value vector in the training data set. The fitness function, as in the GA learning of the user preferences, consists in simple or combined similarity measures. In addition, we can include to the fitness function a complementary measure that represents the individual's size which has to be minimized in order to avoid over-sized trees.

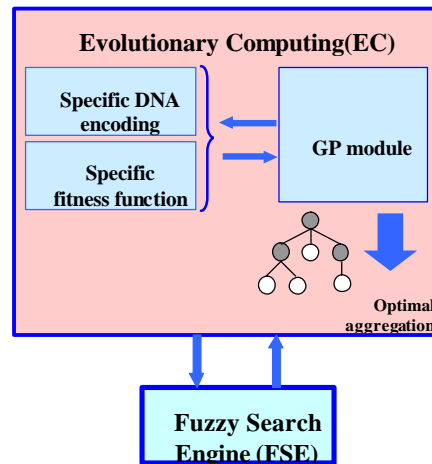


Figure 7 Evolutionary Computing Module: aggregation tree learning.

We have described the use of evolutionary computation methods for optimization problems in the BISC decision support system. It is an original idea in combining fuzzy logic, machine learning and evolutionary computation. We gave some implementation precisions for the university admissions application. We also plan to apply our system to many other applications.

IX. IMPLEMENTATION - FUZZY QUERY AND RANKING

In this section, we introduce fuzzy query and fuzzy aggregation for credit scoring, university admissions and date matching.

Credit Scoring: Credit scoring was first developed in the 1950's and has been used extensively in the last two decades. In the early 1980's, the three major credit bureaus, Equifax, Experian, and TransUnion worked with the Fair Isaac Company to develop generic scoring models that allow each bureau to offer an individual score based on the contents of the credit bureau's data. When you apply for financing, whether it's a new credit card, car or student loan, or a mortgage, about 40 pieces of information from your credit card report are fed into a model (Nikraves and Azvine 2002 and Nikraves et al. 2003). This information is categorized into the following five categories with different levels of importance (% of the score):

- Past payment history (35%)
- Amount of credit owed (30%)
- Length of time credit established (15%)

Search for and acquisition of new credit (10%)
Types of credit established (10%)

The screenshot shows a web-based interface titled "ONLINE FINANCIAL TOOLS". It features a sidebar on the left with navigation options like "Account Planning", "Account Information", "Credit Information", "Financial Information", "Admission Information", and "Registration". The main content area is divided into several sections, each with a heading and a list of input fields:

- Account Planning:** Includes fields for "Account type", "Account status", "Account type", "Account status", and "Account type".
- Account Information:** Includes fields for "Account type", "Account status", "Account type", "Account status", and "Account type".
- Credit Information:** Includes fields for "Credit type", "Credit status", "Credit type", "Credit status", and "Credit type".
- Financial Information:** Includes fields for "Financial type", "Financial status", "Financial type", "Financial status", and "Financial type".
- Admission Information:** Includes fields for "Admission type", "Admission status", "Admission type", "Admission status", and "Admission type".
- Registration:** Includes fields for "Registration type", "Registration status", "Registration type", "Registration status", and "Registration type".

Figure 8. A snapshot of the variable input for credit scoring software.

Given the factors presented earlier, a simulated model has been developed. A series of excellent, very good, good, not good, not bad, bad, and very bad credit scores have been recognized (without including history). Then, fuzzy similarity and ranking have been used to rank the new user and define his/her credit score. In the inference engine, the rules based on factual knowledge (data) and knowledge drawn from human experts (inference) are combined, ranked, and clustered based on the confidence level of human and factual support. This information is then used to build the fuzzy query model with associated weights. In the query level, an intelligent knowledge-based search engine provides a means for specific queries. Initially we blend traditional computation with fuzzy reasoning. This effectively provides validation of an interpretation, model, hypothesis, or alternatively, indicates the need to reject or reevaluate. Information must be clustered, ranked, and translated to a format amenable to user interpretation. *Figure 8* shows a snapshot of the software developed for credit scoring. To test the performance of the model, a demo version of the software is available at: <http://zadeh.cs.berkeley.edu/> (Nikravesh, 2001a).

University Admissions: Hundreds of millions of applications were processed by U.S. universities resulting in more than 15 million enrollments in the year 2000 for a total revenue of over \$250 billion. College admissions are expected to reach over 17 million by the year 2010, for total revenue of over \$280 billion.

The UC Berkeley campus admits its freshman class on the

basis of an assessment of the applicants' high school academic performance (approximately 50%) and through a comprehensive review of the application including personal achievements of the applicant (approximately 50%) (University of California-Berkeley).

At Stanford University, in addition to the academic transcript, close attention is paid to other factors such as student's written application, teacher references, the short responses and one-page essay (carefully read for quality, content, and creativity), and personal qualities.

Given the factors and general admission *criteria*, a simulated-hypothetical model (a Virtual Model) was developed. A series of excellent, very good, good, not good, not bad, bad, and very bad student given the criteria for admission has been recognized. These criteria over time can be modified based on the success rate of students admitted to the university and their performances during the first, second, third and fourth years of their education with different weights and degrees of importance given for each year. Then, fuzzy similarity and ranking can evaluate a new student rating and find its similarity to a given set of criteria.

