Personalizing Keyword Search on RDF Data

Giorgos Giannopoulos[1,2,⋆], Evmorfia Biliri[1], and Timos Sellis[3]

[1] School of ECE, NTU Athens
[2] IMIS Institute, "Athena" Research Center
[3] RMIT University, Australia

Abstract. Despite the vast amount on works on personalizing keyword search on unstructured data (i.e. web pages), there is not much work done handling RDF data. In this paper we present our first cut approach on personalizing keyword query results on RDF data. We adopt the well known Ranking SVM approach, by training ranking functions with RDF-specific training features. The training utilizes historical user feedback, in the form of ratings on the searched items. In order to do so, we join netflix and dbpedia datasets, obtaining a dataset where we can simulate personalized search scenarios for a number of discrete users. Our evaluation shows that our approach outperforms the baseline and, in cases, it scores very close to the ground truth.

1 Introduction

Web search personalization is a well known IR problem that has been tackled by a great amount of works over the years. It lies in changing users' search result list (re-ranking, filtering or suggesting new results) so that the presented results fit their specific information needs. There have been studies on user search behavior and feedback [1], [2], as well as works introducing or improving machine learning techniques for learning ranking functions [1], [2], [4]. Researchers have focused on exploiting users' short/long-term search history [5], [6], context [3], [5] or applying collaborative techniques to personalize search results on users with common interests/search intents [6], [7].

In this paper, we extend the classical setting of personalizing keyword search results on unstructured data, into the scenario where the searched entities are structured, related and organized under a common schema (e.g. RDFS or OWL). Consider the following motivating example: A user wants to search about movies related to Woody Allen in DBpedia, a large RDF dataset. Then, she would pose a keyword query of the form $Q = \{film, woody\ allen\}$. However, the user is a big fan of Scarlett Johansson and this is reflected in her search/rating history, where she has searched and clicked results about Johansson or has consistently rated movies of Johansson with high scores. So, a personalized result list should contain on the top positions results for Woody Allen films that somehow involve Johansson. The problem is not trivial, since:

⋆ This research has been co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.

(a) Relations between searched entities and the schema that characterizes them imposes the construction of new, structure and schema specific training features, beyond the classical IR features. Defining training features is a critical part of the personalization process because it is through them that the quality of a result is *quantified* and *assessed*, w.r.t. the specific user information needs and search history.

(b) While a result list in the classical setting consists of autonomous documents, this does not apply in the setting of keyword search on RDF, where a result may consist of several entities, **along with** the relations that connect them. Thus, issues of combining partial entities to form a complex (graph) result or computing a representative personalization score out of partial scores from the respective entities have to be handled.

In this work, we focus on the first issue, of defining novel, RDF-specific training features and applying the Ranking SVM Model [2] on the RDF search scenario. The experimental results show that our method improves the baseline ranking method and, in cases, performs very close to the ground truth. We note that the second issue is still a subject of our ongoing work.

The paper is organized as follows. Dataset analysis is presented in Section 3. We present our method for training RDF-specific ranking functions and combining them with a keyword search engine to retrieve personalized RDF entities, in Section 4. Section 5 reports on the evaluation. Section 2 reviews related work and Section 6 concludes.

2 Related Work

To the best of our knowledge, most existing works either handle implicitly/indirectly the problem of ranking function training on RDF data or build memory based models to personalize search on RDF. Our work is the first one to propose a model based approach.

The most relevant to our work is [8], where user profile is constructed as a snippet of the knowledge graph and the ontological facts of the knowledge base are utilized to propagate scores from user-accessed entities and facts, thus creating a probabilistic personalized ranking model. In [10] the authors use spreading activation techniques combined with classical search in order to improve the search process in a certain semantic domain. In [12] a statistical approach is applied to learn a user ontology model from a domain ontology. Spreading activation is used for inferencing in the user ontology.

In [9] the authors address the semantic query suggestion task and automatically link queries to DBpedia concepts. Relevant concepts are retrieved for the full query and for each n-gram in the query and then supervised machine learning methods are used to decide which of the retrieved concepts should be kept and which should be discarded. The approach in [11] uses features which can be grouped into dataset specific or dataset independent features. Dataset specific features are extracted from the RDF graph. Dataset independent features are extracted from external sources like web search engines or N-gram databases.

In [13], an extended set of conceptual preferences is derived for a user based on the concepts extracted from search results and clickthrough data. Then, a concept-based user profile is generated and given as input to a support vector machine to learn a concept preference vector for adapting a personalized ranking function.

