Various aspects of user preference learning and recommender systems*

Alan Eckhardt

Department of Software Engineering, Charles University,
Institute of Computer Science, Czech Academy of Science,
Prague, Czech Republic
eckhardt@ksi.mff.cuni.cz

Abstract. In this paper, we describe area of recommender systems, with focus on user preference learning problem. We describe such system and identify some interesting problems. We will compare how well different approaches cope with some of the problems. This paper may serve as an introduction to the area of user preference learning with a hint on some interesting problems that have not been solved yet.

Keywords: user preference learning, data mining, recommender systems

1 Introduction

User preference learning is an important part of any recommender system. We will work with a scenario of user searching for some object (we will refer to user as "she" for not having to distinguish between he and she). Recommendation may help user to find what she is looking for more quickly and efficiently, because she has not to crawl through hundreds of products but sees the recommended products on only one or two pages. Of course, these recommended products may not be an exhaustive list, but they are a hint for user.

In Section 2 some important related work is studied, providing also an introduction to the problematic of user modeling. Then, in Section 3, we describe how user preferences are modeled in our approach. In Section 4 is described a typical scenario of recommendation cycle for user. We also describe how our user model is constructed and ways for estimating usefulness of a user model. In Section 5 are listed some interesting problems associated with learning of user preferences. Finally, in Section 6 are conclusive remarks and more importantly areas for future work in this field are proposed.

1.1 Example

In the whole paper, we will refer to a set of "objects". These objects are supposed to be of interest for user, probably she wants to buy one. In our traditional example, user is

* This paper was supported by Czech projects MSM 0021620838 and 1ET 100300517.

K. Richta, J. Pokorný, V. Snášel (Eds.): Dateso 2009, pp. 56–67, ISBN 978-80-01-04323-3.

buying a notebook. She has some preferences of notebooks, e.g. the maximal price she is willing to pay, the preferred manufacturer or the size of the display.

This example is suitable for our approach because notebooks have well defined attributes that describes the product completely. More about this is in Section 4.1.

1.2 Notation

We will work with a set of objects X. Set X can be also viewed as a set of identifiers of objects (id), which will be often referred to as o. Every object has attributes $A_1, ..., A_N$ with domains $D_{A_1}, ..., D_{A_N}$. If we want to specify the value of an attribute A_i for an object o_j, we will use notation $A_i(o_j)$. We will use $X^i(a)$ when denoting a set of objects for which attribute A_i has the value a. When the attribute will be clear from context (which will be most of the times), we will use only $X(a)$.

2 Related work

User preference modeling was very nicely described in [4] and also in a more general view in [10]. In Figure 1 (which was taken from [4]) are various components of preference modeling. Model is how user preferences are understood – for our purposes Total order of outcomes (or objects) will be most suitable. That means we can create a list of all objects ordered according to user preferences. Language is a way for user to express her preferences. It may be a rating of an object, as $V(o)$ in Figure 1, or a query to the system etc. Language is explored in Section 4.1.

The most interesting part for us is Interpretation, where the information from user is somehow transformed into Model, e.g. the total order of outcomes. However, because of intuition, we will slightly change the notation – we will refer to the method for creating the total order as "user model" or "user preference model". Interpretation may be also viewed as learning phase, where a user preference model is constructed.

In the following two sections, two alternative ways of user modeling are described along with their possible interpretations. First, qualitative models are based on comparing two objects between them, and second, quantitative models are based on evaluation of a single object with a scoring function.

2.1 Qualitative approaches

Preference relations are the most used and studied qualitative approach. There is a huge amount of related work in the field of preference relations. Preference relations represent preferences as a relation between two objects, it is usually assumed that this relation creates some pre-order on X. There are typically three relations, P are strict preferences, I represents indistinguishability of objects or equality of preference and R is union of P and I meaning that it represents non-strict preferences. For example $P(o_1, o_2)$ means that o_1 is strictly preferred to o_2, $I(o_1, o_2)$ on the other hand means that o_1 and o_2 have the same preference and finally $R(o_1, o_2)$ means that o_1 is preferred or equal to o_2.

