Knowledge Processing for Web Search – An Integrated Model

P. Gurský, T. Horváth, J. Jirásek , S. Krajči, R. Novotný, V. Vaneková, P. Vojtáš,

P.J. Šafárik University, Charles University, Czech Academy of Science
{peter.gursky, tomas.horvath, veronika.vanekova, stanislav.krajci, robert.novotny, jozef.jirasek}@upjs.sk, peter.vojtas@mff.cuni.cz

Abstract. We propose a model of a middleware system enabling personalized web search for users with different preferences. We integrate both inductive and deductive tasks to find user preferences and consequently best objects. The model is based on modeling preferences by fuzzy sets and fuzzy logic. We present the model-theoretic semantic for fuzzy description logic **f**-EL which is the motivation of creating a model for fuzzy RDF. Our model was experimentally implemented and integration was tested.

Keywords: middleware, fuzzy DL, fuzzy RDF, relevant objects, user preferences

1. Introduction and motivation

One of the main roles of semantic web is to enable automatic access to web resources and services to be easily used by middleware engines or agents. Our research leads to the model of a middleware system permitting users to search objects of one domain but from heterogeneous sources. In this paper we present different approaches to solve the main task of such a system – return to users the best objects relative to their preferences. Typical objects we want to search are hotels, job offers, papers, pictures, books, presentations, conferences etc.

Let us consider the following example: Imagine a user looking for a hotel which has *good price*, *good distance from an airport* and has *good equipment in rooms*.

The meaning of goodness (preference) can be different for each user (or group of users). For the price of hotels, one user could prefer cheap hotels (student), second prefers expensive hotels (manager) and the other one prefers middle price (professor). The natural meaning of this ordering is that the user determines the relations "'better" or "worst" between two values of a given property. For each such property, user has a notion about the ordering of objects based on real value of property from an attribute domain. We call these orderings of particular attribute domains the *user local preferences*. The global preference will be modeled by fuzzy aggregation operators.

2. Web search

In our system different users can retrieve different answers for the same question. Searching is based on user preferences that can be created during current and previous sessions as well.

For the first contact with a new user we can assign him or her to a user type. We can consider several user types for a concrete domain. The suitable user type identification at the time of first contact requires huge testing with real users and is the aim of our future research. We assume that it can be done by a form with easy questions like gender, job position, age of children etc.

To identify suitable objects for user we need to have his local and global preferences. We can guess them by user type profile but we need to allow user to refine the profile to correspond his real preferences. For many users, it is hard to specify their local or global preferences exactly (by functions, equations, etc). Instead, they rather express their local preferences in natural language (high salary, small distance, etc.), similarly global preferences. If we give to users a form to explain local preferences for all of our properties as well as for the global preference, it could be easily boring for them. Thus we have to use different way to obtain local and global preferences. We follow the idea that everybody can easily say, for a concrete object, how suitable it is. Thus we will require from user to evaluate the objects in scale from the worst to the best.

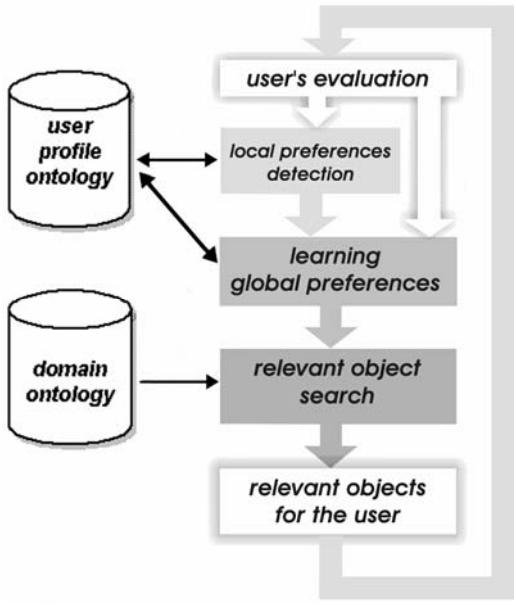

Figure 1. Flow diagram of user profile dependent part

Whole user dependent searching process is illustrated on Figure 1. The objects to evaluation can be a representative set of samples, suitable objects for some user type profile or any objects retrieved during a session. After evaluation of some objects, our

system can learn user preferences based on (possibly hidden) attributes or properties of objects and give to user suitable objects from the whole set of objects. After new objects are given to user, he can evaluate them and start the whole process again.

