# Using Graph Metrics for Linked Open Data Enabled Recommender Systems

Petar Ristoski(✉), Michael Schuhmacher, and Heiko Paulheim

Research Group Data and Web Science,
University of Mannheim, Mannheim, Germany
{petar.ristoski,michael,heiko}@informatik.uni-mannheim.de

**Abstract.** Linked Open Data has been recognized as a useful source of background knowledge for building content-based recommender systems. While many existing approaches transform that data into a propositional form, we investigate how the graph nature of Linked Open Data can be exploited when building recommender systems. In particular, we use path lengths, the K-Step Markov approach, as well as weighted NI paths to compute item relevance and perform a content-based recommendation. An evaluation on the three tasks of the 2015 LOD-RecSys challenge shows that the results are promising, and, for cross-domain recommendations, outperform collaborative filtering.

**Keywords:** Linked Open Data · Recommender systems · Graph metrics · Cross-domain recommendation

## 1 Introduction

Recommender systems are systems that provide a suggestion of items to a user, based on the user's profile and/or previous behavior. They are used, e.g., for music recommendation in streaming services, in online shopping sites, or on news portals and aggregators. The two major types of recommender systems are *collaborative filtering* and *content-based* recommender systems. The former exploit similarity among *users*, i.e., they recommend items that have been consumed and/or ranked high by users that have similar interests as the user for which the recommendation is made. The latter exploit similarities among *items*, e.g., recommending music of the same genre or news articles on the same topic. Combinations of those approaches, known as *hybrid* approaches, have also been widely studied [1].

In particular for content-based recommender systems, Linked Open Data (LOD) has been shown to be a valuable source of background knowledge. Despite data from various domains being published as LOD [23], particularly cross-domain sources such as DBpedia [13] are primarily used in recommender systems. Given that the items to be recommended are linked to a LOD dataset, information from LOD can be exploited to determine which items are considered to be similar to the ones that the user has consumed in the past. For example,


H. Stuckenschmidt and D. Jannach (Eds.): EC-Web 2015, LNBIP 239, pp. 30–41, 2015.
DOI: 10.1007/978-3-319-27729-5_3

DBpedia holds information about genres of books and music recordings, which can be exploited in recommendation systems [9,21]. Most often, selected data is extracted from DBpedia and transformed into a propositional form, i.e., each graph node is represented by a flat vector of binary and/or numeric features. However, DBpedia contains more information than expressed in those propositional forms. In particular, semantic *paths* between entities are a good candidate for building cross-domain recommender systems.

In this paper, we step away from extracting flat, propositional content features [22] from LOD sources, and consider the graph nature of those sources instead. Specifically, we look at paths between the items as a measure for the similarity of those items. We explore different variants of such path-based similarity measures and contrast them with standard collaborative filtering methods.

The rest of this paper is structured as follows. In Sect. 2, we explicate our overall approach. Section 3 shows how our approach is applied to the three tasks of the 2015 LOD-RecSys challenge[1], i.e., top-N recommendations, diversity, and cross-domain recommendations, and discusses the results. We conclude with a review of related work in Sect. 4 and a short summary in Sect. 5.

## 2 Graph Methods for Recommender Systems

In this paper, we consider three graph-based recommendation approaches. To perform the calculations, we first build an undirected weighted graph, where each item is represented as a node. For our implementation, we use the JUNG java library[2], which also offers implementations of different algorithms from graph theory. Since the domains for the three tasks differ, we use the same set of graph algorithms, but a different graph of items with different edge weights for each of the tasks. The different graph algorithms are described in this section, while the dataset-specific construction of the respective graphs is described in the corresponding parts of Sect. 3.

### 2.1 Shortest Path

To recommend relevant items for each user, we try to find the items in the graph that are closest to those items that were liked by the user, i.e., we assume that *proximity* in the graph is a proxy for *similarity*.

For implementing that approach, let $R$ be the set of items a user liked. Then, for each item $t$ in the test set (i.e., the items from which a recommendation is to be made), we compute the negated sum of shortest path lengths (given the edge weights) for all items in $R$ to $t$ as a ranking score:

$$I(t|R) = -\sum_{i=1}^{|R|} sp(R_i, t), \tag{1}$$

[1] http://sisinflab.poliba.it/events/lod-recsys-challenge-2015/.
[2] http://jung.sourceforge.net/.

where $sp(R_i, t)$ is the shortest path from $R_i$ to $t$, where $R_i \in R$. Then, for each user the test items are sorted based on the relevance $I$ in descending order, and the top-N items are recommended to the user.