3 Dataset Pre-processing and Analysis

To be able to train personalization models, it is crucial to have some knowledge about the user's historical needs and preferences. To extract user feedback, we consider user film ratings from the Netflix dataset. In order to enrich the available information, we find the corresponding DBpedia resources for the films in Netflix. Next, we give a brief introduction of the two datasets.

DBpedia knowledge base is a community effort to extract structured information from Wikipedia. The facts are extracted from Wikipedia infoboxes and stored as RDF triples. The triple subject is a resource and the object can be either a resource or a string literal. The DBpedia version we used is characterized by the following statistics: It describes more than 3.64 million things, 1.83 million of which are classified in a consistent ontology. It includes 416,000 persons, 526,000 places, 106,000 music albums, 60,000 films, 17,500 video games, 169,000 organizations, etc. Its ontology constists of 320 ontology classes, 750 object properties and 893 datatype properties. Netflix dataset contains more than 100 million datestamped movie ratings performed by anonymous Netflix users between Dec 31, 1999 and Dec 31, 2005. This dataset gives ratings about 480,189 users and 17,770 movies.

We joined the two datasets in order to exploit the user feedback in the form of movie ratings from Netflix and the RDF structure schema of DBpedia. The join is performed on the Films of each dataset. For a match to be considered valid, we demand exact string matching of the film's title and year of release. We ended up with 5179 matches.

Along with the available OWL ontology for DBpedia concepts, we used Wikipedia categories which are described using the SKOS vocabulary. Through SKOS categories we can obtain useful information such as the film's genre, topics related to film, filming technique, director, decade of release etc. Similarly, we used the YAGO ontology, also available in the DBpedia dataset. In Section 4 we describe how we utilize the aforementioned ontologies to produce training features.

User selection was made according to the following criteria: (a) Number of films the user has rated, to ensure adequate size of training/testing set, (b) Percentage of rated films that were matched against dbpedia entries against total rated films by the user, to ensure consistency of the user profile, (c) Rating distribution: The ratings are integer numbers from 1 to 5. Users that give consistently high or low ratings are considered outliers and, thus, are excluded. Table 1 presents the final set of users divided into two groups: (a) A1-A14 users with many ratings and (b) B1-B11 users with few ratings.

Table 1. Users' rating statistics

user ID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11
#ratings	2185	2206	1649	1770	1752	1726	1656	2092	2440	1935	1840	1675	1635	1890	200	167	164	180	177	264	154	174	291	513	151
mean rating	2.84	2.68	2.93	2.57	3.18	2.91	3.08	3.1	2.57	2.88	3.15	3.16	2.84	3.04	3.61	3.23	3.48	3.29	2.42	3.13	3.06	3.07	3.05	3.63	3.2
#(rating=5)	17	10	17	17	55	27	16	31	7	29	28	19	17	69	3	0	5	0	0	0	3	2	0	6	2
#(rating=4)	107	75	102	50	89	71	64	81	44	49	72	97	71	66	8	11	10	12	6	8	6	14	11	31	9
#(rating=3)	236	129	140	133	121	105	112	216	149	143	151	156	122	87	25	12	7	12	12	43	12	10	40	59	15
#(rating=2)	64	149	62	124	71	85	114	70	162	103	92	58	96	111	2	7	6	8	8	1	6	7	7	6	4
#(rating=1)	13	78	8	30	14	57	25	20	126	63	25	5	21	45	2	3	4	4	9	0	3	1	0	0	0

4 RDF-Specific Ranking Function Training and Personalization

We apply Ranking SVM to train user personalization models. Because of the restrictions imposed by our limited user feedback, we use a large number of features to describe all the information we were able to infer from the film ratings. We were only able to construct query independent features, since there are no training queries in the dataset, so that relations between queries and results can be quantified and learned by our model. We treat all films rated by a user as results to the single keyword query "film" and their ratings as representatives of the desired rank in which the search engine should present them to the user (user feedback). This way, we simulate a process where the user searches for e.g. movies and clicks on some of the results (since we lack data of this kind). The features used for the Ranking SVM method are related to the user search history and the structure of the RDF graph. Actors and Directors are considered closely related to a user's opinion about a film, even though the training involves only resources of the first kind (Films). In order to achieve this, we construct a set of features that can be applied to all three kinds of resources. Next, we present the categorization of training features that we implemented.

