Fuzziness as a Model of User Preference in Semantic Web Search

Veronika Vaneková[1] Peter Vojtáš[2]

1.Pavol Jozef Šafárik University in Košice, Slovakia
2.Charles University in Prague, Czech Republic
Email: veronika.vanekova@upjs.sk, peter.vojtas@mff.cuni.cz

Abstract— *Web 2.0 is based on personalized access to imprecise and incomplete information from heterogeneous sources. In this paper we present a web-based system enabling preference-based search. We use modified Fagin's model of fuzzy preferences based on aggregation of attribute preferences. Aggregation is generated by user ranking of objects. Our contributions are twofold. First we present a formal model and implementation suitable for efficient preferential search. Second we provide several fuzzy measures to evaluate experimental results of our system.*

Keywords— web search, fuzzy logic, fuzzy systems, user preference

1 Introduction

From its birth in 1965, fuzzy logic has spread to many areas. It has influenced logic, databases, neural networks, expert systems, natural language processing and other fields of both theoretical and practical research. Fuzzy logic allows us to model imprecise, vague or uncertain information and to specify vague requirements. Any theory or application can possibly be fuzzified. However, we should first consider the purpose, especially in case of applications. The important question is how much fuzziness we really deal with and where it comes from.

In this paper we propose one such application of fuzzy logic: we use fuzzy sets to represent a notion of user preference in the process of web search. Web search tasks are recently changing from finding a document to solving specific problem for specific user. A typical problem is to buy some item online, make a flight reservation, find a transportation line, apply for a job. Most of these problems actually require searching for objects from some *domain* (e.g. flights, books, notebooks, job offers). The search depends on various attributes of objects (e.g. price, destination, date of departure, airlines and class in the domain of flights). Fulltext search often fails to support such tasks.

Therefore we search data from one arbitrary, but fixed domain. Data is extracted from heterogeneous sources on the web and stored in a standard database or ontology. Our extraction tool does not create fuzzy data. Although the formal model can be easily extended to handle fuzzy data, we focus on the situations where the main source of fuzziness is not in the data itself, but in the imprecise way of human thinking and specifying requirements. Most web users do not know precisely what they want to find, but they are able to distinguish between better and worse objects. Hence we allow users to rate objects and we use inductive learning tool [1] to adjust user preferences. But users also have an opportunity to specify their preferences precisely by means of a special graphical interface. The best objects can be retrieved with top-k threshold algorithm (see [2, 3]).

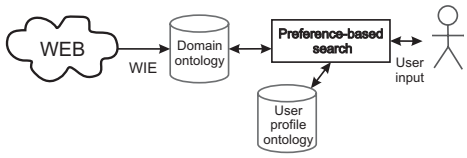

Figure 1: Overview of User-Dependent Search System.

This paper investigates how fuzzy logic can be used to make web search easier and more suitable for imprecise requirements. We compare our approach with other systems and formal models and point out the features necessary for Web 2.0 applications [4] – modeling vague preferences, dealing with huge amounts of data and ordering search results by degree of preference. We also analyze user data from our system with various measures of fuzzy similarity, equality, correlation and we compare the results.

2 Formal Models

Throughout the rest of this paper we will use the domain of job offers which is used also in the implementation. But the formal models described below are suitable for other domains as well. We tested the whole system or its parts in domain of cars, flats and investments.

Web search process is divided into two phases. The first phase is called *WIE* – web information extraction. Information is extracted from HTML pages, each of which contains data for a single object (job offer). Such pages are often generated by server side scripts, so that their structure and HTML tags are similar. The extraction tool [5] compares the page structure with *diff* algorithm and analyzes HTML tags with regular expressions. Differing parts of page sources contain relevant information, most often various attributes of domain objects. We use heuristics (like regular expression patterns for HTML structure) to avoid irrelevant information. Extracted crisp data is stored in domain ontology of job offers. Data repository is the only point that connects extraction phase with searching phase (see Fig. 1). Therefore it is easy to use other extraction tools or search in some existing standard database as well. It is also possible to have fuzzy domain data. Some extraction tools consider reliability of web sources, quality of source data or other conditions to determine fuzzy degree of extracted objects. Our formal model could be extended to consider this type of data as well. Full description of the extraction phase is beyond the scope of this paper, for more information see [6, 7, 5].

