Semantic Preference Retrieval for Querying Knowledge Bases

Christian Scheel
scheel@dai-lab.de

Alan Said
alan@dai-lab.de

Sahin Albayrak
sahin@dai-lab.de

DAI-Labor
Technische Universität Berlin, D-10587, Germany

ABSTRACT

This work deals with the problem of automatically creating semantic queries for knowledge bases from preference feedback. Semantic knowledge bases are a good source for retrieving entities for item recommendation. We show that preference decisions are not only based on entities, but also on their corresponding predicate-object relations. By extracting the weights from trained preference models, the weighted predicate-object relations can be stored to a user model. The objective is to use such prototype entities in a general user model to formulate semantic queries for recommendation retrieval.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Query formulation*

General Terms

Algorithms, Performance, Experimentation, Human Factors

Keywords

Semantic Query, Preference Model

1. INTRODUCTION

In semantic knowledge bases like Freebase[1] and DBpedia[2] structured information about entities is stored. Because these bases are maintained continuously, they are a reasonable source for recommenders to retrieve candidate sets for recommending items. For personalized recommendations it is necessary to retrieve candidate sets which match the user's interests. Therefore two problems have to be addressed. First, there is the problem of retrieving user's interests and second, there is the need of automatically creating queries for accessing such knowledge bases.

[1]`http://www.freebase.com`
[2]`http://dbpedia.org`

For item recommendations, a description of why items are good recommendations is one of the most desirable type of feedback which can be retrieved. This description can be stored in a user profile and can later be used to search for the best matching items. Hence, if the user provides all necessary information, the recommendation task is a search task. The problem is that users often are not willing to provide such feedback.

In this work, we apply personalized preference learning on semantically represented entities, extract weighted predicate-object relations from the trained model and use this information to construct semantic queries to search for items which can be recommended to the user.

In our approach, we first collect preference feedback between items of the same kind. Then, items are represented only with information coming from a knowledge base. It is assumed that the reasons for the collected preference decisions are based on the known predicate-object relations of the items. We therefore apply preference learning to be able to assign weights to these predicate-object relations. These weighted relations can be stored to a user profile and can later be used to formulate queries to access knowledge bases. We show that preference models trained on items which are only represented by semantic features which can be found in semantic knowledge bases perform good enough to extract weighted predicate-object relations from the trained model. Additionally we present a solution for using this information to automatically query semantic knowledge bases for recommended items.

2. RELATED WORK

Models for personalized preference learning, as well as semantic user models are commonly used in the field of recommender systems and other related information retrieval tasks, e.g. [3, 13]. Creating these models with semantic entities can be done by applying graph kernels for structured data to machine learning algorithms [12]. Preference learning and top-n recommendation both attempt at identifying the relevance of items. Top-n recommendation is the process of finding (and ranking) the top n items which should be of interest to the user [8]. Items should be ranked in order of interest to the user, creating a item ranking sub-problem [8]. Commonly, recommender systems are classified into one of three categories depending on how the recommendation process works [1],

- For content-based recommendations (or semantic recommendations), candidate items are identified by their similarity to items already consumed by the user.

- Collaborative filtering-based recommender systems find relevant recommendations by looking at what similar users have consumed.

- Hybrid recommendations utilize methods from both content-based as well as collaborative filtering-based recommendation in order to find the most suitable items.

Content-based recommenders were perhaps most popular in the early days of online recommender systems [8], are today often used in cold start[3] settings where too little information is available to apply collaborative filtering. Initially the method was used in information retrieval [2].

Among the first collaborative filtering recommendation systems, the Grundy system, created user profiles (so-called stereotypes) in order to compare these and find relevant items [17]. More recent approaches include Rashid et al. [16] and their study of several collaborative techniques and how they compare in terms of recommendation for new users.

Arguably most common today – hybrid recommenders, come in a wide variety, e.g. demographic data-based recommendation [6], item meta-data based-recommendations [18], etc. Most systems offering recommendations today know more about the user than just their consumption history, making hybrid extensions relatively simple and at the same time boosting recommendation quality considerably [11]. Additionally, context-aware recommender (structurally a sub-set of hybrid recommenders), take information about the users' current situation into consideration when generating recommendations. Cebrian et al. [5], for instance, use time as a contextual feature in order to generate better music recommendations.