1. Actors based on dbpedia property *starring*. Each actor is represented by a separate boolean feature.
2. Directors based on dbpedia property *director*. Each director is represented by a separate boolean feature.
3. SKOS categories on films. For a film, a feature of this kind has value 1 if the film belongs to the category and 0 otherwise. For actors and directors the feature's value is the number of the category films in which they participated.
4. SKOS categories on actors/directors. For actors/directors feature values are boolean depending on whether they are related to a category. For films, the value is the number of actors/directors that participate in the film and belong to the category.
5. Film genre from imdb. Feature values are calculated in the same way as in 3.
6. Film genre from imdb and skos combined. For films, the feature has boolean values, where 1 means that the film is of the specified genre and/or belongs to the specified skos category. For actors/directors the feature's value is the number of films in which they participated that have value 1 in the feature.
7. Yago classes for films. The features represent all classes of the YAGO ontology that have as direct member any film of our dataset. Feature values are calculated in the same way as in 3.
8. Yago classes for actors and directors. The features represent all classes of the YAGO ontology that have as direct member any actor/director of our dataset. Feature values are calculated in the same way as in 4.
9. Entity's degree: both incoming and outgoing properties of the entity.
10. Total number of *starring/director* properties.

We implement a baseline keyword search engine by indexing resources, classes, properties and literals of our dataset with lucene. We consider as query a comma delimited series of keywords or keyword phrases, where the comma symbol delimits keywords (phrases) meant to search different entities. For example, query $Q = \{film, woody\ allen\}$ means that we search for separate entities about "film" and separate

Algorithm 1. RDF results retrieval for keyword queries

```
for each keyword $K_i$ do
    Retrieve the full result list $RL_i$ of size $N_i$ and the respective scores $S_{ij}$ for each result $r_{ij}$
    Normalize scores $S_{ij}$ in the interval $[0, 1]$
    For each result $r_{ij}$ search whether the rest keywords $K_k$ $k \neq i$ are found in its abstract textual description. If so, double its score.
    For each result $r_{ij}$ search whether it has been retrieved using the label property of the entity. If so, double its score.
    Input the final result list into RSVM and retrieve the personalized, re-ranked list $PRL_i$
    Prune the result list of each keyword according to a given threshold.
end for
```

entities about "woody allen". For each of the keywords (keyword phrases), multiple resources are retrieved as possible results to the user's search intention. Algorithm 1 describes the retrieval process, that incorporates the personalized re-ranking of retrieved entities.

5 Evaluation

In this section we compare the effectiveness of our approach for search personalization as compared (a) to the ground truth given by Film ratings in the dataset and (b) to the baseline entity ranking given by the retrieval engine, **without** personalization. For our experiments, we considered 80% of the available dataset as training set and the rest 20% as test set. We note that the ground truth is given exclusively from the test set, that does not participate in the training process. The evaluated users are given in Table 1.

Figure 1 shows the average rating score of results at each position, for each approach. Graph *ground truth* represents the ideal ranking of the highest rated results at the top, followed by decreasingly rated results. *personalization* gives our ranking and graph *baseline ranking* the ranking of the search engine without personalization.

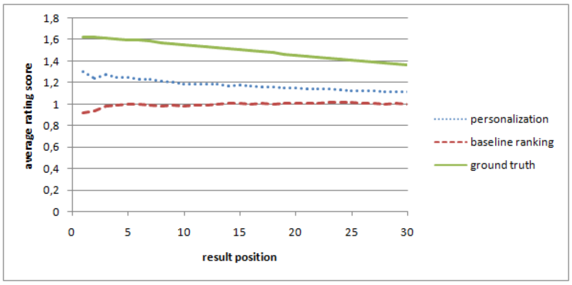

Fig. 1. Average rating scores for each approach

As we can see, our personalization method, first of all, outperforms the baseline ranking, by returning in higher positions more highly rated results. Secondly, the graph is almost constantly decreasing, meaning that there are not significant outliers, e.g. many results with low rating at the top positions, that could cause the graph to suddenly decrease and then increase back. Finally, although the ground truth is consistently better (as expected) than our method, the two graphs demonstrate almost the same behavior, that is, our graph decreases almost in the same way the ground truth does. Part of our ongoing work is to examine whether the size of the training set (i.e. the size of user's feedback) significantly affects the personalization effectiveness, by performing separate analysis for users with many and users with few ratings.

In Figure 2 we show the effect on personalization of training ranking functions with different groups of training features (for clarity, we present two different graphs, due the large amount of feature combinations). Note that this is a preliminary study, aiming to just give an intuition of the effect. That's why we present results for only one random user (B3). A more thorough evaluation is part of our ongoing work.