ISBN: 978-989-95079-6-8

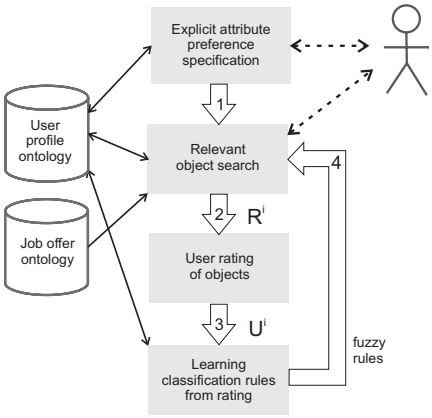

Figure 2: Flow Diagram of User-Dependent Search.

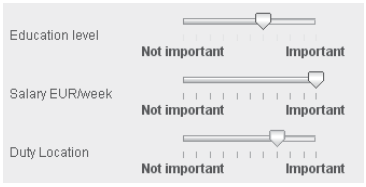

Figure 3: Interface for Attribute Importance.

The second phase, preference-based search, is detailed on Fig. 2. The task of job offer search can be repeated. The system reads user preferences (specified directly) or user feedback (previous search results rated in five-degree scale), updates stored preferences and finds new results. Every such *search cycle* provides more information about the user and helps the system to find more exact preferences.

In the rest of this section we will describe all phases of user-dependent search from Fig. 2 in more detail, together with underlying formal models. We especially emphasize the sources of fuzziness and their influence on the formal model.

2.1 Explicit Preference Specification

Current web portals often provide either a text field for fulltext search or some form to specify required attribute values. Job offer portals usually allow applicants to specify at least place, job position and contract type (i.e. permanent or temporary). But it is impossible to express things like "I prefer York to Birmingham" or "my decision depends on salary more than on location". Our system allows for vague user preferences and provides a suitable graphical interface.

First, the user is presented a list of all *attributes* from the domain of job offers that could influence his decision (like salary, benefits, required education level, required experience, place, job position, contract type, start date). The interface displays a slider marked with "Not important" and "Important" for each attribute (Fig. 3). User can drag the slider to specify how much he cares about the corresponding attribute. The settings from Fig. 3 will produce weighted average

$$@^U = \frac{6 \cdot \mu^U_{education\_level} + 10 \cdot \mu^U_{salary} + 7 \cdot \mu^U_{location}}{23}$$

Weighted averages and other types of aggregation will be described below.

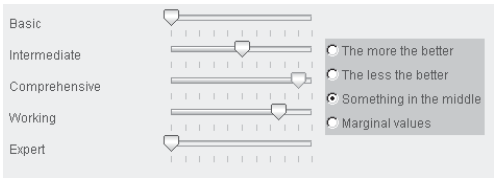

Figure 4: Interface for Attribute Preference.

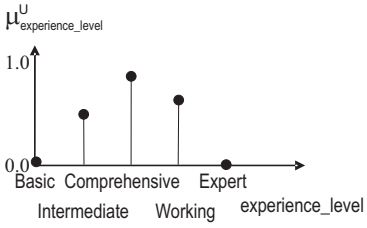

Figure 5: Preference to experience level, acquired from user input.

The next step is to specify preferred attribute values (only for attributes that are marked as "Important" to some non-zero degree). Now the graphical interface contains one slider for every attribute value. Fig. 4 shows the input interface for required experience level. The user prefers those job offers that require comprehensive experience (possibly because he has such experience in the profession).