Learning process has two main phases – learning of local preferences and learning of global preferences. First we need to learn which values of attributes are better for the user – the user local preferences. As we mentioned, the natural meaning of this ordering is that the user determines the relations "'better" or "worst" between any two values of a given property. Learning of local preferences is described in chapter 2.1.

2.1. Detecting local preferences

To learn local preferences, we give to user a representative sample of hotels (see table 1) with several attributes. The attributes in our example are: distance from the city center, price of the accommodation and equipment of rooms (without equipments, TV, internet or both). The user classifies every hotel into one of the three classes (categories): poor, good and excellent according to the relevance of the hotel to him. In real world we use 7 classes of Likert scale.

Table 1. Representative sample of hotels evaluated by the user

Hotel	distance	price	equipment	evaluation
Apple	100 m	99 $	Nothing	poor
Danube	1300 m	120 $	Tv	good
Cherry	500 m	99 $	Internet	good
Iris	1100 m	35 $	internet,tv	excellent
Lemon	500 m	149 $	Nothing	poor
Linden	1200 m	60 $	internet,tv	excellent
Oak	500 m	149 $	internet,tv	good
Pear	500 m	99 $	Tv	good
Poplar	100 m	99 $	internet,tv	good
Rhine	500 m	99 $	Nothing	poor
Rose	500 m	99 $	internet,tv	excellent
Spruce	300 m	40 $	Internet	good
Themse	100 m	149 $	internet,tv	poor
Tulip	800 m	45 $	internet,tv	excellent

The local preferences can be detected in several ways. In this chapter we discuss two statistical models. Note that "equipment" is a text attribute so it needs no ordering detection. We focus on distance and price.

The first method of ordering detection (local preferences) is QUIN [13] - a system of qualitative data mining which discovers trends in data. This system can also display graphs (Figure 2), where the x-axis represents attribute values and the y-axis represents the user evaluation from the data. QUIN finds the following trends in our sample data:

```
   M-,+(price,distance) (Cov=97%=62(?31%)/64)
class=eval | distance price {97%=62(?31.25%)/64}
```
(1)

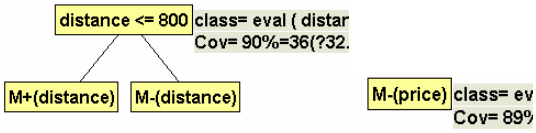

Figure 2. Qualitative trees computed by QUIN when considering the attributes separately.

The result (1) means that the price has negative influence and the distance has positive influence on the user evaluation/ranking. In other words, the user classification decreases with increasing price of a hotel and it increases with increasing the distance of a hotel from an airport. If we check every attribute separately without considering any other attributes, we get the results as on figure 2.

These results state that the user evaluation increases when the distance is less or equal than 800 meters and it decreases in other cases. The user evaluation also increases when the price decreases on the whole domain of prices.

Notice that these results are different from the case when we check the attributes together (1). If user evaluates/ranks the data according to all attributes together, he or she defines the global preferences. So if we want to achieve better results for local preferences, we must consider every attribute separately. Note that these local preferences can also depend on text attributes. Then there are two possibilities: if there is some ordering between values of this attribute, it must be stated by the user (for example green cars are better than red ones and these are better than yellow …). Then we convert these values to numbers according to the user ordering. In the second case there is no ordering between the attribute values. Then here we can use our model of fuzzy similarities. Nevertheless in this paper and example we do not consider these attributes in the local preference checking process for the sake of effectiveness of this process (note that in this case the results can be a little distorted). More about this method can be found in [6].

The second model of checking the ordering (local preference) is the usage of multivariate polynomial regression MPR [11]. MPR gives the following result:

```
eval = +0.000837224*distance -0.0100111*price +2.4805
```  **(2)**

This result is similar to (1). The attribute distance has positive influence and the attribute price has negative influence to the user evaluation.

2.2. Learning global preferences

We learn user's global preferences by the method of Ordinal Classification with Monotonicity Constraints described in [6,7]. Our method is based on Inductive Logic Programming ILP system ALEPH [8], the relational data mining model [1].