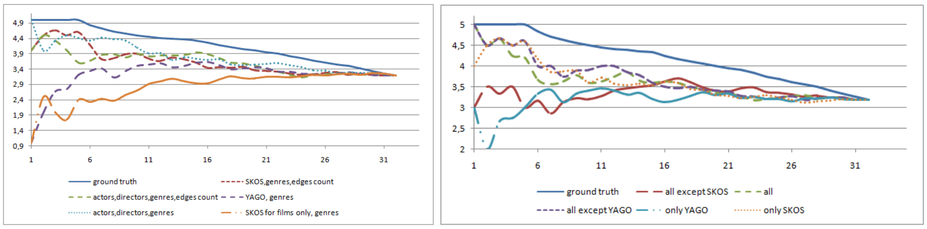

Fig. 2. Effect of different training features use on the personalization

We can see that the performance greatly varies for different feature groups. For example, for this particular user, features representing YAGO classes seem to have a negative effect, although, when **all** features are combined, the YAGO features influence is smoothed and the graph approximates the ground truth graph. On the other hand, using only SKOS features seems, in general, to favor personalization. As future work, we intend to examine if there exists user information that can be inferred from film ratings that can help us predict which features would be more appropriate for each user.

We should note here that the presented results regard Film entities, on which we have explicit user ratings. Similar experiments on ranking Actor and Director entities gave us poor results which are not reported in graphs due to lack of space. Improving the ranking of entities with only implicit feedback available is part of our ongoing work.

6 Conclusion

In this paper, we presented a methodology for personalizing keyword queries on RDF data. We defined a series of RDF-specific ranking function training features and used Ranking SVM to build personalization models. In order to do so, we joined utilized information from two datasets (DBpedia, Netflix) producing a proper dataset to be able to simulate user feedback on keyword search on RDF data. We applied our method on a baseline entity ranking engine and demonstrated its effectiveness.

Our ongoing and future work involves extending the current methods in order to be able to personalize complete graph results and not just separate entities. The challenges are (a) to be able to efficiently combine separate entities in order to obtain meaningful results, (b) to examine which groups of training features and why are more effective and (c) how entities with implicit user feedback can be effectively personalized.

References

1. Agichtein, E., Brill, E., Dumais, S.: Improving web search ranking by incorporating user behavior information. In: Proc. of the ACM SIGIR Conference (2006)
2. Joachims, T.: Optimizing search engines using clickthrough data. In: Proc. of the ACM SIGKDD Conference (2002)
3. Kim, J.-W., Candan, K.-S.: Skip-and-prune: cosine-based top-k query processing for efficient context-sensitive document retrieval. In: Proceedings of the ACM SIGMOD Conference (2009)
4. Qin, T., Zhang, X.-D., Wang, D.-S., Liu, T.-Y., Lai, W., Li, H.: Ranking with multiple hyperplanes. In: Proceedings of the ACM SIGIR Conference (2007)
5. Shen, X., Tan, B., Zhai, C.: Context-sensitive information retrieval using implicit feedback. In: Proceedings of the ACM SIGIR Conference (2005)
6. Sugiyama, K., Hatano, K., Yoshikawa, M.: Adaptive web search based on user profile constructed without any effort from users. In: Proceedings of the ACM WWW Conference (2004)
7. Giannopoulos, G., Brefeld, U., Dalamagas, T., Sellis, T.: Learning to rank user intent. In: Proceedings of the CIKM Conference (2011)
8. Dudev, M., Elbassuoni, S., Luxemburger, J., Ramanath, M., Weikum, G.: Personalizing the Search for Knowledge. In: 2nd PersDB (2008)
9. Meij, E., Bron, M., Hollink, L., Huurnink, B., de Rijke, M.: Learning Semantic Query Suggestions. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 424–440. Springer, Heidelberg (2009)
10. Rocha, C., Schwabe, D., Poggi, M.P.: Hybrid approach for searching in the semantic web. In: Proc. of the 13th International Conference on World Wide Web, pp. 374–383 (2004)
11. Dali, L., Fortuna, B., Duc, T.T., Mladenić, D.: Query-independent learning to rank for RDF entity search. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 484–498. Springer, Heidelberg (2012)
12. Jiang, X., Tan, A.H.: Learning and inferencing in user ontology for personalized Semantic Web search. Information Sciences: An International Journal 179(16), 2794–2808 (2009)
13. Leung, K.W.-T., Lee, D.L., Ng, W., Fung, H.Y.: A Framework for Personalizing Web Search with Concept-Based User Profiles. ACM Transactions on Internet Technology 11(4), Article 17 (2012)