Formally, user preference related to one particular attribute is a typical example of fuzzy subset. We consider a domain Δ of objects (job offers) and a finite set of domain attributes $\{A_1, \ldots, A_n\}$. Every attribute A_i is a function $A_i : \Delta \longrightarrow \Delta_{A_i}$, where Δ_{A_i} is a set of possible attribute values. Then *attribute preference* for user U and attribute A_i is defined as $\mu^U_{A_i} = \Delta_{A_i} \longrightarrow [0,1]$. Thus $\mu^U_{A_i}$ defines a fuzzy subset of Δ_{A_i}, which is usually a finite set with some natural element ordering.

Fuzzy set $\mu^U_{experience\_level}$ generated from user input on Fig. 4 is shown on Fig. 5. Job offers that require basic or expert experience are unacceptable, while job offers with comprehensive experience are the most preferred. Every attribute preference defines non-strict linear ordering of all objects based on their membership values. The ordering generated by the fuzzy set $\mu^U_{experience\_level}$ would not preserve original ordering of attribute values. Note that some distinct attribute values (e.g. basic and expert experience) have equal membership values. Such objects are *indiscernible* with respect to attribute $experience\_level$.

The most suitable membership functions to represent real user preferences are monotonic and trapezoidal. Some attributes are preferred similarly by all users (e.g. high salary is better than low), however, other attribute preferences may differ significantly (e.g. preference for required language level). These *preference types* can be seen on the right side of Fig. 4.

If we consider more than one attribute, we need to determine the final preference degree for every object, based on attribute preferences. User requirements are often conflicting so that the orderings are different, sometimes even reverse. Therefore we use *aggregation function* to acquire overall ordering. Aggregation function is n-ary monotone function $@^U : [0,1]^n \longrightarrow [0,1]$.

ISBN: 978-989-95079-6-8

Let us consider an object $x \in \Delta$ with attribute values $A_1(x), \ldots, A_n(x)$ and attribute preferences $\mu^U_{A_1}(A_1(x)) = p_1, \ldots, \mu^U_{A_n}(A_n(x)) = p_n$. The global preference degree of x will be $@^U(p_1, p_2, \ldots, p_n)$. A typical aggregation function is weighted average which reflects the idea that some attributes can be more important than the others. Our system acquires the weights from the interface shown on Fig. 3. We will describe other types of aggregation functions in Section 2.3.

2.2 Relevant Object Search

After storing user preferences the system proceeds to the next phase - relevant object search. It is a modification of Fagin's threshold algorithm [3] called *top-k* [2], based on the idea that users are interested only in a few best results from potentially huge dataset and that different users can have different attribute preferences.

Top-k uses a preprocessing phase to generate lists of objects ordered by attribute values. A typical structure for indexing is a B+ tree which allows us to follow records (attribute values) from highest to lowest or vice versa. User preference can define a different ordering of objects, but it can be simulated as well. If the user specified a trapezoidal membership function ("Something in the middle" preference), we start traversing values in both directions from the middle and we merge two non-increasing lists into one non-increasing list. Thus we gain a list of objects ordered from most preferred to least preferred. (The original threshold algorithm considered only one fixed ordering for each attribute.)

If our domain contains attributes $A_1, \ldots, A_n$, we prepare n ordered lists $L_1, \ldots, L_n$. Lists can be stored in heterogeneous repositories, or even acquired from web services. Top-k algorithm uses sorted access to the lists, reads attribute values and calculates preference values. One object can have different position in each ordered list, so some attribute values will be unknown in the meantime. After reading all attribute values of some object x, top-k algorithm calculates aggregation function to determine overall preference value of the object. A threshold value is determined to see if any object with some unknown values has a chance to be more preferred than x. If not, object x is written on the output.

This algorithm is very efficient for complex preferential search. The preprocessing phase causes that we can find k best objects without traversing the whole dataset. The output is an ordered list of k best objects (according to given preferences) together with overall preference values. Despite the fact that source data is crisp, we obtain fuzzy results similar to "object $x1$ is preferred by user U to degree $0,8$". This fuzziness originates from vague user preference representation and it is reflected in the results.