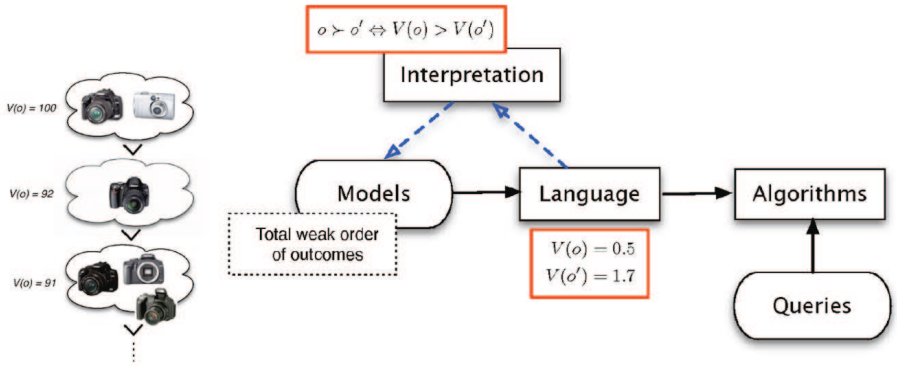

Fig. 1. Preference model components.

Preference relations in database systems and their integration into SQL by preference queries was studied by Chomicki in [8], [9]. Also Kießling contributed to this field with [25].

A different approach was suggested by Kießling in [19], [24]. This approach is based on the idea of preference relations but it uses relations over attribute values rather than relations over whole objects. This is more like our approach based on fuzzy logic. However Kießling does not use scoring functions but uses special predicates POS, NEG etc. to represent relation between two attribute values. An example from [24] is from the area of cars: POS(transmission, automatic) and NEG(make, Ferrari) meaning that automatic transmission is preferred to any other type and any maker is preferred to Ferrari.

As for learning of preference relations, a great contribution is from Fürnkranz and Hüllermeier [17], [20].

2.2 Quantitative approaches

The other approach, also adopted by us, is quantitative. It sorts objects by a score defined by a scoring function. This approach is arguably less expressive than the qualitative one – it can not express e.g. a cycle in preferences. There are also some very interesting works in this area.

Content based models Content based models uses attributes of object for construction of scoring function. For example Fagin in [16] proposed a way of combining numerous fuzzy inputs. Another classical work is from Agrawal [6].

Collaborative filtering Besides content based models, such as the one presented in Section 3, there is another widely used user model that is based on the preferences of

other users. Collaborative filtering was proposed in early 90's in [18] and further developed. One of the well-known systems using collaborative filtering is GroupLens [27].

Collaborative filtering is based on the idea of similarity of users. When we want to know how user u_1 will like object o_1, one way is to look how other people liked o_1. `Amazon.com` succeeds in describing this approach in one sentence "Customers Who Bought This Item Also Bought...".

The better way is to restrict only to those users that are similar to u_1. The similarity may be computed in various way, the most common is the similarity of ratings of objects other than o_1. Other possibility is to compute the similarity of user profiles – e.g. find managers, from 25 to 30, divorced, with interest in psychology and computer science. There is a hidden assumption that similarity in profile imply similarity in preferences, which may not be always true.

3 User model based on fuzzy logic

In this section, we describe user model we are using. This model is based on a scoring function that assigns every object a score that represents the rating of that object. User rating of an object is a fuzzy subset of X, i.e. a function $R(o) : X \rightarrow [0, 1]$, where 0 means least preferred and 1 means most preferred. Our scoring function is divided into two steps.

Local preferences In the first step, which we call local preferences, every attribute value of object o is normalized using a fuzzy set $f_i : D_{A_i} \rightarrow [0, 1]$. The meaning is that 1 represents most preferred value and 0 stands for the least preferred value. These fuzzy sets are also called objectives or preferences over attributes. With this transformation, the original space of objects' attributes $\prod_{i=1}^{N} D_{A_i}$ is transformed into $[0, 1]^N$. Moreover, we know that the object with transformed attribute values equal to $[1, ..., 1]$ is the most preferred object. It probably does not exist in the real world, though. On the other side, the object with values $[0, ..., 0]$ is the least preferred, which is more probable to be found in reality.