Collecting user preferences in web search can be done implicitly [9, 15]. When recommended items are presented in list form, this approach can be applied. The alternative of collecting explicit preference feedback by presenting two items, where the user has to choose the preferred one can be applied, too.

3. SEMANTIC USER MODELS

In this section we propose a semantic user model which is able to construct personalized semantic queries by collecting preference decisions from users. We show how entities should be represented to train preference models and how to use the trained models to retrieve weighted predicate-object relations which help formulating semantic queries.

3.1 Structure of the Semantic User Model

Let S be a candidate set of items of the same type, for instance a set of persons, movies, animals, books, etc. The proposed semantic user model will retrieve item recommendations from S by limiting S to only those entities matching retrieved criteria.

In the sense of semantic representation of an element $s \in S$ is a collection of triples, where all subject-predicate-object relations refer to the same subject identifier. It is also possible that objects are transitively connected to the subject. Because all elements in S are of the same type, they can be described by the same attributes. The entities share the same semantic predicates and often even the same predicate-object relations.

[3]Cold start refers to the concept of new users or system which do not have any information about their users

For each predicate-object relation, a list of entities can be retrieved which share this predicate-object relation. Furthermore, such a list can be retrieved for more than just one predicate-object relation. For instance in Freebase[4] a query can be formulated for retrieving all movies directed by Christopher Nolan with actor Christian Bale. It is possible to access any knowledge base by formulating similar semantic queries to the example in Figure 1. A semantic query is a collection of limiting properties. Nevertheless this kind of query is too complex for users to formulate, but could be formalized automatically.

To create a semantic query automatically, the focus lies on the limiting properties of a query. The property limiting the result list to a given type (compare with Figure 1 line *2*) is already given by the type of the elements in S. Hence, only the further limiting properties (compare with Figure 1 line *3* and *4*) have to be retrieved somehow to formulate such a query automatically.

When such weighted predicate-object relations can be retrieved, we propose to store this information into the semantic user model, such that semantic queries can be formulated automatically.

3.2 Retrieving and Representing Entities

The base for training data and candidates for recommendation is S. We therefore retrieve semantic representations of entities of the same type by formulating a semantic query limiting the result list to a given type. For each element in this result list the complete semantic representation has to be retrieved.

As stated, each $s \in S$ is a collection of triples. To train preference models this representation is not suitable. Most machine learning algorithms require a feature vector representation of entities. Any semantic representation has to be transformed to match this representation. Additionally each index in each representing feature vector must always refer to the same attribute.

For transforming such semantic representation to a feature vector representation, the properties in all triples in S have to be collected, first. We refer to the set of selected properties with P. For better performance while training the preference models, we suggest to manually select the properties in P. For instance for movies any entity could be represented with the properties director, producer, actor, release date, genre, and production company.

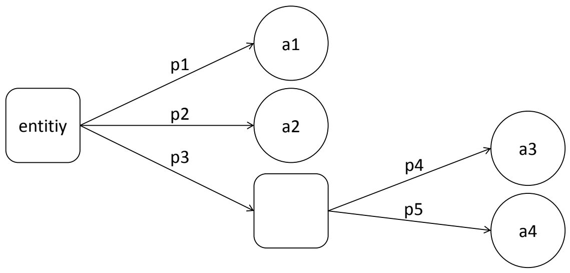
Most of these properties are not real-valued and hence have to be converted to such representation. If a property is a number, then this property can directly be taken as one element in the feature vector. While a date value can also easily be represented as a number (time since a fixed point in time), other properties might have several different characteristics. For instance there can be many persons who have acted in the same movie. In such a case all found property-object relations have to be taken as different features. Such Boolean features, representing presence and absence of characteristics, can be represented as the numeric values 1 and 0.

Figure 2(a) illustrates an entity consisting of five triples. While the predicate-object relations p1-a1 and p2-a2 are directly connected to the entity attributes a3 and a4 are transitively connected to the entity. To represent the en-

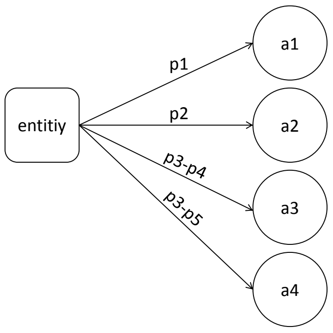
[4]`http://www.freebase.com/queryeditor`

```
[{
  "type":"/film/film",
  "directed_by":[{"id":"/en/christopher_nolan"}],
  "starring":{"actor":{"id":"/en/christian_bale"}}
}]
```

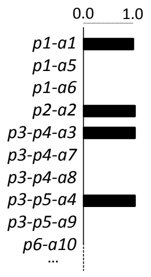
Figure 1: MQL query for all movies directed by Christopher Nolan including actor Christian Bale.