2.3 Learning Classification Rules from Ratings

Although the preference model is fuzzy, it is intended to represent real user preferences as precisely as possible. If they do not correspond, the user obtains different results, or results in wrong order. The graphical interface gives user a chance to rate the results in 5-degree scale (bad, poor, average, good, excellent) as can be seen on Fig. 6. Degrees can also be represented as percents, stars, smilies, etc. Then we use ordinal classification [8] to find a new aggregation function. As we showed in [9], it is sufficient to learn an aggregation function,

Figure 6: Interface to view and rate search results.

provided that types of attribute preferences did not change (e.g. from "The more the better" to "The less the better"). The new aggregation function has a form of *fuzzy rules*.

Fuzzy rules are well-known from fuzzy controllers [10], fuzzy inference systems [11] and many other application areas. A fuzzy rule consists of antecedent and consequent. *Antecedent* is typically a (fuzzy) conjunction or disjunction of fuzzy condition clauses. *Consequent* can be some action (in case of fuzzy controllers) or a fuzzy predicate (in case of fuzzy inference systems).

controller rule: IF temperature IS cold THEN heater IS high

inference system rule: IF wind IS strong AND barometer IS falling THEN forecast IS bad

Different types of rules look syntactically similar, but they differ in the semantics of consequent. In the first case the consequent "heater IS high" means that the heater device will be *set* to high level. In the latter case the consequent "forecast IS bad" means that the variable forecast will be assigned a fuzzy value bad, but no physical action will take place. Condition clauses contain crisp variables (temperature, wind, barometer) and fuzzy sets (cold, strong, falling) in both cases. Connectives AND, OR are fuzzy t-norms and t-conorms, IF-THEN implications are also fuzzified. There is usually a larger set of rules which are processed, their outputs are combined (aggregated) in some way into a single fuzzy set. It is then defuzzified to determine crisp output value. The most common defuzzification method is *centroid calculation*, which returns the geometric center of area under the fuzzy membership function.

Our usage is more similar to fuzzy inference systems because we use fuzzy rules to find a preference value, not to control some electronic device. It is somewhat simplified to reflect our preference model. The following example shows a set of classification rules. We chose fuzzy rules as aggregation method because they are human-understandable and very close to natural way of thinking.

IF $\mu^U_{salary}(salary(x)) \geq 0.7$ AND $\mu^U_{place}(place(x)) \geq 0.6$ AND $\mu^U_{position}(position(x)) \geq 0.5$ THEN $good^U(x) \geq 0.8$

IF $\mu^U_{experience\_level}(experience\_level(x)) \geq 0.3$ AND $\mu^U_{salary}(salary(x)) \geq 0.6$ THEN $good^U(x) \geq 0.6$

It is easy to see some differences compared to fuzzy rules mentioned above. Our fuzzy rules are induced from a small set of rated objects, so they are more specific. Condition clause $\mu^U_{salary}(salary(x)) \geq 0.7$ can be written more intuitively as $salary(x)$ IS good ≥ 0.7. We use arbitrary fuzzy sets $\mu^U_{A_i}$ instead of linguistic variables like "cold".

ISBN: 978-989-95079-6-8

Our condition clauses use α-cut defuzzification and thus our conjunctions and implications are crisp. If object x satisfies the antecedent, then the aggregation function $good^U$ will be greater or equal to the value specified in the consequent. If some object satisfies more antecedents, we choose the greatest output value. If it does not satisfy any rule antecedent, the result will be set to 0. A set of rules makes up a monotone aggregation function. An object y with all attribute preference values greater or equal to object x will be "good" to greater or equal degree than x.

3 Implementation

The system for user-dependent web search is implemented and available on http://x64.ics.upjs.sk:8080/nazou/. It is a web application created in Java with Wicket[1] framework. Java servlets run under Tomcat application server. The server also supports Sesame[2] database to store both our ontologies, i.e. job offer ontology and user preference ontology (see Fig. 2). User preference ontology schema is fixed, while domain ontology is arbitrary (see [12, 9] for details).

The application is divided into relatively independent tools: top-k search, inductive learning, and user preference acquisition called UPreA. UPreA provides user preferences for other tools, processes user inputs and manages all changes in user profile. It communicates with Sesame database through Java API to store and update user information.