This will provide the ability to answer "what if" questions in order to decrease uncertainty and provide a better risk analysis to improve the chance for "increased success" on student selection or it can be used to select students on the basis of "diversity" criteria. The model can be used as for decision support and for a more uniform, consistent and less subjective and biased way. Finally, the model could learn and provide the mean to include the feedback into the system through time and will be adapted to the new situation for defining better criteria for student selection.

In this study, it has been found that ranking and scoring is a very subjective problem and depends on user perception and preferences in addition to the techniques used for the aggregation process which will effect the process of the data mining in reduced domain. Therefore, user feedback and an interactive model are recommended tools to fine-tune the preferences based on user constraints. This will allow the representation of a multi-objective optimization with a large number of constraints for complex problems such as credit scoring or admissions. To solve such subjective and multi-criteria optimization problems, GA-fuzzy logic and DNA-fuzzy logic models are good candidates.

To solve such subjective and multi-criteria optimization problems with a large number of constraints for complex problems such as university admissions, the BISC Decision Support System is an excellent candidate.

Date Matching: The main objective is to find the best possible match in the huge space of possible outputs in the databases using the imprecise matching such as fuzzy logic concepts, by storing the query attributes and continuously refining the query to update the user's preferences. We have also built a Fuzzy Query system, which is a Java application that sits on top of a database.

With traditional SQL queries (relational DBMS), one can select records that match the selection criteria from a database. However, a record will not be selected if any one of the

conditions fails. This makes searching for a range of potential candidates difficult.

In this program, one can basically retrieve all the records from the database, compare them with the desired record, aggregate the data, compute the ranking, and then output the records in the order of their rankings. Retrieving all the records from the database is a naïve approach because with some preprocessing, some very different records are not needed from the database. However, the main task is to compute the fuzzy rankings of the records so efficiency is not the main concern here.

The major difference between this application and other date matching system is that a user can input his hobbies in a fuzzy sense using a slider instead of choosing crisp terms like "kind of" or "love it". These values are stored in the database according to the slider value.

X. CONCLUSION

In this study, we introduced fuzzy query, fuzzy aggregation, evolutionary computing and the BISC decision support system as an alternative for ranking and predicting the risk for credit scoring, university admissions, and several other applications, which currently utilize an imprecise and subjective process. The BISC decision support system key features are 1) intelligent tools to assist decision-makers in assessing the consequences of decision made in an environment of imprecision, uncertainty, and partial truth and providing a systematic risk analysis, 2) intelligent tools to be used to assist decision-makers answer "what if" questions examine numerous alternatives very quickly and find the value of the inputs to achieve a desired level of output, and 3) intelligent tools to be used with human interaction and feedback to achieve a capability to learn and adapt through time. In addition, the following important points have been found in this study: 1) no single ranking function works well for all contexts, 2) most similarity measures work about the same regardless of the model, 3) there is little overlap between successful ranking functions, and 4) the same model can be used for other applications such as the design of a more intelligent search engine which includes the user's preferences and profile (Nikravesh, 2001a and 2001b).

ACKNOWLEDGMENT

Funding for this research was provided by the British Telecommunication (BT) and the BISC Program of UC Berkeley.

REFERENCES

1. Fagin R. (1998) Fuzzy Queries in Multimedia Database Systems, Proc. ACM Symposium on Principles of Database Systems, pp. 1-10.
2. Fagin R. (1999) Combining fuzzy information from multiple systems. J. Computer and System Sciences 58, pp 83-99.
3. J. R. Koza, Genetic Programming : On the Programming of Computers by Means of Natural Selection, Cambridge, Mass. : MIT Press, USA 1992, 819 pages.
4. John H. Holland. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. MIT Press, 1992. First Published by University of Michigan Press 1975.
5. Mizumoto M. (1989) Pictorial Representations of Fuzzy Connectives, Part I: Cases of T-norms, T-conorms and Averaging Operators, Fuzzy Sets and Systems 31, pp. 217-242.
6. Nikravesh M. (2001a) Perception-based information processing and retrieval: application to user profiling, 2001 research summary, EECS, ERL, University of California, Berkeley, BT-BISC Project. (<http://zadeh.cs.berkeley.edu/> & <http://www.cs.berkeley.edu/~nikravesh/> & <http://www-bisc.cs.berkeley.edu/>).
7. Nikravesh M. (2001b) Credit Scoring for Billions of Financing Decisions, Joint 9th IFSA World Congress and 20th NAFIPS International Conference. IFSA/NAFIPS 2001 "Fuzziness and Soft Computing in the New Millennium", Vancouver, Canada, July 25-28, 2001.
8. Masoud Nikravesh and Ben Azvine (2002), Fuzzy Queries, Search, and Decision Support System, Journal of Soft Computing, Volum 6, # 5, August 2002.
9. Masoud Nikravesh, B. Azvine, R. Yagar, and Lotfi A. Zadeh (2003) "New Directions in Enhancing the power of the Internet", Editors:, to be published in the Series Studies in Fuzziness and Soft Computing, Physica-Verlag, Springer (August 2003)