Global preferences In the second step, called global preferences, the normalized attributes are aggregated into the overall score of the object using an aggregation function $@ : [0, 1]^N \rightarrow [0, 1]$. Aggregation function is also often called utility function.

Aggregation function may take different forms; one of the most common is weighted average, as in the following formula:

$@(o) = (2 * f_{Price}(o) + 1 * f_{Display}(o) + 3 * f_{HDD}(o) + 1 * f_{RAM}(o))/7$,

where f_A are fuzzy sets for normalization of attribute A.

Another totally different approach was proposed in [15]. It uses the training dataset as partitioning of normalized space $[0, 1]^N$. For example, if we have an object with normalized values $[0.4, 0.2, 0.5]$ with rating 3, every other object with better attribute values (e.g. $[0.5, 0.4, 0.7]$) is supposed to have rating at least 3. In this way, we can find the highest lower bound on any object with unknown rating. In [15] was also proposed

a method for interpolation of ratings between the objects with known ratings and even using the ideal (non-existent) virtual object with normalized values $[1, ..., 1]$ with rating 6.

In other words, we can say that the pareto front is constructed in the first step. Pareto front is a set of objects that are not dominated by any other object. We say that object o_1 dominates object o_2 iff $\forall i = 1, ..., N : f_i(o_1) > f_i(o_2)$, i.e. o_1 is better in every attribute than o_2. In the second step, we choose the best object from the pareto front.

4 A recommender system

A recommender system tries to help user to find the object she is looking for. It is necessary for user to transfer some information about her preferences to the system. It is convenient for user to describe her preferences in an intuitive and simple way. The more complex user interface is, the more structured information the system gets but much less users will use it (according to [26], as little as 9 out of 260 people provided a feedback to their system). This fact also penalizes preference relations – user is supposed to compare two objects, but the number of couples is quadratic to the number of objects.

There is an example how a recommender system may work in Figure 2. System presents user with a set of objects S_0. User rates some of these objects and this information is sent back to the system as feedback U_0. User model is constructed from user's ratings and a personalized set S_1 is sent to user. Again, user rates some objects (U_1) and system updates its user model and sends S_2 and this cycle may go on until user is satisfied or bored. In Section 4.1 various possible types of feedback from user are studied. In Section 4.3 and 4.2 the construction and update of user model is described.

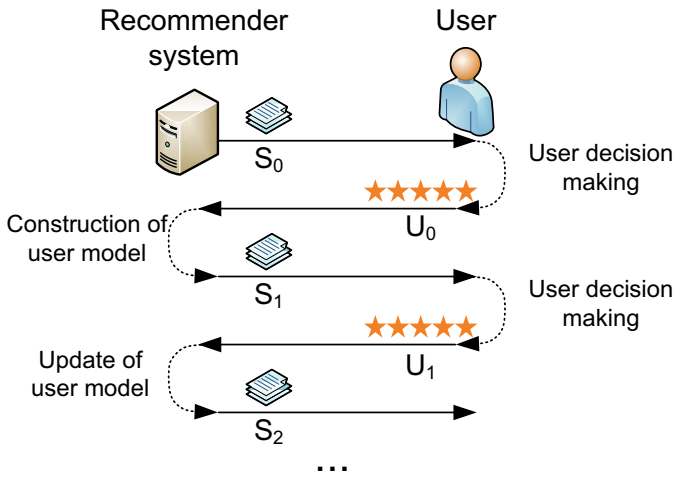

Fig. 2. A use of a recommender system in steps.

The process of recommendation is in Figure 3. User does some actions with the system, which are processed by various components of the system. You can see that some inputs may be processed by multiple components. There are three examples in the figure – analysis of user behaviour, collaborative filtering, analysis of ratings and direct query. Each of these components then creates user model which is used for prediction of preference of all objects. Some models, like collaborative filtering, use the information about other users or other additional information. All these models are then combined together to provide most precise recommendation for user. Furthermore, when the system identifies some of the situation discussed in 5, it can favour the model that behaves best in this situation or the other way round. Collaborative filtering is not good when there is a small number of users, so if that is the case, it can be disfavoured.