4 System and Model Evaluation

Our goal is to experiment with the implemented system and to find out if our model is a suitable representation of real user preferences. As our model is fuzzy, we are interested in fuzzy degree of correspondence between our results and some "ideal" results. Analysis of experiments is further complicated by the fact that user preferences are evolving in time and user inputs can be inconsistent.

We anticipate vague user preferences and therefore we use inductive learning to adjust the aggregation function. Our preferences define an ordering of all objects from the domain, while user ratings define another ordering of some small subset of objects. Fig. 2 shows orderings that are generated in search cycles: R^i is the ordering of search results and U^i is the ordering specified by user ratings. We explore the correspondence of R^i and U^i. If these orderings are similar, the user must have assigned higher ratings to objects from the front of the result list and therefore our preference model reflected real user preferences well. Throughout the rest of this section we assume that R^i and U^i are permutations of objects $\{o_1, \ldots, o_n\}$. Also note that results R^0 are retrieved with user-defined aggregation function (weighted average), while other results R^i, $i \geq 1$ are retrieved with aggregation (fuzzy rules) learned from previous ratings.

The main problem is how to determine the correspondence of two fuzzy subsets. We present several measures below and compare the results. We analyse how R^i corresponds with U^i in six successive search cycles. In fact, some users stopped searching after the first or the second cycle, so the number of analyzed ordered lists is slightly decreasing. Data for this analysis (search results R^i and user ratings U^i) comes from real users who tested the implemented system.

Table 1: Correlations of result ordering and user evaluation.

Cycle	0.	1.	2.	3.	4.	5.
τ-correlation	0.73	0.77	0.79	0,77	0,81	0,81
Result lists	174	134	102	53	26	15

Table 2: Weighted similarity of result ordering and user evaluation.

Cycle	0.	1.	2.	3.	4.	5.
similarity	0.53	0.41	0.39	0,56	0,61	0,64

4.1 Correlation of Orderings

The first possibility is Kendall τ-correlation coefficient [13]. It is defined as $\tau = \frac{n_c - n_d}{\frac{1}{2}n(n-1)} - 1$, where n_c is the number of concordant pairs and n_d is the number of discordant pairs in both ratings. A pair of elements is discordant if their order is reversed. We modified this formula to $\tau = \frac{2P}{\frac{1}{2}n(n-1)}$, so that results belong to interval $[0, 1]$ to conform with our fuzzy model. However, in this case only results above 0.5 mean a positive correlation.

Kendall τ-correlation originally deals with strictly ordered lists. However, in our case many objects from R^i or U^i can share the same fuzzy value. We distinguish ties according to the (strict) order in which they are presented to user. This order in fact depends on indexing methods used in top-k search.

4.2 Weighted order similarity

If the order of some objects change, τ-correlation decreases evenly, regardless of the position where the discordance arose. We can also assume that the position of discordance is important – discordance on the 1^{st} and the 2^{nd} position is more significant than discordance on the 8^{th} and the 9^{th} position. This idea is reflected in *weighted order similarity*. We choose a decreasing vector of weights such that differences between adjacent values are also decreasing, e.g. $w = \langle 20, 15, 11, 8, 6, 5, 4, 3, 2, 1\rangle$ for $n = 10$. Weighted order coefficient is defined as $\mathcal{W} = 1 - \sum_{j=1}^{n} \left| w_{p_j} - w_{q_j} \right|$ where p_j is a sequence number of R^i_j and q_j is a sequence number of U^i_j. Note that this coefficient does not consider all pairs of elements as τ-correlation, only the sequence number of the same element in both orderings.

4.3 Fuzzy Equality

Another suitable measure is *fuzzy equality* [14], which is stronger condition than similarity. In this case we consider fuzzy aggregation values $@^U$, not only orders. We define fuzzy equality of fuzzy sets μ_A, μ_B as: $\equiv^\bullet (\mu_A, \mu_B) = \inf_{x\in\Delta}\{\vee^\bullet(\rightarrow^\bullet (\mu_A, \mu_B), \rightarrow^\bullet (\mu_B, \mu_A))\}$. It means that μ_A, μ_B are equal if $\mu_A(x)$ implies $\mu_B(x)$ and vice versa, for all x from the domain. We can choose different types of fuzzy implication and conjunction, namely Łukasiewicz, Gödel and product. The universal quantifier is always represented as infimum for all elements from the domain.

