Entity Recommendations Using Hierarchical Knowledge Bases

Siva Kumar Cheekula[1], Pavan Kapanipathi[1], Derek Doran[1], Prateek Jain[2], and Amit Sheth[1]

[1] Kno.e.sis Center, Wright State University, USA
{siva,pavan,derek,amit}@knoesis.org
[2] IgnitionOne Solutions
jainprateek@gmail.com

Abstract. Recent developments in recommendation algorithms have focused on integrating Linked Open Data to augment traditional algorithms with background knowledge. These developments recognize that the integration of Linked Open Data may offer better performance, particularly in cold start cases. In this paper, we explore if and how a specific type of Linked Open Data, namely hierarchical knowledge, may be utilized for recommendation systems. We propose a content-based recommendation approaches that adapts a spreading activation algorithm over the DBpedia category structure to identify entities of interest to the user. Evaluation of the algorithm over the Movielens dataset demonstrates that our method yields more accurate recommendations compared to a previously proposed taxonomy driven approach for recommendations.

Keywords: Content-based recommendation, Entity Relationships, Wikipedia, Semantics, Hierarchy, Knowledge Bases

1 Introduction

Entities are an underlying part of our everyday activities on the web. 50% of search queries are entities [20]. Recently, search engines such as Google and Yahoo [4] include an additional feature of recommending related entities based on the search query. Although the related entity feature has been lately added by search companies, over the years, entity recommendations have been popular in the name of item recommendations such as movies, song albums, e-commerce, and locations. For example, Netflix recommends movies, Pandora recommends songs and Amazon recommends shopping items. Recommendation engines in these cases utilize different attributes of the user such as prior ratings to these entities (movies, songs), users' demographics, and popularity of the entities among other users. Furthermore, existing knowledge about the entities can be utilized as an additional attribute to provide better recommendations.

Knowledge infused recommendation algorithms have gained significant attention due to their competitive performance [5] and ability to overcome cold

start challenge [15, 23]. Although many researchers are pursuing this area of research, utilizing knowledge bases for recommendations is still largely unexplored and holds the potential of improving entity recommendation algorithms [5, 15]. Considering this line of thought, in this work, we intent to explore hierarchical knowledge derived from crowd sourced knowledge bases for recommendations. Our approach is content based and adapts the spreading activation algorithm on the pruned DBpedia [1] category structure to identify personalized entities as recommendations.

Memory in human brain has been argued to be organized as a hierarchical structure [14]. However, this theory has not been enough explored by researchers to understand users on the web. Some who have explored this area have exploited hierarchical structure that are either manually created [25] or automatically created [24] from the descriptions of the items. Creating taxonomies manually is a tedious and time intensive process whereas automatically creating them using the descriptions of items might lack coverage due to their short descriptions. In this context, we believe that DBpedia's category structure extracted from crowd sourced Wikipedia overcomes the above mentioned drawbacks. Especially, since Wikipedia is crowd sourced, the domain coverage is significantly better. We utilize an automatically pruned hierarchy from DBpedia for recommendations.

In this paper, we introduce a novel approach that utilizes hierarchical structure from DBpedia for entity recommendation. The content-based approach transforms the DBpedia category structure into a taxonomy, and utilizes an adaptation of the spreading activation algorithm [21] to assign values to categories in the taxonomy based on the explicit ratings provided by the user. These scores of the categories are disseminated back to the unrated entities. The score gained by an entity correspond to the degree to which it should be recommended to a user. We evaluate using the Movielens dataset and show that our approach performs better than previous work that proposed a taxonomy driven approach for recommendations.

The rest of the paper is organized as follows: In Section 2 we detail the related work. Section 3 describes our approach, followed by evaluation in Section 4. Section 5 concludes with our thoughts on future work.

2 Related Work

Recommendation systems is a popular area of research [5, 15]. The systems focus on recommending music [2, 18], books [8], items in e-commerce shops [25], and movies [9, 17]. These systems may be classified into three different types, namely (1) content-based, where prior ratings of a user and her demographic information may be considered; [9, 13]; (2) collaborative, where the popularity and ratings of entities from similar or affiliated other users are considered [6]; and (3) hybrid, which is a mix of both content-based and collaborative [11]. Content-based approaches focus on the individuality of a user in submitting recommendations, making them more transparent and amenable to overcoming the cold start problem [15, 23].