In the learning process we compute the user's global preferences by using his/her local preferences (learned e.g. by QUIN). We use ILP because our local preferences (orderings) can be represented well in this framework (by definite clauses). For illustration, consider that we have discretized values of distance to three classes: near, middle and far. The different orderings of these classes for different users can be:

near ≤ middle ≤ far (non-decreasing)
near ≤ far ≤ middle
far ≤ middle ≤ near (non-increasing)
far ≤ near ≤ middle
middle ≤ near ≤ far (the middle values are "better" than near or far)
middle ≤ far ≤ near

A usual aggregation function can be easily simulated by monotone classification rules in the sense of many valued logic. The results of our approach (the user's global preferences) computed from the data in our illustrative example are the following classification rules about the relevance of hotels to user preferences (see Figure 3):

- evaluation = excellent IF distance>=500 AND price<=99 AND services={tv,internet}
- evaluation = good IF (distance>=500 AND services={tv})
- evaluation = good IF (price<=99 AND services={internet})

From our results we can obtain also additional information about crisp attributes (equipment) The meaning of any classification rule is as follows: If the attributes of object x fulfill expressions on the right side of the rule (body) then the overall value of x is at least the same as on the left side of the rule (head). We can, of course, assign the explicit values to vague concepts like excellent or good. During the simulation of computation of aggregation function we can simply test the validity of requirements of the rules from the strongest rule to weaker ones. When we find the rule that holds, we can say that the overall value of the object is the value on the left side of the rule. Since we test the sorted rules, we always rank the object with the highest possible value.

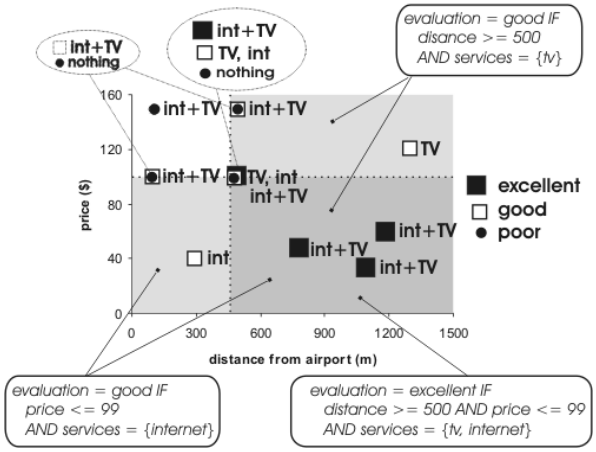

Figure 3. The results of our approach in the case of our illustrative example.

We can see that the ordered meaning of the classification is preserved in our results: hotels classified in the grade "excellent" by the user also fulfills requirements for "good" and "poor" hotels, and "good" hotels fulfills requirements for "poor" ones

(e.g. the hotel with 800 m distance from airport and 45$ price equipped with internet and TV is "at least as appropriate as" the hotel 300 m far from the airport and 40$ price equipped just with internet) according to the local preferences (more far and more cheap is the better).

2.3. Fuzzy RDF based on fuzzy description logic

In this chapter we analyze the model of fuzzy RDF/OWL based on a model of fuzzy description logic. The terms like cheap, expensive or near represent fuzzy sets as in [14]. Our model of fuzzy RDF includes such fuzzy sets stored as RDF triples.

One important feature of our model is that we can prepare fuzzy RDF independently from user global preference. This is because we can adapt to user by adjusting his aggregation function @. This makes the processing of data more effective, because we do not need to order data for every user query. We already have the data ordered and we just combine the relevancies from fuzzy RDF into one result.

We introduce the model of building fuzzy RDF based on fuzzy description logic **f-**EL proposed in [10]. This logic removes some features of both classical and fuzzy description logic (like negation, universal restriction and fuzzy roles), but it adds aggregation operator @. We loose the ability to describe fuzziness in roles (i.e. our data from domain ontology are crisp; we do not consider uncertainty in values. User preferences are represented as fuzzy concepts and they are the source of fuzziness in results.), but we gain combination of particular user preferences to a global score by his @. Advantage of this description logic is lower complexity of querying. Expressivity is lower than that of full fuzzy DL but still sufficient for our task and embedability into web languages and tools (see [9]).

Basic building boxes of description logic are concepts and roles. Here, roles express properties of resources (hotels in our case). Although the basic model of expressing RDF triples <subject, predicate, object> are oriented graphs, we use here the language of logic: predicate(subject, object).

The alphabet consists of sets N_C of concepts names, N_R role names and N_I instance names. The roles in **f-**EL are crisp and concepts are fuzzy. Our language of description logic further contains constructor $\exists$ and a finite set of aggregation functions symbols $@_U$ for each user and/or for each group of users. Concept descriptions in **f-**EL are formed according to the following syntax rules

$C \rightarrow \top|A|@(C_1,\ldots,C_n)|\exists r.C$

In order to give this syntax a meaning, we have to define interpretations of our language. In **f-**EL we have interpretations parameterized by a (possibly partially, usually linearly) ordered set of truth values with aggregations.