(a) Entity with transitively connected attributes.



(b) Same entity represented with property chains.



(c) Feature vector representation.

Figure 2: Entity represented by semantic relations. By directly connecting objects (attributes) to the subject (entity) it is possible to flatten the structure so that the object can be represented as a feature vector. Because the existence of a predicate-object relation is a Boolean value, it is represented as 1.0. The non-existence is represented as 0.0. The transformation from (a) over (b) to (c) can be reversed.

tity directly with the transitively connected attributes, the predicates can be expressed as predicate chains as can be seen in Figure 2(b). Exemplary for movies predicate p1 could be 'genre', p2 'subject', p3 'starring', p4 'actor', and p5 'role'. Given that these attributes have several different characteristics, for instance $p1 \mapsto \{a1, a5, a6\}$, the corresponding feature vector elements would be $"p1 - a1" : true$, $"p1 - a5" : false$, $"p1 - a6" : false$, $"p2 - a2" : true$, $"p3 - p4 - a3" : true$, and $"p3 - p5 - a4" : true$.

3.3 Personalized Preference Models

Preference models usually serve to predict preferences between items. In [7] a preference model was trained with item pairs and its relation, but for the ranking process, this model only depends on items. For each item, a utility function computes a score, which serves as sorting criteria. Such a utility function $f : W \times S \to \mathbb{R}$ is often used in learning to rank, to keep the computational costs low. For any item $s_1 \in S$ and $s_2 \in S$, applies that $f(s_1) > f(s_2) \iff s_1 \succ s_2$.

An alternative to a utility function is the pairwise prediction of preferences. If there is a set of items, then preference models can predict the preferences between these items, to result in a ranked list of items.

In contrast to the ability to rank items we propose to train preference models only for the purpose of extracting weights. All items are uniformly represented as feature vectors where each element represents a property-object or a property-value relation. The objective of extracting weights is to assign these weights to the semantic relations.

Collecting Preferences.

Collecting preferences is crucial for the whole process of training preferences and retrieving weighted predicate-object relations which can be stored into the user model. Hence, there will always be a cold start problem.

Feedback in the context of recommendations is often based on ratings. These ratings can be binary, e.g. “item have been used” or numeric e.g. a star rating. Two items which have been rated differently in the same context can be used to express a preference relation. This retrieval of preference pairs is necessary when working with benchmark datasets like Letor, where relevance judgments are numeric [14]. In this scenario, the same context refers to the same query.

A more natural way to retrieve preferences is to present two items and let the user choose the preferred one. Such explicit feedback is considered high quality feedback, because such preference decisions are most likely correct.

In contrast to explicit feedback, there is the possibility to extract preference relations implicitly from clickthrough data [9, 15]. The proposed rules result in preference relations with minor quality (80.8% to 84.1% correct preference relations [10]).

The collected preferences can be used to train preference models. A preference consists of two items and their preference relation. The items are represented with feature vectors where all included elements are context independent.

Creating Preference Models.

For training, we suggest to use a support vector machine (SVM) [7] with linear kernel, even though any linear regression approach could be applied. In general, training is done by learning the differences between items. Each preference in the training set consists of two items represented as feature vectors and a corresponding label referring to the relation "first item is preferred" ($\succ$) or "second item is preferred" ($\prec$). By subtracting the second vector from the first, the resulting vector reflects their differences. Some of these differences may point to the reason for the found preference which are noted as one of the two classes $\succ = 0$ and $\prec = 1$.

Formally, the subtraction of the vectors is retrieved from the utility function $s_1 \succ s_2 \iff f(s_1) > f(s_2) \iff \omega \bullet s_1 > \omega \bullet s_2 \iff \omega \bullet (s_1 - s_2) > 0$.

While training, the model is adapted to differentiate between instances of these two classes. After training, the classifier can predict the preference relation for two items.

Weighted Predicate-object Relations.