Finally, we compare results of these fuzzy measures on Fig. 7. τ-correlation and weighted similarity seem to be the most

[1] http://wicket.apache.org/
[2] http://www.openrdf.org/

ISBN: 978-989-95079-6-8

Table 3: Fuzzy equality of result ordering and user evaluation.

Cycle	0.	1.	2.	3.	4.	5.
$\equiv^{\bullet}_{L}$	0.19	0.41	0.32	0,35	0,42	0,42
$\equiv^{\bullet}_{G}$	0.23	0.22	0.14	0,20	0,22	0,24
$\equiv^{\bullet}_{p}$	0.17	0.28	0.16	0,23	0,31	0,27

Table 4: τ_R-correlation of result lists R^i, R^{i+1}.

i	0	1	2	3	4
$\lvert R^i \cap R^{i+1} \rvert$	4,50	6,21	5,95	4,88	5,52
$\tau_{\cap}$	0,65	0,80	0,76	0,63	0,76
τ_R	0,53	0,67	0,64	0,56	0,64

suitable measures. Fuzzy equalities impose too strict conditions on fuzzy values. We do not expect result lists and user ratings to be totally equal because we learn new preferences from their difference. Correlation increases most significantly between 1^{st} and 2^{nd} cycle where we start to use fuzzy rules. So experiments proved that fuzzy rules are suitable representation of user preference.

Another interesting point is that τ-correlation and weighted similarity have the same trends up to fifth cycle, while all fuzzy equalities have the same trends from second to fifth cycle. Łukasiewicz equality has the greatest results and Gödel equality has the smallest results because the corresponding fuzzy implications also have this feature.

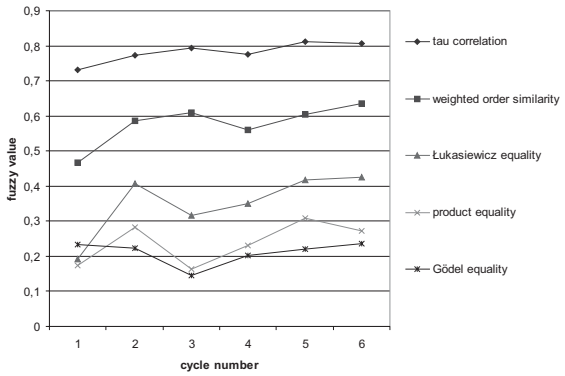

Figure 7: Comparison of different fuzzy measures.

4.4 Correlation of Subsequent Result Lists

We are also interested in the correlation of k-tuples R^i, R^{i+1}. If R^i and R^{i+1} have empty intersection, this correlation will be 0. In case of non-empty intersection, we can compute usual τ-correlation coefficient of the common elements $\tau_{\cap}$ and multiply the result with $\frac{\lvert R^i \cap R^{i+1} \rvert}{n}$. Thus we obtain τ_R. Average results for the first 5 cycles are shown in Table 4.

5 Related Work

In this section we present a short survey of similar preference models and searching methods. Many searching methods allow user to select desired attribute values, but they mostly produce a conjunctive query from user requirements. Our model is more general because aggregation functions are generalization of both (fuzzy) conjunctions and disjunctions.

A similar problem is addressed by *fuzzy multicriteria decision making* [15, 16]. It supports decision based on various attributes called *criteria*. Attribute preferences are usually represented as *fuzzy preference relations* – binary relations defined for every two-tuple of objects. Preference relation defines a degree to which one object is better than another object, based on the corresponding attribute. Other models of attribute preference mentioned in [16] are *utility functions* (identical with our fuzzy functions) and crisp *preference orderings* of objects. Aggregation is defined as *OWA operator* [17]. It is ordered weighted average associated with a normalized weight vector $w = (w_1, \ldots, w_n)^T$. Note that weight w_i is not connected with preference value of i^{th} attribute, but with i^{th} greatest preference value. This operator produces one "collective" preference relation from attribute preferences.