Although we have explored just the hierarchical knowledge, broadly our work is considered as a recommendation algorithm that utilizes Linked Open Data. Other recommendation techniques that use Linked Open Data have utilized all types of relationships for recommendations [9, 18, 17]. Passant in his work [18], recommends entities based on its "linked data semantic distance (LDSD)" from other explicitly rated entities of the user. The LDSD does not restrict any property in the DBpedia graph and considers the DBpedia as an undirected graph. In [9], Di Noia et al. have harnessed DBpedia in order to recommend movies based on the content of the user. Similar to Passant's work, Di Noia et al. also utilize all the properties of DBpedia and introduce a function that measure the similarity between the entity to be recommended and the entity rated by the users. Ostuni et al. [17] present a hybrid recommendation system, named Sprank, that utilize both neighborhood users ratings and the users' explicit ratings for recommendations. They create a bipartite graph between users and entities (rated and unrated) of length 4 from DBpedia. From the analysis of the paths, path-based features are extracted. Then they apply a learning to rank algorithm to recommend the most relevant items to the user. From these approaches it is hard to analyze the value created by each property on Linked Open Data for recommendations. We, on the other hand, have tried to explore different features of hierarchical relationships on DBpedia to improve recommendation algorithm.

The integration of hierarchical knowledge in recommendation is becoming an emerging area of interest [11, 16, 24, 25]. However, the taxonomies considered are either manually created [25] or are automatically derived from the content descriptions of the items [24]. In our work, we instead use a hierarchical structure in Wikipedia (DBpedia), which is a large crowdsourced taxonomy, for recommendation. Most of the approaches are hybrid and utilize latent factor models in order to accomplish the goal of recommendations. However, although Ziegler et al. [25] utilize a complete different taxonomy (Amazon's book taxonomy) for evaluation, their approach can be perceived as an adaptation of spreading activation algorithm. Their approach can be clearly distinguished into content-based and collaborative parts. In this work, we evaluate our work against the content-based part of Zeiegler's work as the baseline for comparison.

3 Approach

Our algorithm recommends entities given entity ratings in the following steps, as illustrated in Figure 1:

- **Preprocessing DBpedia Category Graph:** In a one time preprocessing step, the DBpedia category graph is transformed into a taxonomic structure with multiple inheritance.
- **Determining Interest Categories:** Explicit ratings of entities, provided by a user as input to the algorithm, is spread to the corresponding categories of the DBpedia taxonomy by adapting a spreading activation algorithm. The categories in the taxonomy are scored reflecting the degree of user's interest.

– **Recommending Entities:** The score of the categories from the previous step is again spread to the unrated entities in the taxonomy. The highest scored entities are recommended to the user.

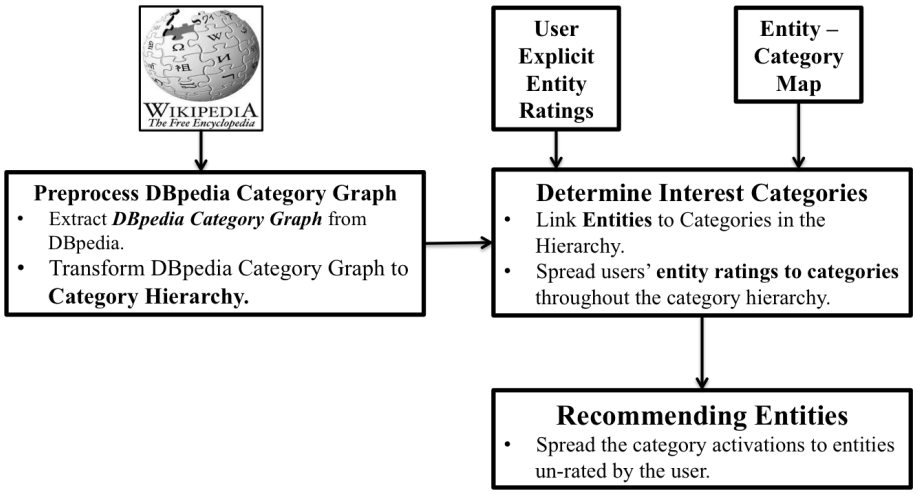

Fig. 1. Approach for recommending entities from DBpedia Category Graph

We next detail the operation of these three steps.

3.1 Preprocessing DBpedia Category Graph

Wikipedia, a collaborative encyclopedia, is a source of not only human readable knowledge but also semi-structured knowledge that is leveraged by a variety of applications. This semi-structured data is the wikilinks added to Wikipedia articles by users. For example, wikilinks on the Albert Einstein [3] Wikipedia page includes an infobox with information such as birth date, profession, death date, and awards. The semi-structured information across Wikipedia has been extracted and transformed into a structured RDF format as DBpedia [1], which is one of the most popular datasets on Linked Open Data[4] [3]. DBpedia therefore can be thought of as a highly structured form of Wikipedia. It has been principally utilized to infuse knowledge for tasks such as semantic similarity [22], and recommendations [2, 17, 18].