Taking a SVM with linear kernel allows us to extract weights ω from the trained model. These weights can be used as linear regression by computing the scalar product. The weights for each feature in the feature vector are directly connected to the importance of the features to the preference decisions.

The feature vectors in the training set can be restored to the former semantic representation. When doing the same transformation with the extracted weight vector instead of a feature vector, a prototype entity is created with weighted predicate-object relations. By removing zero-valued weights, this prototype entity can represent a pattern for finding the most preferred entity. These weighted predicate-object relations can be stored to the proposed user profile.

4. EXPERIMENTS AND EVALUATION

In this evaluation it is shown that preference models can be trained on semantic entities. Good performing models are the precondition to use these models for constructing queries.

4.1 Data

The experiments have been conducted on movie data. For the 250 top rated movies of IMDb (in march 2012) the Freebase entries have been retrieved. We have chosen popular movies to increase the probability that users have seen movies from this collection. Then users provided preference feedback for these movies.

The data used in our experiments was collected through an online user study where users were asked to provide preference feedback. Preference feedback was collected by presenting five movie poster thumbnails along with mouse over information about the title, director and actors. Users had to remove unknown movies before providing feedback. Afterwards they had to sort the remaining movies in the order of their preferences.

A list of five movies $[m_1, m_2, m_3, m_4, m_5]$ can be split into the ten preference relations $m_1 \succ m_2$, $m_1 \succ m_3$, $m_1 \succ m_4$, $m_1 \succ m_5$, $m_2 \succ m_3$, $m_2 \succ m_4$, $m_2 \succ m_5$, $m_3 \succ m_4$, $m_3 \succ m_5$, $m_4 \succ m_5$. A list of four movies still leads to six preferences.

In total, 23 users provided feedback for 5173 movie preferences. The user base consists of scientists and massively multiplayer online game players. The annotator agreement for the 1134 preference relations coming from different users, but include the same movies is only 54%. Furthermore, the agreement for the 404 relations rated twice by the same users is 88%.

4.2 Experimental Setup

In the first part of the experiment each user data is evaluated individually. For each user all preference pairs are retrieved from the collected data. Each movie in such a preference pair is represented as a feature vector as shown in Figure 2. A set consisting of 50 randomly selected preferences is taken to perform a five-fold cross validation to evaluate the performance of a SVM with linear kernel. Hence this set is split into training set, test set and validation set. In each step of this performance evaluation 50 user preferences are added to the set, until all preferences have been taken or until the set consists of 450 preferences. There are four users with at least 450 collected training samples, meaning that the last evaluation is averaged by 4. At set size 50, the evaluation is averaged by 23. In the second part of the experiment all user data is evaluated, accordingly. Because training samples for each step are chosen randomly, the first and second part is repeated 10 times to average the evaluated performance.

The performance is measured with the prediction error. This error describes the percentage of wrongly predicted preference pairs on the test sets.

The quality of the general preference model can be compared with the mean of all individual models. The general model is evaluated with the preference feedback of all users, while the mean personalized model is the mean of individual trained user data. Users did not provide the same amount of feedback. The quality of the mean personalized model at any training set size is averaged only on known performance values at this point.

Although it depends on the experiments, the overall objective of creating semantic queries automatically is not part of the experiments and is addressed in the discussion.

4.3 Results

Figure 3 shows the performance of the trained preference models. Starting with small training sets of 50 training samples, the general model shows a prediction error of 40.2%, while the mean prediction error of all users is 29.8%. The lowest individual prediction error at this point was 15.8% and the highest 34.6%. The prediction errors of both models decrease with raising number of training samples. In the shown range, the error of the general model never falls under the starting performance of the mean personalized model. If all data is trained, the general model results in a prediction error of 33.3% what is still higher than the mean individual starting performance. The best performance value of the mean personalized model is 14.5%.

4.4 Discussion

In this section we discuss the found results under three aspects. First, the performance of the general and the individual models is discussed and second, the approach of creating semantic queries (see Section 3) is adapted to the general model and third, arising limitations and resulting requirements are discussed.

General Model and Individual Models.