### 2.2 K-Step Markov Approach

The PageRank algorithm is frequently used to compute importance for nodes in a graph. The PageRank score of a node can be seen as the probability of visiting that node with a random walk on the graph [16]. The K-Step Markov approach represents an algorithm variant of the PageRank algorithm with priors and computes the importance of any node in a given graph based on a given root set of nodes [28]. More precisely, the approach computes the relative probability that the system will spend time at any particular node in the graph, given that it starts in a root set of nodes $R$ and ends after $K$ steps.

The result is an estimate of the transient distribution of states in the Markov chain, starting from $R$: as $K$ gets larger it will converge to the steady-state distribution used by PageRank, i.e. the standard version of PageRank without priors. Thus, the value of K controls the relative tradeoff between a distribution "biased" towards the root set of nodes $R$ and the steady-state distribution which is independent of where the Markov process started. The relative importance of a node $t$ given a root set $R$ can be calculated using the equation:

$$I(t|R) = \left[A_{P_R} + A^2_{P_R} + \ldots + A^K_{P_R}\right] \tag{2}$$

where $A$ is the transition probability matrix of size $n \times n$, and $P_R$ is an $n \times 1$ vector of initial probabilities for the root set $R$.

In order to create recommendations, we again start with the set of items $R$ which a user likes, and then use the K-Step Markov approach to find the top-N nodes that have the highest stationary probability. For the transition probability we use the edge weights after turning them into proper probabilities.

### 2.3 WeightedNIPaths

For predicting the relevance of an item node for a given user within the graph, we also make use of the WeightedNIPath algorithm [28]. Building upon the Shortest Path approach from Sect. 2.1, the idea is here to consider not only the single shortest path, but to take into account all distinct paths and add a decay factor $\lambda$ to penalize longer paths. Therefore, we compute the number of distinct paths between the source nodes, i.e. all nodes/items that have been rated $r \in R$ by the user, and each other node $t$, i.e. all unrated and potentially to be recommended nodes. Our intuition is that items which are frequently (indirectly) connected to positively rated items, have some content-based connection and should thus get a higher recommendation score by this method.

Formally, for a set of items $R$ liked by a user, and a candidate item $t$, we compute the importance score as

$$I(t|R) = \sum_{r \in R} \sum_{i=1}^{|P(r,t)|} \lambda^{-|p_i|} \tag{3}$$

where $P(r,t)$ is a set of maximum-sized node-disjoint paths from node $r$ to node $t$, $p_i$ is the $i$th path in $P(r,t)$, and $\lambda$ is the path decay coefficient. As before, the top-N items are recommended.

With this approach, movies that e.g. share the same actors and director will be closer related than two movies that have only the director in common. While being similar to the shortest path approach described above, WeightedNIPath takes into account all paths and discounts each edge of a path by a factor, thus penalizing longer paths (we use a discount factor of 3). The rationale here is that the longer a path, the less closely related are the items this path connects. In addition, based on initial experiments, we limit the path length to 2, which is also common practice when working with DBpedia as a semantic network [25].

## 3 Evaluation

We evaluate the item recommendation performance of the three graph algorithms with benchmark data from the Linked Open Data-enabled Recommender Systems Challenge 2015.[3] For this challenge, three training datasets from different domains, i.e., movies, books, and music, are provided. Those datasets were generated by collecting data from Facebook profiles about personal preferences ("likes") for the items. After a process of user anonymization, the items available in the dataset have been mapped to their corresponding DBpedia URIs. An overview of the size of the datasets is given in Table 1.

For all three tasks, each approach was evaluated on an unseen gold standard using an online evaluation system provided by the challenge, and compared to standard collaborative filtering as a baseline.

**Table 1.** Datasets overview

| Dataset | #Items | #Ratings | Task |
|---|---|---|---|
| movies | 5,389 | 638,268 | 1 & 3 |
| music | 6,372 | 854,016 | 2 |
| books | 3,225 | 11,600 | 3 |

### 3.1 Task 1: Top-N Recommendations from Unary User Feedback

In this task, top-N recommendations for the movie domain are to be made. The input is unary feedback (i.e., whether a user likes an item) under open world semantics, i.e., no negative examples (dislikes) are provided. The evaluation is made based on recall, precision, and F-measure for the top 10 recommendations.