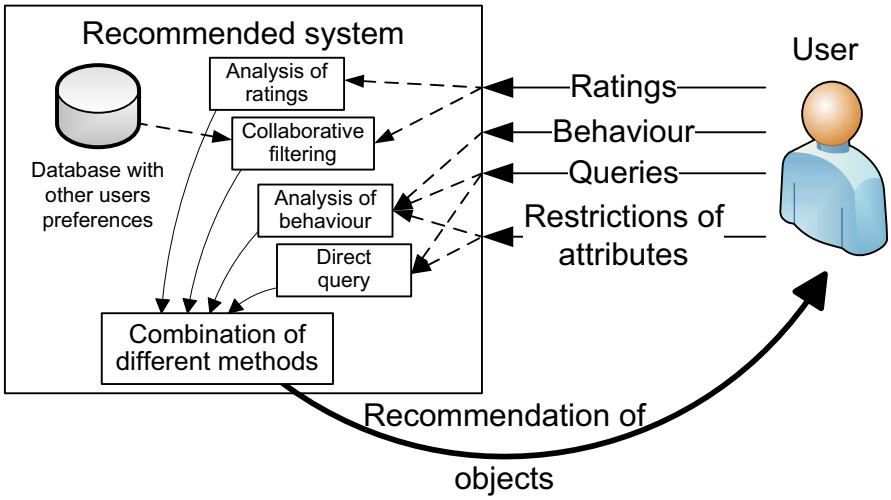

Fig. 3. Structure of a recommender system.

4.1 Input from user

From the information user provides to the system, her user model is built. User model should be capable to determine which objects user will like or to what degree an object will be preferred. The construction of user model is of most interest for us. We are working with user ratings. These ratings user associates to a small number of objects. This is key aspect of user model construction – it can not be expected that user will rate hundreds of objects. When doing experiments, we often limit the size of training set to 40 objects.

Other approaches may expect different forms of information from user other than ratings. For example for preference relations, comparisons between two objects is the

expected input. The full-text query issued by user may be also viewed as a source of information about what user wants – document retrieval uses queries as its only information from user.

Datasets There exists publicly available datasets of user ratings such as Netflix [3]. In these datasets exist users with even thousands of ratings. Unfortunately, most of these datasets have a very small number of attributes.

- Books [5] - have author, title, year of publication and publisher. It contains 433 670 ratings with non-zero rating from 77 805 users.
- Jokes [23] - have no attributes, except the text of the joke itself. It contains 4.1 million of ratings by 73 421 users.
- Films [2], [3] - have many attributes (from IMDB [1]) but they are complicated. One film can have many actors, many producers, many directors etc. The normal attributes such as length of a film are not determined easily, because they differ across countries, editions or releases. Movielens contains 10 million ratings from 71 567 users. Netflix contains over 100 million ratings.

All this is data from users who were using a system for a long time, several years in most cases.

Behaviour analysis User behaviour interpretation was studied in [21] and [22]. Because user tends not to give very much information about herself, the interpretation of her behaviour may provide a useful information that supports the explicit actions she had done (such as ratings of objects). There are many events that can be monitored, such as the time spent on a web page with details of an object, clicking on a page, scrolling down a document, filtering the content of the page, issuing a query etc. These actions are then interpreted as if they were motivated by her preferences, e.g. the longer user stays on a page, the more preferred is the object on that page. When user browses the shop by categories or restricts values of some attribute, it is a clear statement what she likes. For example when user narrows her search to notebooks with display size 14", we can deduce that 14" is the best size of display. This information helps when creating local preferences (in Section 4.2). The order in which such restrictions are applied may represent importance of attributes. Typical counter-example against interpretation of behaviour is when user is going for a coffee, leaving the browser open on that page, or a user searching for an object for a friend (see also Section 5 for this issue).

In following sections, we outline some methods for creating local and global preferences, which are contributions we made to this area in the past.

4.2 Learning local preferences

The acquisition of local preferences differs for different types of attributes. For nominal attributes, we use a method based on representative value that is computed from user ratings [11, 13].

For numerical attributes, the problem is more complicated – there is usually only one object with value a. A typical example is price – there is often only one notebook

with price e.g. $941\$$. We can stick with traditional methods such as linear regression, where the input is formed from the ratings of objects and their prices. However linear regression may be affected by the distribution of data, but we want a function that corresponds to real values across the whole domain, not only there where are most values. Because of this, we proposed a way of using representants for numerical domain in [11].