We have shown that the mean personalized model performs better than the general model for any training set size. The best performance value of the mean personalized model is close to the agreement level of preference relations rated twice by same users (see Section 4.1). The performance

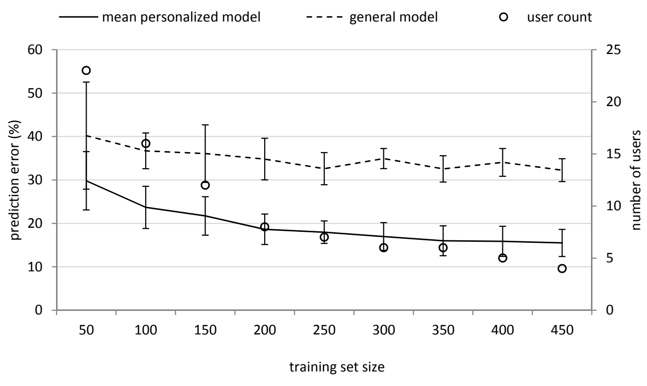


Figure 3: Model quality depending on the size of the training set. The prediction error was evaluated with five-fold cross validations.

of the general model is also better than the inter annotator agreement for preference relations coming from different users, but include the same movies. In the collected data this agreement was 54%, while the trained general model correctly classified 66.7% of all preferences. Both, the inter annotator agreement and the performance of the general model imply that preference models should be trained individually.

Analysis of Weighted Predicate-subject Relations.

From the trained preference models the linear weights can be extracted. As shown, personalized preference models perform better than the general model. The low prediction errors even for small training sets reflect its quality. Hence, the weights must reflect the reasons for retrieved preference decisions. Table 1 shows the highest absolute weights and corresponding features of the general preference model.

Because the weights reflect the importance of the features to the preference decisions, the table can be interpreted as a partial list of the most important and least important features. The features are directly retrieved from semantic predicate-object and predicate-value relations. Hence, this importance can be applied to these semantic structures and can be stored in a user profile.

It is possible to transfer the weight vector into a semantic query. Because the data is coming from Freebase, a query written in MQL can be created automatically (see Figure 4). This figure shows an automatically created MQL query retrieved from the weight vector of the trained general model (compare with Table 1). The '...' in line 11 stands for 5189 removed lines. Besides its size, this query is too restrictive to retrieve any movie. Hence, only parts should be combined to query Freebase. Unfortunately it is not possible to pass the found weights, but negative weights can be respected with a forbidden flag.

Strengths and Limitations.

The user study was used to show that the whole process of retrieving, learning and predicting preferences on semantic data is possible. This ability was a precondition to create semantic queries automatically. To confirm the achieved prediction errors, the user study is still in progress.

The constructed query could be used to access knowledge bases, but knowledge bases currently cannot handle queries of this size and complexity. In information retrieval ranking functions with weighted signals are already commonly used in commercial search engines [4]. For any knowledge base it should be possible to assign weights to the query parts in the same way. As workaround, only the top-k predicates can be used to query the knowledge base or each of these predicates can be used to get result sets which are merged in a post-processing step according to the predicate weights.

5. CONCLUSION

This work is based on preference data which was collected in a user study. Users were asked to sort short lists of movies, according to their preferences. Each movie was represented by semantic preference-object relations found in Freebase. The Freebase representation was transformed to a feature vector representation, which can be reversed to the original structure. The collected preference data consisting of the retrieved feature vectors was trained with a support vector machine with linear kernel, to be able to extract the weights for each preference-object relation.

It was shown that models trained on context-independent features from Freebase are able to predict user preferences. According to the collected data, it is better to train preference models individually as to train a general model. No suggestion for the optimal number of preference feedback for training could be found, but individually trained preference models on 50 collected preferences resulted in good performance values.

Because the trained preference models performed well, the extracted weights can be taken to create semantic queries automatically. Therefore the weight vector was transformed to its corresponding Freebase representation. Such a prototype entity with weighted preference-object relations can be stored to a user model and can be used for creating semantic queries. We have exemplary shown the creation of an MQL query for querying Freebase and discussed necessary steps to achieve that such a query could actually lead to personalized item recommendations. Currently, there is the limitation that no query language for semantic knowledge bases supports weighted predicate-object relations and that applying too many constraints to a query leads to a computational expensive search process.

6. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005.

[2] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[3] M. Balabanović. Exploring versus exploiting when learning user models for text recommendation. *User Modeling and User-Adapted Interaction*, 8:71–102, 1998. 10.1023/A:1008205606173.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, Apr. 1998.

[5] T. Cebrián, M. Planagumà, P. Villegas, and X. Amatriain. Music recommendations with temporal context awareness. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 349–352, New York, NY, USA, 2010. ACM.