[3] http://sisinflab.poliba.it/events/lod-recsys-challenge-2015/.

**Graph Extraction.** The graph we construct consists of some direct relations of the movies, as well as their actors, genres, and characters. To generate the graph, we used the RapidMiner Linked Open Data extension [19,20]. We extracted the following relations:

- **movie:** *rdf:type*, *dcterms:subject*, *dbpedia-owl:starring*, *dbpedia-owl:director*, *dbpedia-owl:distributor*, *dbpedia-owl:producer*, *dbpedia-owl:musicComposer*, *dbpedia-owl:writer*, and *dbpprop:genre*
- **movie_actor:** *rdf:type*, *dcterms:subject*, and *is dbpedia-owl:starring of*
- **movie_genre:** *rdf:type* and *dcterms:subject*
- **movie_character:** *rdf:type*, *dcterms:subject*, *dbpedia-owl:creator*, and *dbpedia-owl:series*

Each DBpedia entity is represented as a node in the graph, where the relations between the entities are represented as undirected edges between the nodes in the graph. We use inverse document frequency (IDF) to weight the edges. For example, if an actor plays in five movies, then the IDF for each of those five relations is $log\frac{1}{5}$.[4]

In order to make the approach work also for rather weakly interlinked resources, we introduce additional edges in the graph based on the abstracts in DBpedia. To that end, we preprocess the abstract, i.e., we convert the abstract to lower case, perform tokenization, stemming, and stop words removal. Then, each token is represented as a node in the graph. Eventually, we use the resulting graph of abstract tokens and genre relations to find paths between entities from the movie domain and entities from the books domain. The relations between the entities and tokens are, as before, represented as undirected edges between the nodes in the graph and edges are also again IDF-weighted as for task 1.

In addition to the graph, we extracted the following *global* (i.e., not user-related) popularity scores:

- Number of Facebook likes[5]
- Metacritic score[6]
- Rotten Tomatoes score[7]
- DBpedia Global PageRank [26]
- Local Graph PageRank: Computed using the *JUNG* java library on the previously generated graph
- Aggregated Popularity: Using Borda's rank aggregation [4], we aggregated all those popularity scores into one.

Furthermore, following the approach in [21], we also use a stacking approach [27] for combining all the recommendations, i.e., collaborative, content-based, and global.

[4] To compute edge weights from IDF, we first normalize the IDF scores to $[0; 1]$, and then assign $1 - IDF_{normalized}$ as a weight to the edges, so that edges with a larger IDF value have a lower weight.

[5] https://www.facebook.com/.

[6] http://www.metacritic.com/.

[7] http://www.rottentomatoes.com/.

**Table 2.** Results for top-N recommendations from unary user feedback (the best results are marked in bold).

| Approach | P@10 | R@10 | F1@10 |
|---|---|---|---|
| User-Based KNN (k=20) | 0.0954 | 0.1382 | 0.1129 |
| User-Based KNN (k=50) | 0.1025 | 0.1485 | 0.1213 |
| User-Based KNN (k=80) | **0.1032** | **0.1493** | **0.122** |
| Shortest Path | 0.0597 | 0.0859 | 0.0704 |
| K-Step Markov Approach (K=4) | 0.0496 | 0.0703 | 0.0581 |
| WeightedNIPaths (H=2) | 0.0151 | 0.0217 | 0.0178 |
| Shortest Path (w abstract) | 0.0525 | 0.0746 | 0.0616 |
| K-Step Markov Approach (K=4) (w abstract) | 0.0562 | 0.0804 | 0.0662 |
| WeightedNIPaths (H=2) (w abstract) | 0.0216 | 0.0309 | 0.0254 |
| Borda's rank aggregation | 0.0572 | 0.0824 | 0.0676 |
| Stacking with polynomial regression | 0.0099 | 0.0137 | 0.0115 |

**Results.** The results are depicted in Table 2. It can be observed that the collaborative filtering based approaches clearly outperform the content-based ones. From the graph-based approaches, the shortest paths are the best performing approach.

As already observed in [21], the global ranking scores, aggregated with Borda's rank aggregation, are a strong competitor to content-based approaches. On the other hand, the stacking approach combining multiple recommendation approaches does not work well. This is in particular due to the fact that only positive evidence is given, which makes it hard to train a regression algorithm.