For a preference structure – a set of truth values P=[0,1], a P-interpretation is a pair $I = <\Delta^I, \bullet^I>$ with nonempty domain Δ^I and fuzzy interpretation of language elements:

$A^I: \Delta^I \rightarrow P$, for $A \in N_C$

$r^I \subseteq \Delta^I \times \Delta^I$, for $r \in N_R$.

$(\exists r.C)^I(x) = \sup\{C^I(y):(x,y) \in r^I\}$

$(@(C_1,\ldots,C_n))^I(x) = @^{\bullet}(C_1^I(x),\ldots,C_n^I(x))$

Example. N_R = {*price, distance_from_airport*} where *price* and *distance_from_airport* are RDF predicates from domain ontology.

N_C = {$cheap_U$, $close_U$} where $cheap_U$ and $close_U$ are fuzzy functions explicitly figured by user preference ontology.

N_I = {*Apple, Danube*, 99, 120,…}

In Herbrand-like interpretation $\mathcal{H}$ we have:

$Apple^{\mathcal{H}}$ = Apple, $Danube^{\mathcal{H}}$ = Danube, $99^{\mathcal{H}}$ = 99, $120^{\mathcal{H}}$ = 120

$cheap_U{}^{\mathcal{H}}(99) = 0.53$, $cheap_U{}^{\mathcal{H}}(120) = 0.42$

$price^{\mathcal{H}}$ ={(Apple, 99), (Danube, 120)}

For simplification we overload our concept $cheap_U$ also for hotels (usually we must create new concepts in this case e.g. $cheap\_hotel_U$):

$cheap_U{}^{\mathcal{H}}(x) = (\exists\ price.cheap)^{\mathcal{H}}(x)$ then

$cheap_U{}^{\mathcal{H}}(\text{Apple}) = \sup\{cheap_U{}^{\mathcal{H}}(y)\text{: (Apple,y)} \in price^{\mathcal{H}}\}=0.53$

$cheap_U{}^{\mathcal{H}}(\text{Danube})=\sup\{cheap_U{}^{\mathcal{H}}(y)\text{: (Danube,y)} \in price^{\mathcal{H}}\}=0.42$

The supremum affects the case when we have more different prices for one hotel. Then we take the biggest result.

$(@(cheap_U,\ close_U))^{\mathcal{H}}$ (Apple) = $@^{\bullet}(cheap_U{}^{\mathcal{H}}$ (Apple), $close_U{}^{\mathcal{H}}$ (Apple)) = (3* $close_U{}^{\mathcal{H}}$(Apple)+2* $cheap_U{}^{\mathcal{H}}$(Apple))/5

Fuzzy description logic **f-**EL is the motivation of creating a model for fuzzy RDF. Modelling fuzzy classes by classical RDF we can model a fuzzy instance $cheap_U{}^{\mathcal{H}}$(Apple) = 0.53 by RDF triple: <Apple, $cheap_U{}^{\mathcal{H}}$, 0.53>.

This is an embedding of a fuzzy logic construct into classical RDF, which needs to translate also constructions of DL, namely $\exists price.cheap$ in fuzzy DL turns to composition of roles, where $\exists price.(\exists cheap.\top)$ needs an aggregate max (resp. top-k) extending concrete domain DL (see [15]). It is out of the scope of this paper to describe this concrete domain DL and embedding of our fuzzy DL in more detail. What we claim here is an experimental implementation (of both our fuzzyDL, special concrete domain crisp DL and a related model of RDF with extended syntax and semantics of owl:someValuesFrom).

Now, in our model, we can specify user profiles and represent these profiles in Fuzzy RDF triples based on data from domain ontology. Our next task is to find best objects for individual users.

Consider the type of user preferring cheap hotels. There can be many users that prefer cheap hotels. We want to share the same instance of the class *cheap* for them. However, their notion of "cheapness" can be different. For example one user can say that cheap hotels have price lower than 30$, while for some other user cheap hotel ends at 50$. Fortunately we can easily change the overall evaluation of objects to reflect these individual requirements. In the following theorem $f_1,\ldots, f_n$ are functions that represent local preferences expressed in user preference ontology. Function values $f_1(x),\ldots, f_n(x)$ are literals from fuzzy RDF expressing the relevance of object x. Note that we can use the same values and adjust the aggregation function only.