Table 1: Predicate-object relations with largest and smallest weights as learned from all collected data.

feature	predicate	object	weight
"/film/film/rating":["id":"/en/r_usa"]	rating	Restricted, R	25
"/film/film/genre":["id":"/m/03btsm8"]	genre	Action & Adventure	22
"/film/film/language":["id":"/en/english"]	language	English language	17
"/film/film/genre":["id":"/en/film_adaptation"]	genre	Film adaptation	16
"/film/film/genre":["id":"/en/airplanes_and_airports"]	genre	Aviation/Airport theme	13
"/film/film/distributors":["distributor":["id":"/en/warner_bros"]]	distributor	Warner Bros. Pictures	13
...	...	...	...
"/film/film/rating":["id":"/en/g_usa"]	rating	General Audiences, G	-12
"/film/film/genre":["id":"/en/world_cinema"]	genre	World cinema	-12
"/film/film/language":["id":"/en/french"]	language	Frensh language	-14
"/film/film/genre":["id":"/en/romance_film"]	genre	Romance	-14
"/film/film/genre":["id":"/en/black-and-white"]	genre	Black-and-white	-18

```
[{
 "type":"/film/film",
 "/film/film/rating":[{"id":"/en/r_usa"}],
 "/film/film/genre":[{"id":"/m/03btsm8"}],
 "/film/film/language":[{"id":"/en/english"}],
 "/film/film/genre":[{"id":"/en/film_adaptation"}],
 "/film/film/genre":[{"id":"/en/airplanes_and_airports"}],
 "/film/film/distributors":[{"distributor":[{"id":"/en/warner_bros"}]}],
...
 "/film/film/rating":[{"optional":"forbidden","id":"/en/g_usa"}],
 "/film/film/genre":[{"optional":"forbidden","id":"/en/world_cinema"}],
 "/film/film/language":[{"optional":"forbidden","id":"/en/french"}],
 "/film/film/genre":[{"optional":"forbidden","id":"/en/romance_film"}],
 "/film/film/genre":[{"optional":"forbidden","id":"/en/black-and-white"}]
}]
```

Figure 4: MQL query from the weight vector retrieved from the trained general model (compare with Table 1).

[6] M. A. Ghazanfar and A. Prugel-Bennett. A scalable, accurate hybrid recommender system. In *Proceedings of the 2010 Third International Conference on Knowledge Discovery and Data Mining*, WKDD '10, pages 94–98, Washington, DC, USA, 2010. IEEE Computer Society.

[7] R. Herbrich, T. Graepel, and K. Obermayer. Support Vector Learning for Ordinal Regression. In *In International Conference on Artificial Neural Networks*, pages 97–102, 1999.

[8] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, Jan. 2004.

[9] 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.

[10] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in Web search. *ACM Trans. Inf. Syst.*, 25(2), 2007.

[11] R. Krestel and P. Fankhauser. Personalized topic-based tag recommendation. *Neurocomput.*, 76(1):61–70, Jan. 2012.

[12] U. Lösch, S. Bloehdorn, and A. Rettinger. Graph kernels for rdf data. In E. S. et. al., editor, *Proc. of the 9th Extended Semantic Web Conference (ESWC'12)*. Springer, Mai 2012. Best Research Paper Award of ESWC 2012.

[13] T. Plumbaum, S. Wu, E. W. De Luca, and S. Albayrak. User Modeling for the Social Semantic Web. In *2nd Workshop on Semantic Personalized Information Management: Retrieval and Recommendation*, 2011.

[14] T. Qin, T.-Y. Liu, J. Xu, and H. Li. LETOR: A Benchmark Collection for Research on Learning to Rank for Information Retrieval. *Information Retrieval Journal*, 13(4):346–374, 2010.

[15] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 239–248, New York, NY, USA, 2005. ACM.

[16] 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 *Proceedings of the 7th international conference on Intelligent user interfaces*, IUI '02, pages 127–134, New York, NY, USA, 2002. ACM.

[17] E. Rich. User modeling via stereotypes. *Cognitive Science*, 3(4):329 – 354, 1979.

[18] R. Wetzker, C. Zimmermann, C. Bauckhage, and S. Albayrak. I tag, you tag: translating tags for advanced user models. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 71–80, New York, NY, USA, 2010. ACM.