Due to its bad performance, we have not considered the stacking solution for the subsequent tasks.

### 3.2 Task 2: Diversity Within Recommended Item Sets

The second task is to make recommendations in the music domain. Here, the focus is on *diverse* recommendations, i.e., entities from different genres. The evaluation is made based on the average of F-measure and intra-list diversity (ILD) for the top 20 recommendations.

**Graph Extraction.** To generate the graph, we extracted the following relations:

- **music_artist:** *rdf:type*, *dcterms:subject*, *dbpedia-owl:genre*, *dbpedia-owl:associatedBand*, *dbpedia-owl:associatedMusicalArtist*, *dbpedia-owl:genre*, and *dbpedia-owl:occupation*
- **music_band:** *rdf:type*, *dcterms:subject*, *dbpedia-owl:associatedBand*, *dbpedia-owl:associatedMusicalArtist*, *dbpedia-owl:genre*, and *dbpedia-owl:bandMember*

- **music_album:** *rdf:type*, *dcterms:subject*, *dbpedia-owl:artist*, and *dbpedia-owl:genre*
- **music_composition:** *rdf:type*, *dcterms:subject*, *dbpedia-owl:musicalArtist*, *dbpedia-owl:musicalBand*, and *dbpedia-owl:genre*
- **music_genre:** *rdf:type* and *dcterms:subject*
- **abstract:** *dbpedia-owl:abstract*

As for the previous task, each DBpedia entity is represented as a node in the graph, where the relations between the entities are represented as undirected edges between the nodes in the graph. Like for the previous task, we use IDF to weight the edges.

For making the predictions, with user-based k-NN, we simply predict the top 20 items, as we already observe a high ILD with this approach. For shortest paths and the k-step Markov approach, we follow the approach in [21]: We first generate ranked lists. From those lists, we then pick the top 10 items, and then iteratively fill them up with the next 10 items in the list that do not share a genre with those that are already in the list.

**Table 3.** Results for diversity within recommended item sets (the best results are marked in bold).

| Approach | P@20 | R@20 | F1@20 | ILD@20 |
|---|---|---|---|---|
| User-Based KNN (k=20) | 0.09 | 0.2477 | 0.1321 | 0.9039 |
| User-Based KNN (k=50) | 0.0963 | 0.2649 | 0.1412 | 0.9028 |
| User-Based KNN (k=80) | **0.0973** | **0.2677** | **0.1427** | 0.9032 |
| Shortest Path | 0.0343 | 0.0948 | 0.0504 | 0.8964 |
| K-Step Markov Approach (K=4) | 0.0312 | 0.0863 | 0.0458 | 0.9077 |
| WeightedNIPaths (H=2) | 0.0343 | 0.0933 | 0.0502 | 0.8588 |
| Shortest Path (w abstract) | 0.0077 | 0.0203 | 0.0112 | **0.9717** |
| K-Step Markov Approach (K=4) (w abstract) | 0.0217 | 0.0585 | 0.0317 | 0.9699 |
| WeightedNIPaths (H=2) (w abstract) | 0.0358 | 0.0975 | 0.0523 | 0.8785 |
| Borda's rank aggregation | 0.0356 | 0.0977 | 0.0522 | 0.922 |

**Results.** The results for task 2 are depicted in Table 3. Again, we can see that on average, the collaborative filtering approaches produce the better results in terms of F-measure, with the ILD being comparable. It is furthermore remarkable that the ILD is that high for the collaborative filtering approaches, which were not specifically altered for producing diverse recommendations. The highest ILD score is achieved with the Shortest Path (with abstract), however at the cost of a very low recall, precision, and F1 score.

### 3.3 Task 3: Cross-Domain Recommendation

The third task poses a different setting, as it asks for using feedback (user ratings) from one domain, here movies, to provide recommendations for another domain, namely books. Like for the first task, recall, precision, and F-measure for the top 10 recommendations are used as evaluation metrics.