A curse of Wikipedia (and hence DBpedia) as a collaborative system is that its category structure is not organized as a formal taxonomy. This is because users are allowed to link any Wikipedia category as a category of an entity, without regard to whether the categorical relationship is reasonable. If such

[3] `http://en.wikipedia.org/wiki/Albert_Einstein`
[4] `http://lod-cloud.net/`

a hierarchy could be extracted from DBpedia, however, it could be invaluable for entity recommendations. For example, if we can determine that a user is interested in *category:films by james cameron*[5] and *category:horror films*[6], it would be reasonable to recommend entities that are children of both categories (namely, horror movies by James Cameron to the user).

We followed our previous approach [12] to infer the hierarchical structure of DBpedia category graph. The process can be summarized as follows:

1. All DBpedia categories corresponding to Wikipedia administrative categories[7] are removed based on a keyword set [19];
2. Select "Category:Main Topic Classification", which subsumes more than 98% of the categories, as the root node of the taxonomy.
3. Determine the abstract level of each category based on its shortest path length to the root node. Our intuition is that, the closer the category is to the root category, the more abstract it is. Hence, we assigned hierarchical levels that reflected its shortest distance to the root category.
4. Remove all the edges in the category graph that limit the DBpedia category graph from being structured as a taxonomy (e.g, remove all directed edges pointing to an entity that is more abstract than its source node).

3.2 Determining Interest Categories

In this work, the Spreading Activation theory is used to assign appropriate scores for the categories in the DBpedia category taxonomy. With an assumption that human memory is represented as semantic network of concepts [21], spreading activation algorithms spread scores of initially activated concepts to its neighboring concepts in the semantic network. The semantic network is formally a network data structures consisting of nodes connected by edges. The spreading of scores is controlled by an application dependent activation function. In order to simulate the human memory, the activation function usually includes a decay factor that decays the scores as and when the activation is performed for nodes farther from the initially activated nodes. Formally, a simple activation function is as follows:

$$A_i = \sum_{j \in C(i)} A_j \times W_{ij} \times D \tag{1}$$

where i is the node to be activated, A_i is the activation value of node i, $C(i)$ is the set of activated child nodes of i, W_{ij} is the weight of the edge connecting i and j, and D is a decay factor.

We first determine the scores for each of the categories in the taxonomy by adapting the spreading activation algorithm. The user's explicit ratings of entities are used as an initial assignment to find the categories of importance.

[5] `http://en.wikipedia.org/wiki/Category:Films_directed_by_James_Cameron`
[6] `http://en.wikipedia.org/wiki/Category:Horror_films`
[7] `http://en.wikipedia.org/wiki/Category:Wikipedia_administration`

Although explicit ratings for entities are generally available, finding the mappings of these entities to a DBpedia category is a challenging task. We instead utilize an existing Movielens dataset [10] that was already mapped to DBpedia by Di Noia et.al. [9]. Entities from the Movielens dataset therefore correspond to the leaf nodes of the taxonomy. Edges associated with an entity may point to categories at different levels in the taxonomy. For example, the film *Die Hard* is associated with categories *English Language Films*, *1988 Films*, *American Action Films*, *Terrorism in Fiction* and *Films directed by John McTiernan* which are at different hierarchical levels depending on their distance from the root node. Since we have the entities that the user has rated linked to the taxonomy, they can serve as source nodes for a spreading activation function. Based on experiments carried out in our prior work [12], we consider the following two spreading activation functions:

- Bell Log - This activation function normalizes the scores based on the number of subcategories in the next hierarchical level. This is introduced to counter the uneven distribution of nodes at the hierarchical levels in the taxonomy. Bell log performs a log normalization of the count. The activation function is shown below.

$$A_i = FL_i \times \sum_{j \in C(i)} A_j \tag{2}$$

  where $FL_i = \frac{1}{log(nodes_{h_i+1})}$, h_i is the hierarchical level of i, and $nodes_l$ is the number of nodes at hierarchical level l.

- Intersect booster - This activation function adds additional score to categories that are the common ancestors of multiple initially activated entities (i.e. the user's explicitly rated entities). This parameter has shown potential in scoring the best categories of interest in our previous work [12]. It is given by:

$$A_i = FL_i \times B_i \times \sum_{j \in C(i)} A_j \tag{3}$$

  where FL_i is the bell log normalization, and $B_i = \frac{ExplicitEntities_i}{Max(ExplicitEntities_{child_i})}$, $ExplicitEntities_i$ is the number of entities explicitly rated by users and are activating category i. The denominator $Max(ExplicitEntities_{child_i}$ is the maximum of the explicitly rated entities that have activated one of the immediate subcategory of category i.