Theorem. Let $f_1,\ldots, f_n$ be bijective fuzzy functions, $f_i=a_ix+b_i$, $b_i\neq 0$. Let $g_1, \ldots, g_n$ be partially linear functions such that they preserve the ordering of $f_1, \ldots, f_n$. Let @ be n-ary aggregation function. Then there exists n-ary aggregation function @′ such that:

$$(\forall x)\ @(g_1(x), \ldots, g_n(x)) = @'(f_1(x), \ldots, f_n(x)).$$

Proof: The theorem above says about existence of an aggregation function @′. We will show how to find this function.

Our implicit assumption is that f_i and g_i are defined on the same unite interval J. Function g_i is linear over certain subintervals of J. We take one such subinterval and name it J_1. The following holds for f_i and g_i over J_1:

$$(\forall\, x \in J_1)\;\; f_i(x) = a_i(x)+b_i \text{ and } \; g_i(x) = c_i(x)+d_i$$

Now we define new function h as:

$$\left(\forall x \in J_1\right)\;\; h\left(y\right) = c_i \frac{\left(y - b_i\right)}{a_i} + d_i$$

We repeat this process for every subinterval of J where g_i is linear. The aggregation function @′ is:

$$@'(y_1, ..., y_n) = @(h_1(y_1), ..., h_n(y_n))$$

If we substitute $f_i(x)$ for every y_i:

$$@'(f_1(x), ..., f_n(x)) = @(h_1(f_1(x)), ...,h_n(f_n(x_n)))$$

For arbitrary subinterval $J_1 \subseteq J$ where every g_i is linear we get

$$h_i(f_i(x)) = c_i \frac{f_i(x) - b_i}{a_i} + d_i = c_i \frac{a_i x + b_i - b_i}{a_i} + d_i = c_i x + d_i = g_i(x)$$

$$@'(f_1(x), ..., f_n(x)) = @(h_1(f_1(x)), ..., h_n(f_n(x))) = @(g_1(x), ..., g_n(x))$$

□

This theorem allows users to specify their own meaning of “cheapness” exactly by fuzzy function and similarly for other attributes.

2.5. Relevant object search

Now we have orderings of properties from learning of local preferences, rules from learning of global preferences and prepared ordered data stored in fuzzy RDF. Our last task is to find top k objects to user. We use the extension of middleware search of Ronald Fagin [2]. The main idea is to browse only necessary data until the system is sure that it has top-k objects already. Thus we do not need to calculate with whole data from domain ontology. Retrieving only k best objects can save time and is usually sufficient for user. Saving of time grows with the number of objects stored in domain otology and also with the number of properties we consider.

R. Fagin considered input ordered lists in [2]. Each list includes goodness of a concrete property for each object. Data in the lists are ordered from the best to the worst in particular property. Fagin considered two kinds of accesses to lists: sorted and random access. The sorted access gets the next best object from the list after each access, so we can retrieve data ordered from the best to the worst. The random access asks for the value of a particular property it includes for a concrete object. We want to minimize the number of accesses to lists and of course the time of searching. The random access has one big disadvantage. Each random access requires searching of

the value of concrete object. In the case of sorted access the results are prepared immediately (values can be preordered, in fuzzy RDF in our case, to the several orderings before user comes) and, moreover, data can be sent in blocks. Because of this feature in our system we prefer mainly the algorithms, which use only sorted access.

U. Güntzer et al. [5] proposed stream-combine algorithm as the first one from the random access only approaches. Much better and supported by good formal model was No Random Access algorithm presented by Fagin et al. [3]. As an improving P. Gurský [4] presented 3P-NRA (3 phased No Random Access) algorithm which reduces unnecessary operations from No Random Access algorithm and proposes some heuristics.

As we found in our experiments, using the techniques of top-k search can significantly save number of needed accesses and accordingly the time needed especially in huge sets of objects.

Data for experiments were generated with various distributions of values. We used 2 exponential and 2 logarithmic distributions with 10000 objects and 6 types of aggregation functions. Using different combination of source data and aggregation functions we compose 25 different inputs for algorithms. In the final results we use the averages of the particular results. In future we need to make robust test over bigger artificial data as well as real data.

3. Conclusions

In this paper we have described a model of system enabling users to search objects from the same domain and heterogeneous sources. Data are collected from various sources are processed to vector index and to a domain ontology. The system implements both user independent search and personalized search for users with different preferences. Our theoretical model integrates all parts of the system from collected data in classical RDF form to user query answering. We do not have a model for web resource downloading and ontology annotation part.