Lately, we identified another problem with numerical domains. It may happen that a value of another attribute, often nominal, affects the normalization of a numerical attribute. For example, when flying with British Airways, you may prefer lower price, because the comfort is fine in economy class, but with Aeroflot, you may prefer higher price because the overall comfort is worse. This phenomenon is related to ceteris paribus preferences [28] and CP-nets [7].

4.3 Learning global preferences

Method called "Statistical" was described in [11] and in [12]. It is based on the evaluation of distribution of ratings across attribute domain D_A and from this distribution it is possible to derive weight that A plays in the decision process of user. Then the weighted average with these weights is used as aggregation function.

When constructing scoring function called "Instances" that uses the objects from training set as lower bounds (from [15]), there is little to do. It does not need any further transformation or analysis. This is unfortunately balanced by a more complicated computation during evaluation of new objects.

4.4 How to measure usefulness of a recommender system

In this section we identify several ways of measuring usefulness of a recommender system. As in data-mining, we adopt the idea of training (Tr) and testing sets (Ts). User model is constructed from objects in training set and then its performance is measured on testing set. In the following, the real preference of an object o will be denoted as $R(o)$ and preference user model estimates as $\widehat{R}(o)$.

RMSE RMSE stands for root mean squared error. It is widely used as error measure in data-mining community. It is computed as $\sqrt{\sum_{o \in Ts} (\widehat{R}(o) - R(o))^2 / |Ts|}$.

When considering user preferences, we introduced a modified RMSE, weighted RMSE. It associates more weight to the preferred objects than to the non-preferred. The formula is $\sqrt{\sum_{o \in Ts} R(o) * (\widehat{R}(o) - R(o))^2 / \sum_{o \in Ts} R(o)}$. The less is RMSE, the better the system performs.

Tau coefficient Tau coefficient is known in economy. It is used to measure the similarity of the ordered lists of objects. It is assumed that both lists contain the same set of objects. For our purpose, we compare the ordering of objects by real user preferences $R(o)$ with the ordering user model would make $\widehat{R}(o)$.

Tau coefficient is based on concordant and discordant pairs. A pair $(o_1, o_2), (p_1, p_2)$ is concordant, if $sgn(R(o_1) - R(o_2)) = sgn(\widehat{R}(p_1) - \widehat{R}(p_2))$, where sgn is signum function. Then the coefficient is computed as $\tau = \frac{n_c - n_d}{\frac{1}{2}n(n-1)}$ where n_c is number of concordant pairs, n_d is the number of discordant pairs and n is the number of objects in lists. The higher Tau coefficient is, the better for the system.

We can apply weighting scheme to Tau coefficient, too. Objects with higher real preferences will matter more than objects with low preference, when comparing the two orderings.

ROC curves ROC curve is a method for capturing the performance of a system under different conditions. In terms of classification of positive and negative examples, it tells how the recall (the ratio between the number of correctly classified positive objects and the total number of positive objects) of the system increases if we relax the ratio of correctly classified negative objects and the number of all negative objects.

The problem here is that we do not have a simple positive-negative scenario. Ratings have typically the domain $\{1,2,3,4,5\}$. The possible solution is to consider several cuts in this domain and measure at each of these cuts. We can take as positive objects with rating 5 and the rest as negative. Then, objects with ratings 5 and 4 will be considered positive and the rest as negative etc. In this way, we will get 5 different ROC curves.

Amount of information required from user Another important characteristic of a recommender system is amount of information that is required from user. Possible approaches are described in Section 4.1. Often, the higher precision of a system is outweighed by a larger investment from user. But the intuition says that few users will be willing to perform complicated tasks, no matter the increase of helpfulness of the system. The less information the system needs from user, the better for user.