The *main difference* is that multicriteria decision is mainly designed for a small number of objects (alternatives). Each preference relation requires quadratic space w.r.t. number of objects. Decision making process consists of computing aggregated preference relation and transforming this relation to a global ranking of objects. Efficiency issues are left to implementations. Our model is closely connected with efficient top-k searching method which allows finding best objects without computing aggregation for all of them and sorting the results. Advantages of fuzzy multicriteria decision include support of sophisticated aggregation methods like Choquet integral and various modifications of OWA operator.

Another method to find interesting objects according to more attributes is based on *Skyline* [18]. Every attribute forms one dimension and every object is represented as a point in multi-dimensional space. Skyline is defined as a set of points which are not *dominated* by any other point (Pareto optimal), which means that no other point is equally good or better in all dimensions and strictly better at least in one dimension. A special Skyline clause can be integrated to relational database systems. Efficient methods of finding Skyline are proposed in [18, 19], including partitioning, indexing with B-trees and searching with top N algorithm (similar to top-k).

Compared to our system, Skyline produces a set of *unordered*, potentially interesting objects. Despite the similar searching methods, results can be different from our top-k results. First object o returned by top-k always belongs to Skyline set, but not necessarily the following objects, which can be in fact dominated by o, but still very close to it. Skyline also does not support attribute preferences other than simple "The more the better" or its reverse for ordinal attributes, thus it assumes the same attribute preferences for all users.

Another similar approach is *rank-aware querying* [20]. It uses general aggregation (scoring) functions to compute object ranking and returns a list of results ordered by ranking. Aggregations and new rank-join operators are integrated into relational database system. Rank-join operator performance is optimized by transforming a query into an execution plan and finding the most efficient plan. As in the case of Skyline operator, rank-aware querying does not support various attribute preferences or fuzzy rules. Simple preferences like "The less the better" can be simulated with scoring functions.

All-purpose fuzzy database system *FSQL Server* is proposed in [21]. It supports attribute preferences (called *linguistic labels*) for both ordered and unordered attribute do-

ISBN: 978-989-95079-6-8

mains. It is possible to define special fuzzy connectives to aggregate the results, but weighted average and fuzzy rules are not supported. In addition to fuzzy features needed in our system, FSQL supports fuzzy comparison, fuzzy constants, special values and α-cuts (fulfillment thresholds). FSQL is intended to support all common fuzzy features, while our system is designed for a single purpose of preferential web search.

From theoretical point of view, *rough set theory* [22] also has some common ideas with our model. It uses conditional attributes and decision attributes (similar to our user rating). Objects with the same attribute values belong to the same equivalence class and they are called *indiscernible* by those attributes. However, a set that contains one element from an equivalence class and does not contain another object from the same equivalence class, cannot be defined exactly (using only available attributes). Such set is called *rough set* and it can be defined with help of lower and upper approximation.

The main goal of rough set theory is to synthesize *decision rules* to determine the value of decision attribute from conditional attributes. The main difference is that a rough set *does not generate any ordering*, we can only distinguish objects inside the set from "boundary" objects.

6 Conclusions

Some approaches mentioned in the previous section are focused on theoretical support of decision process, some deal more with efficiency and optimization. We address both issues to achieve a clear-cut formal model and efficient implementation. Our main purpose is to enable user-specific web search in one arbitrary domain. All parts described in section 2 are important for personalized search: web information extraction, interface for explicit preference specification, top-k search, user rating and inductive learning. To the best of our knowledge, other fuzzy preference systems do not address all these phases to connect web and user.

So our approach has the following contributions:

- *Formal model* – fuzzy sets and aggregation as a model of user preference, suitable for preferential search.
- *Expressive power* – arbitrary fuzzy sets and fuzzy rules are strong tools to represent preferences.
- *Implementation* – web-based system implements specified functionality.
- *Model evaluation* – various fuzzy measures are used to analyze the results.