**Graph Extraction.** To generate the graph, we extract the same features as for the top-N recommendation tasks on movies (see Sect. 3.1), but including in addition the *dbpedia-owl:abstract* for each item. For the book domain, we extract the following relations:

- **book:** *rdf:type*, *dcterms:subject*, *dbpedia-owl:genre*, *dbpedia-owl:author*, and *dbpedia-owl:subsequentWork*
- **book_writer:** *rdf:type* and *dcterms:subject*
- **book_character:** *rdf:type* and *dcterms:subject*
- **book_genre:** *rdf:type* and *dcterms:subject*
- **abstract:** *dbpedia-owl:abstract*

**Results.** The results for task 3 are depicted in Table 4. Here, in contrast to task 1 and 2, we find that the graph-based approaches clearly outperform the collaborative filtering (CF) ones. We also observe that User-based KNN is comparably low, which leads us to the suspicion that the cross-domain nature of this task poses a serious challenge to regular CF approaches – which obviously need to operate on already observed items. In an extreme cross-domain scenario, there would be no historical user preference information available on the items to be ranked and any CF approach would fail. In contrast, the graph-based approaches can apparently find reasonable relations in the graph between books and movies (e.g., common genres, or mentioning of similar terms in the abstract) and leverage those for creating meaningful predictions.

**Table 4.** Results for cross-domain recommendation (the best results are marked in bold).

| Approach | P@10 | R@10 | F1@10 |
|---|---|---|---|
| User-Based KNN (k=20) | 0.0162 | 0.026 | 0.0199 |
| User-Based KNN (k=50) | 0.022 | 0.0353 | 0.0271 |
| User-Based KNN (k=80) | 0.0258 | 0.0416 | 0.0318 |
| Shortest Path | 0.0326 | 0.0539 | 0.0407 |
| K-Step Markov Approach (K=4) | 0.0659 | 0.1077 | 0.0818 |
| WeightedNIPaths (H=2) | 0.0358 | 0.0299 | 0.0227 |
| Shortest Path (w abstract) | 0.0627 | 0.1026 | 0.0778 |
| K-Step Markov Approach (K=4) (w abstract) | **0.078** | **0.1276** | **0.0968** |
| WeightedNIPaths (H=2) (w abstract) | 0.0195 | 0.0314 | 0.024 |
| Borda's rank aggregation | 0.0301 | 0.0493 | 0.0374 |

## 4 Related Work

It has been shown that LOD can improve recommender systems towards a better understanding and representation of user preferences, item features, and contextual signs they deal with. LOD has been used in content-based, collaborative, and hybrid techniques, in various recommendation tasks, i.e., rating prediction, Top-N recommendations and diversity in content-based recommendations. An overview of cross-domain recommender systems is given in [2].

Among the earliest such efforts is *dbrec* [17], which uses DBpedia as knowledge base to build a content-based music recommender system. The recommendations are based on measure (named Linked Data Semantic Distance) which computes the distance between two items in the DBpedia graph, using only the object properties. Heitmann et al. [9] propose an open recommender system which utilizes Linked Data to mitigate the new-user, new-item and sparsity problems of collaborative recommender systems. They first publish an existing music artists database as LOD using the FOAF ontology[8]. Then, they link the data to DBpedia and DBtune MySpace. Using the new connections between the users, artists and items, the authors are able to build a collaborative recommender system.

Di Noia et al. [5] propose a model-based recommender system that relies on LOD, and can use any arbitrary classifier to perform the recommendations. First, they map the items from the local dataset to DBpedia, and then extract the direct property-value pairs. The resulting data is converted to feature vectors, where each property-value pair represents a feature. The work is extended in [6] where the similarities between the items are calculated using a vector space model. In this approach, for each item the direct property-value pairs are extracted from DBpedia, Freebase and LinkedMDB[9], which are represented as a 3-dimensional matrix where each slice refers to an ontology property and represents its adjacency matrix. In [15], the authors present SPrank, a hybrid recommendation algorithm for computing top-N item recommendations from implicit feedback exploiting the information available as LOD. In the approach, the authors try to extract features able to characterize the interactions between users, items and entities capturing the complex relationships between them. To do so, they extract all the paths that connect the user to an item in order to have a relevance score for that item, which are then used as features in the recommender algorithm. In more recent work [14], the authors propose a content-based recommender based on a neighborhood-based graph kernel, which computes semantic item similarities by matching their local neighborhood graphs.

In [21], we present a hybrid multistrategy book recommender system, where we use stacking regression and rank aggregation to combine the results of multiple base recommenders. The features for the recommenders are generated using the previously described RapidMiner LOD extension, from multiple LOD sources. Another approach by Schmachtenberg et al. [24] uses background knowledge from LinkedGeoData, to enhance a location-based recommendation system.