Improvement in user ease of work with the system The main task of recommender systems is to help the user. All the above mentioned measures are good because they can be computed analytically, but they cannot capture the real improvement for user. This improvement can be only given by the real user working with the various recommender systems and comparing them between them. There are some problems with this comparison

- Every man perceives the ease of work differently. There should be a large number of people testing the systems to be compared to evaluate overall suitability.
- The system has to be implemented including the user interface. Different methods work with different inputs from the user and they have to be able to monitor these inputs.
- People testing the system has to have a motivation to work with the system. If they are not, they would not be critical or they would apply some criteria irrelevant in the real usage of the system. This is most difficult to achieve.

5 Interesting problems of recommender systems

In this section, some interesting problems are described. All these problems relate to user preference learning and can be viewed as the main contribution of this paper, besides the introductory part.

Cold start When a system is based on user ratings, it has serious troubles in the beginning of its existence. This problem is most pronounced for collaborative filtering-based system, because they need a lot of rating from a lot of users for making correct predictions. For content based recommendation, this is not much of a problem because the accuracy of recommendation does not depend on the number of users of the system. The following issue however affect both approaches.

New user When a new user starts using the system, it is hard to recommend something because the system has little information about her. This issue may be overcome with a default user profile, but this solution is not personalized enough.

There is a possibility to learn something quickly about a new user – it is by choosing a good set S_0 in Figure 2. When there are very different objects in S_0, we can learn immediately what user certainly does not like and what area is of her interest. When S_0 is constructed randomly, there is a higher possibility that there would not be any object the area of user's interest. Suitable S_0 may be found for example using clustering.

Identifying context for user When a user works with the system, it is assumed that she searches for some object she wants. But there are other possible motives such as scanning the area or finding some object for a friend. This context of work deeply influences user behaviour and may severely damage existing user model.

Small rating scale Typical rating is expressed on the scale $\{1,2,3,4,5\}$. When having thousands of objects, this scale is far too small for capturing the ordering among the best objects. We proposed a method Phases for overcoming this obstacle in [14], but it is not a complete analyze and there are many more aspects of this problem. Phases are suited for a single session, for long-term usage there should be another way of differentiating among the objects with rating 5.

This problem could be of course solved by enlarging the scale of ratings to e.g. $\{1,...,100\}$ but its benefit is questionable. Will user really feel the difference between ratings 64 and 65? Will user be able to apply it consistently?

Negative preferences In our model, 0 means the lowest preference. When a manufacturer ASUS has the preference 0, it penalizes the notebook, but in the overall score it may be outweighed by other attributes which are more preferred by user. Negative preferences are typically more stronger than this – a notebook made by ASUS is strongly disregarded. The strength may be expressed by the weight of the attribute, but it is no solution, because for other manufacturers, this attribute may not have such a big weight. Our task is to model a better way of penalizing objects with a highly non preferred value and the way of expressing such strong negative preference.

Time dimension Main problem with acquisition of user preferences is the small size of training data. When user is using the system for a longer period, the training data may get bigger. But here comes a new problem – user preferences are typically unstable. What user preferred a month ago may not be good today. But the high rating user provided a month ago is the same high rating provided today.

There is a need to find a method for penalization of older ratings or promotion of newer ones. The system needs to guess whether to apply the penalization at all – maybe the preferences stay the same when buying a house, but they may change quickly for a movie.

6 Conclusion

We have described a recommender system and identified several interesting issues and problems that occur during user model construction. The main contribution is the suggestion of several possible ways how to measure helpfulness of a recommender system. The experimental evaluation is often lacking in more theoretical focused papers. Also the list of some problems or typical situations in recommender systems may be inspiring for searching of their solutions. We hope that this analysis would be helpful to anyone new in recommender systems and user modeling field.

6.1 Issues for future work

All problems in Section 5 are worth studying, but we would like to address the two last problems in near future – that of Negative preferences and Time dimension.