The model of user dependent search is based on modeling preferences by fuzzy sets and fuzzy logic. We present the semantics of fuzzy description logic **f**-EL. User dependent search integrates both inductive and deductive approach. By induction we can learn local and global preferences. Results of induction as well as hand-filled orderings can be easily modeled in user ontology and fuzzy RDF. Deductive part of the system uses the preferences to find suitable objects for a user. User can express his preferences precisely by fuzzy functions and aggregation function. Easiest way is to evaluate several objects in a scale and let the system to learn preferences. Repeating this process (evaluating a set of objects and finding new objects) leads to refining the user profile. User, who is content with his profile and with the presented search results, does not have to evaluate objects again. When the domain ontology is actualized, user can just get the best objects according to his profile. Effective identification of suitable user type for a new user is the aim of our future research. It can be done by collecting data from real users. We want to collect some personal information like age, gender, job position, etc. and the explicit specification of preferences from each user and then analyze the data in search for dependency.

Presented model was experimentally implemented and integration was tested using Cocoon and Spring Framework [12]. Both searching methods (user dependent

and user independent search) are compound of tools which can be further improved separately. After the phase of gathering data about users, it will be possible to compare the efficiency, speed and complexity of these methods and decide about their practical usage.

Our approach is used also in the Slovak project *NAZOU – Tools for acquisition, organization and maintenance of knowledge in an environment of heterogeneous information resources* [17] to find relevant job offers for the user according to him/her preferences.

Similar strategy of communication with users (evaluation of sample objects) was used in [16], where the learning part of the system was covered by a neural network. This approach does not permit to model user preferences. We did not found any other similar approach to the whole process.

Acknowledgements. Partially supported by Czech projects MSM 0022520838, 1ET100300419 and 1ET100300517 and Slovak projects VEGA 1/3129/06 and NAZOU.

References

[1] Džeroski, S., Lavrač, N.: An introduction to inductive logic programming. Relational data mining, S. Džeroski, N. Lavrač eds., Springer 2001, 48-73.
[2] Fagin, R.: Combining fuzzy information from multiple systems, J. Comput. System Sci. (1999) 58:83-99
[3] Fagin, R., Lotem, A., Naor, M.: Optimal Aggregation Algorithms for Middleware. In Proc. 20th ACM Symposium on Principles of Database Systems (2001) 102-113
[4] Gurský, P.: Towards better semantics in the multifeature querying. Proceedings of Dateso 2006, ISBN 80-248-1025-5, (2006), 63-73
[5] Güntzer, U., Balke, W., Kiessling, W.: Towards efficient multi-feature queries in heterogeneous enviroments. ITCC, IEEE Computer society (2001) 622 – 628
[6] Horváth, T., Vojtáš, P.: Ordinal Classification with Monotonicity Constraints. In. ICDM 2006, Leipzig, Germany: LNAI 4065, Springer, 2006, pp. 217 – 225.
[7] Horváth, T., Krajči, S., Lencses, R., Vojtáš, P.: An ILP model for a graded classification problem. J. KYBERNETIKA 40 (2004), No. 3, (2004), p:317–332.
[8] Srinavasan, A.: The Aleph Manual. Technical Report, Comp.Lab., Oxford University
[9] Vojtáš, P.: Fuzzy logic aggregation for Semantic Web search for the best (top-k) answers, in Fuzzy logic and the semantic web, E. Sanchez ed. Elsevier (2006) 341-360
[10] Vojtáš . "A fuzzy EL description logic with crisp roles and fuzzy aggregation for web consulting" , In Proc. IPMU'2006, B. Bouchon-Meunier et al eds. EDK (2006) 1834-1841
[11] System MPR: <http://fuzzy.cs.uni-magdeburg.de/~borgelt/software.html#regress>
[12] The Spring Framework. http://www.springframework.org/
[13] Bratko, I., Šuc, D.: Learning qualitative models. AI Magazine 24 (2003) 107-119
[14] Vojtáš, P.: Fuzzy logic programming, Fuzzy Sets and Systems, 124(3) (2001) 361-370
[15] Baader, F., Kuesters, R., Wolter, F.: Extensions to description logics, in HDL 219-261
[16] Naito, E., Ozawa, J., Hayashi, I., Wakami, N.: A proposal of a fuzzy connective with learning function and query networks for fuzzy retrieval systems. In Fuzziness in database management systems. P. Bosc and J. Kacprzyk eds. Physica Verlag, 345-364, 1995
[17] http://nazou.fiit.stuba.sk