[8] http://xmlns.com/foaf/spec/.
[9] http://www.linkedmdb.org/.

The features for the recommendation system were generated using the FeGeLOD tool [18], the predecessor of the RapidMiner LOD extension.

Furthermore, several cross-domain recommender systems based on LOD data have been proposed in the literature. Fernández-Tobías et al. [7] proposed an approach that uses DBpedia as a cross-domain knowledge source for building a semantic network that links concepts from several domains. On such a semantic network, which has the form of a directed acyclic graph, a weight spreading activation algorithm [3] retrieves concepts in a target domain (music) that are highly related to other input concepts in a source domain (points of interest). The work is extended in [11,12] by finding richer semantic relations between architecture and music concepts in DBpedia.

A similar LOD-enhanced graph-based approach is presented in [8,10]. The approach is based on an enhanced spreading activation model that exploits intrinsic links between entities across a number of data sources.

## 5 Conclusion

In this paper, we have proposed to derive weighted graphs from DBpedia, and apply graph algorithms on it to retrieve item-based recommendations. We studied the usage of three different graph algorithms working on different subgraphs of the DBpedia graph. Our approaches rely on (a) shortest paths, (b) a variant of PageRank with priors (K-Step Markov), and (c) the sum of all distinct, connecting paths (WeightedNIPaths).

We find that in situations where the complete user feedback is available, collaborative filtering outperforms all studied graph-based approaches. In contrast, in situations where user feedback is scarce – here: for making cross-domain predictions – graph-based approaches are a reasonable way to build recommender systems. This observation makes the approaches proposed in this paper an interesting candidate for various settings. For example, in cold start situations, where no ratings for new products exist (yet), they should be included in recommendations. Second, when trying to open new market segments for an existing customer base, such methods can be helpful.

So far, we have considered only DBpedia as LOD source. In future work we can explore the existing *owl:sameAs* links in DBpedia to build richer and denser graphs from domain specific LOD sources, e.g., LinkedMDB[10] for movies, MusicBrainz[11] for music, the British National Bibliography[12] for books, etc. The information retrieved from the domain specific LOD sources should be more accurate and extensive, which should lead to better performance of the recommender systems. Furthermore, when building the graphs only specific relations were included, which we believed were the most relevant for the task. However, the graphs may be built in unsupervised manner, i.e., including all properties for all of the entities, and expanding the graph to several hops. The proposed

[10] http://www.linkedmdb.org/.
[11] https://wiki.musicbrainz.org/LinkedBrainz.
[12] http://bnb.data.bl.uk/.

approaches should still be able to make good recommendations on such graphs, because we use IDF to weight the edges, i.e., the most relevant edges will have a higher weight. This way, we would be able to apply the approaches on data from any domain/s without the need for manual feature engineering.

**Acknowledgements.** The work presented in this paper has been partly funded by the German Research Foundation (DFG) under grant number PA 2373/1-1 (Mine@LOD). Part of this work was performed on the computational resource bwUniCluster funded by the Ministry of Science, Research and the Arts Baden-Württemberg and the Universities of the State of Baden-Württemberg, Germany, within the framework program bwHPC. We would like to thank our colleague Robert Meusel for his valuable contribution to our system.