Acknowledgment

Partially supported by Czech projects 1ET100300517, MSM-0021620838 and Slovak project VEGA 1/0131/09.

References

[1] Tomáš Horváth. A model of user preference learning for content-based recommender systems. *Accepted for publication to COMPUTING AND INFORMATICS*.

[2] Peter Gurský and Peter Vojtáš. On top-k search with no random access using small memory. *Proceeding of ADBIS*, pages 97–111, 2008.

[3] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. *PODS '01: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 102–113, 2001.

[4] John Musser and Tim O'Reilly. *Web 2.0 Principles and Best Practices*. O'Reilly Radar Reports, 2006.

[5] D. Maruščák, R. Novotný, and P. Vojtáš. Unsupervised structured web data and attribute value extraction. *Proceedings of Znalosti*, 2009.

[6] Robert Baumgartner, Sergio Flesca, and Georg Gottlob. Visual web information extraction with lixto. pages 119–128, 2001.

[7] Eugene Agichtein. Web information extraction and user modeling: Towards closing the gap. *IEEE Data Eng. Bull.*, 29(4):37–44, 2006.

[8] Tomáš Horváth and Peter Vojtáš. Ordinal classification with monotonicity constraints. *ICDM 2006*, pages 217–225, 2006.

[9] Peter Gurský, Tomáš Horváth, Jozef Jirásek, Stanislav Krajči, Róbert Novotný, Veronika Vaneková, and Peter Vojtáš. Knowledge processing for web search - an integrated model. *Studies in Computational Intelligence (vol. 78)*, pages 95–104, 2007.

[10] Michio Sugeno. *Industrial Applications of Fuzzy Control*. Elsevier Science Inc., New York, NY, USA, 1985.

[11] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *Int. J. Hum.-Comput. Stud.*, 51(2):135–147, 1999.

[12] Peter Gurský, Veronika Vaneková, and Jana Pribolová. Fuzzy user preference model for top-k search. *Fuzzy Systems, 2008 (IEEE World Congress on Computational Intelligence)*, pages 1606–1612, 2008.

[13] Maurice Kendall and Jean D. Gibbons. *Rank Correlation Methods*. A Charles Griffin Title, 5 edition, 1990.

[14] Bernard De Baets and Radko Mesiar. Metrics and t-equalities. *Journal of Mathematical Analysis and Applications*, 267(2):531 – 547, 2002.

[15] Ronald R. Yager. Extending multicriteria decision making by mixing t-norms and owa operators: Research articles. *Int. J. Intell. Syst.*, 20(4):453–474, 2005.

[16] F. Chiclana, F. Herrera, and E. Herrera-viedma. Integrating three representation models in fuzzy multipurpose decision making based on fuzzy preference relations. *Fuzzy Sets and Systems*, 97:33–48, 1998.

[17] Ronald R. Yager and Janusz Kacprzyk, editors. *The ordered weighted averaging operators: theory and applications*. Kluwer Academic Publishers, 1997.

[18] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. The skyline operator. *Proceedings of the 17th International Conference on Data Engineering*, pages 421–430, 2001.

[19] Jan Chomicki, Parke Godfrey, Jarek Gryz, and Dongming Liang. *Intelligent Information Processing and Web Mining*, volume 31/2005 of *Advances in Soft Computing*, chapter Skyline with Presorting: Theory and Optimizations, pages 595–604. Springer-Verlag, 2005.

[20] Ihab F. Ilyas, Rahul Shah, Walid G. Aref, Jeffrey Scott Vitter, and Ahmed K. Elmagarmid. Rank-aware query optimization. *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 203–214, 2004.

[21] Jose Galindo. *Fuzzy Databases: Modeling, Design, and Implementation*. IGI Publishing, 2006.

[22] Lech Polkowski and Andrzej Skowron. Rough sets: A tutorial. *Rough Fuzzy Hybridization: A New Trend in Decision-Making*, 2000.

ISBN: 978-989-95079-6-8