References

1. The internet movie database. *http://www.imdb.com/.*
2. *Movielens dataset. http://www.grouplens.org/node/73.*
3. *Netflix dataset. http://www.netflixprize.com.*
4. *Preference handling an introductory tutorial. http://www.cs.bgu.ac.il/ brafman/tutorial.pdf.*
5. Improving Recommendation Lists Through Topic Diversification*, 2005.*
6. *R. Agrawal and E. L. Wimmers. A framework for expressing and combining preferences.* SIGMOD Rec.*, 29(2):297–306, 2000.*
7. *C. Boutilier, R. I. Brafman, H. H. Hoos, and D. Poole. Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements.* Journal of Artificial Intelligence Research*, 21:2004, 2004.*
8. *J. Chomicki. Preference queries.* CoRR*, cs.DB/0207093, 2002.*
9. *J. Chomicki. Preference formulas in relational queries.* ACM Trans. Database Syst.*, 28(4):427–466, 2003.*
10. *J. Doyle. Prospects for preferences.* Computational Intelligence*, 20:111–136(26), May 2004.*
11. *A. Eckhardt. Inductive models of user preferences for semantic web. In J. Pokorný, V. Snášel, and K. Richta, editors,* DATESO 2007*, volume 235 of* CEUR Workshop Proceedings*, pages 108–119. Matfyz Press, Praha, 2007.*

12. *A. Eckhardt, T. Horváth, D. Maruščák, R. Novotný, and P. Vojtáš. Uncertainty issues in automating process connecting web and user. In P. C. G. da Costa, editor,* URSW '07 Uncertainty Reasoning for the Semantic Web - Volume 3*, pages 97–108. The 6th International Semantic Web Conference, 2007.*
13. *A. Eckhardt, T. Horváth, and P. Vojtáš. Learning different user profile annotated rules for fuzzy preference top-k querying. In H. Prade and V. Subrahmanian, editors,* International Conference on Scalable Uncertainty Management*, volume 4772 of* Lecture Notes In Computer Science*, pages 116–130, Washington DC, USA, 2007. Springer Berlin / Heidelberg.*
14. *A. Eckhardt, T. Horváth, and P. Vojtáš. PHASES: A user profile learning approach for web search. In T. Lin, L. Haas, R. Motwani, A. Broder, and H. Ho, editors,* 2007 IEEE/WIC/ACM International Conference on Web Intelligence - WI 2007*, pages 780–783. IEEE, 2007.*
15. *A. Eckhardt and P. Vojtáš. Considering data-mining techniques in user preference learning. In* 2008 International Workshop on Web Information Retrieval Support Systems*, 2008.*
16. *R. Fagin. Combining fuzzy information from multiple systems. pages 216–226, 1996.*
17. *J. Fürnkranz and E. Hüllermeier. Pairwise preference learning and ranking. In* In Proc. ECML-03, Cavtat-Dubrovnik*, pages 145–156. Springer-Verlag, 2003.*
18. *D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry.* Commun. ACM*, 35(12):61–70, 1992.*
19. *S. Holland, M. Ester, and W. Kiessling. Preference mining: A novel approach on mining user preferences for personalized applications. In* Knowledge Discovery in Databases: PKDD 2003*, pages 204–216. Springer Berlin / Heidelberg, 2003.*
20. *E. Hüllermeier and J. Fürnkranz. Learning preference models from data: On the problem of label ranking and its variants. In G. D. Riccia, D. Dubois, R. Kruse, and H. Lenz, editors,* Preferences and Similarities*, pages 283–304. Springer-Verlag, 2008.*
21. *T. Joachims. Optimizing search engines using clickthrough data. In* KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM Press.*
22. *T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In* SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 154–161, New York, NY, USA, 2005. ACM.*
23. *G. K. Eigentaste: A constant time collaborative filtering algorithm.* Information Retrieval*, 4:133–151(19), July 2001.*
24. *W. Kiessling. Foundations of preferences in database systems. In* VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 311–322. VLDB Endowment, 2002.*
25. *W. Kiessling and G. Köstler. Preference sql: design, implementation, experiences. In* VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 990–1001. VLDB Endowment, 2002.*
26. *S. E. Middleton, N. Shadbolt, and D. D. Roure. Capturing interest through inference and visualization: Ontological user profiling in recommender systems. In* K-CAP2003*, 2003.*
27. *A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl. Getting to know you: learning new user preferences in recommender systems. In* IUI '02: Proceedings of the 7th international conference on Intelligent user interfaces*, pages 127–134, New York, NY, USA, 2002. ACM.*
28. *G. H. Wright. The logic of preference reconsidered. In* Theory and Decision*, volume 3, pages 140–169, 1972.*