## References

1. Burke, R.: Hybrid recommender systems: Survey and experiments. User Model. User-Adapted Interact. **12**(4), 331–370 (2002)
2. Cantador, I., Fernández-Tobıas, I., Berkovsky, S., Cremonesi, P.: Cross-domain recommender systems (2015)
3. Allan, M.: Collins and Elizabeth F Loftus. A spreading-activation theory of semantic processing. Psychol. Rev. **82**(6), 407 (1975)
4. de Borda, J.C.: Mémoire sur les élections au scrutin. Histoire de l'Academie Royale des Sciences (1781)
5. Di Noia, T., Mirizzi, R., Ostuni, V.C., Romito, D.: Exploiting the web of data in model-based recommender systems. In: Proceedings of the Sixth ACM Conference on Recommender Systems, RecSys 2012, pp. 253–256. ACM, New York, NY, USA (2012)
6. Di Noia, T., Mirizzi, R., Ostuni, V.C., Romito, D., Zanker, M.: Linked open data to support content-based recommender systems. In: Proceedings of the 8th International Conference on Semantic Systems, I-SEMANTICS 2012, pp. 1–8. ACM, New York, NY, USA (2012)
7. Fernández-Tobías, I., Cantador, I., Kaminskas, M., Ricci, F.: A generic semantic-based framework for cross-domain recommendation. In: Proceedings of the 2Nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems, HetRec 2011, pp. 25–32. ACM, New York, NY, USA (2011)
8. Heitmann, B., Dabrowski, M., Passant, A., Hayes, C., Griffin, K.: Personalisation of social web services in the enterprise using spreading activation for multi-source, cross-domain recommendations. In: AAAI Spring Symposium: Intelligent Web Services Meet Social Computing (2012)
9. Heitmann, B., Conor Hayes, C.: Using linked data to build open, collaborative recommender systems. In: AAAI Spring Symposium: Linked Data Meets Artificial Intelligence (2010)
10. Heitmann, B., Hayes, C.: SemStim at the LOD-RecSys 2014 challenge. In: Presutti, V., et al. (eds.) SemWebEval 2014. CCIS, vol. 475, pp. 170–175. Springer, Heidelberg (2014)
11. Kaminskas, M., Fernández-Tobıas, I., Ricci, F., Cantador, I.: Knowledge-based identification of music suited for places of interest. Inf. Technol. Tourism **14**(1), 73–95 (2014)

12. Kaminskas, M., Fernández-Tobías, I., Cantador, I., Ricci, F.: Ontology-based identification of music for places. In: Cantoni, L., (Phil) Xiang, Z. (eds.), Information and Communication Technologies in Tourism 2013, pp. 436–447. Springer, Heidelberg (2013)
13. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. Seman. Web J. (2013)
14. Ostuni, V.C., Di Noia, T., Mirizzi, R., Di Sciascio, E.: A linked data recommender system using a neighborhood-based graph kernel. In: Hepp, M., Hoffner, Y. (eds.) EC-Web 2014. LNBIP, vol. 188, pp. 89–100. Springer, Heidelberg (2014)
15. Ostuni, V.C., Di Noia, T., Mirizzi, R., Di Sciascio, E.: Top-n recommendations from implicit feedback leveraging linked open data. In: IIR, pp. 20–27 (2014)
16. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical Report 1999–66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120
17. Passant, A.: dbrec — Music recommendations using DBpedia. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part II. LNCS, vol. 6497, pp. 209–224. Springer, Heidelberg (2010)
18. Paulheim, H., Fürnkranz, J.: Unsupervised generation of data mining features from linked open data. In: International Conference on Web Intelligence, Mining, and Semantics (WIMS 2012) (2012)
19. Paulheim, H., Ristoski, P., Mitichkin, E., Bizer, C.: Data mining with background knowledge from the web. In: RapidMiner World (2014)
20. Ristoski, P., Bizer, C., Paulheim, H.: Mining the web of linked data with rapidminer. J. Web Seman. (2015). To appear
21. Ristoski, P., Loza Mencía, E., Paulheim, H.: A hybrid multi-strategy recommender system using linked open data. In: Presutti, V., et al. (eds.) SemWebEval 2014. CCIS, vol. 475, pp. 150–156. Springer, Heidelberg (2014)
22. Ristoski, P., Paulheim, H.: A comparison of propositionalization strategies for creating features from linked open data. In: Linked Data for Knowledge Discovery (2014)
23. Schmachtenberg, M., Bizer, C., Paulheim, H.: Adoption of the linked data best practices in different topical domains. In: Mika, P., et al. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 245–260. Springer, Heidelberg (2014)
24. Schmachtenberg, M., Strufe, T., Paulheim, H.: Enhancing a location-based recommendation system by enrichment with structured data from the web. In: Web Intelligence, Mining and Semantics (2014)
25. Schuhmacher, M., Ponzetto, S.P.: Knowledge-based graph document modeling. In: Proceedings of the 7th ACM International Conference on Web Search and Data Mining, WSDM 2014, pp. 543–552. ACM, New York, NY, USA (2014)
26. Andreas Thalhammer. Dbpedia pagerank dataset. Downloaded from (2014). http://people.aifb.kit.edu/ath/#DBpedia_PageRank
27. Ting, K.M., Witten, I.H.: Issues in stacked generalization. Artif. Intell. Res. **10**(1), 271–289 (1999)
28. White, S., Smyth, P.: Algorithms for estimating relative importance in networks. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 266–275. ACM, New York, NY, USA (2003)