3.3 Recommending Entities

The spreading activation functions discussed above are used to score each categories in the taxonomy based on the degree of users' interest. We subsequently use these scores to make recommendations for a user by *spreading back these scores* to all of the unrated entities living at the bottom of the hierarchy. We

term this operation as a *reverse spreading*. It is important to note that the reverse spreading is just one level deep, i.e. we spread the activation values of categories only to the entities that are just directly below it in the taxonomy. This is because, in Section 3.2, we utilized different activation functions to score the most relevant categories of user interests in the taxonomy. In doing so, we expect the output of the previous step to be normalized interest scores for each category. Hence, if a category at hierarchical level 5 is scored higher than one of its subcategory at hierarchical level 6, then it is assured that our approach terms the category at hierarchical level 5 to be more important to the user.

We initially experiment with a simple reverse spreading activation function where the scores of the parents i.e. categories in the hierarchy of each entity is summed up:

$$A_i = \sum_{j \in N(i)} A_j \tag{4}$$

Where $N(i)$ is the set of category nodes associated with entity i in the taxonomy. A challenge with 4 is that some categories have a larger number of entities, in which case a large number of entities may be scored significantly higher just due to one single category. Hence, we normalize the scores based on the number of entities (outdegree) subsumed by the category. It is important to note that, a category can subsume either an entity or other subcategories, hence two types of edges exist. In this activation function, we only consider the edges that link to an entity from the corresponding category for the outdegree. This updated reverse activation function is thus given as:

$$A_i = \sum_{j \in N(i)} \frac{A_j}{Outdegree(j)_{entity}} \tag{5}$$

where $Outdegree(j)_{entity}$ is the number of entities directly associated with category j.

Another challenge with both of these reverse activation functions is the identical spreading of a node's value to all of its child nodes. Hence, all categories are treated with similar importance to an entity. In contrast, the Wikipedia has a convention that suggests its users in ordering categories to articles [8]. The convention states that the categories of a particular entity should be in order of its significance to the entity. For example, the categories for the Wikipedia article *Cincinatti Reds* are *Major League Baseball teams, Sports in Cincinnati, Ohio,* and *Sports clubs established in 1882*. These categories are ordered as they are listed, implying that *Category:Major League Baseball Team* is the most significant category of *Cincinnati Reds* than the rest. As the intuition behind this suggestion certainly makes the reasonable categorization, it is important to accommodate it in our algorithm. Therefore we introduce a parameter, which is the priority of the category to the entity. The priority is the rank of the category in order, hence we multiply the activation value with its inverse:

[8] `http://en.wikipedia.org/wiki/Wikipedia:Categorization`

$$A_i = \sum_{j \in N(i)} \frac{A_j}{Outdegree(j)_{entity} \times P_{ij}} \tag{6}$$

where P_{ij} is the priority rank of category j to entity i.

4 Evaluation

This section describes the evaluation of our recommendation approach against a baseline method. As our approach aims at determining the most relevant entities of interest to a user, we measure the quality of the approach as the capability of listing the most interesting entities in the top ranks of the predicted entities. In the following sections, we first present the details of the dataset, then the evaluation approach and finally discuss the results.

4.1 Dataset

We considered the data set by Movielens [10] which is widely used for recommendation systems evaluation. The Movielens data set consists of 1,000,209 ratings for 3,883 movies by 6,040 users. As our algorithm is based on DBpedia, it is necessary for each movie in the Movielens dataset to have a corresponding entity/resource in DBpedia. Hence, we utilize the mappings open sourced by Di Noia et al. [9] that maps movies in the dataset to an appropriate DBpedia resource. The mappings contain 3,148 movies out of 3,883 total movies in the dataset. Since we are a content-based approach, sparse data (users who have rated very few movies), significantly impacts the recommendation performance [5]. Hence, similar to [9, 17] we eliminated users who has ratings for less than 20 movies. After these filtering steps, our data set contains 3,148 movies rated by 5,886 users with approximately 0.9M ratings.

4.2 Baseline approach

A similar adaptation of spreading activation theory on taxonomies has been presented by Zeigler et al. [25]. This work presents a hybrid approach that applies spreading activation theory on a hand crafted taxonomy. Similar to our approach it also utilizes a normalized activation function[9] to score the categories of interest for each user. The output of this step is a vector of categories for each user v_u. Next, they perform the similar spreading for each item to be recommended resulting in a vector of categories v_i. Content-based recommendation for a user u finds the pearson correlation between the vector of each item i, v_i and v_u (vector representing the user). This is termed as the product proximity. In order to make a fair comparison of content based recommendations, we ignore the user proximity of Ziegler et al. [25], which is the collaborative part of the algorithm.

[9] More details of the activation function used is in [25].

4.3 Evaluation Approach

Recommendation system which predicts the user ratings for an entity are evaluated using the error metrics(such as RMSE, MAE). However, Cremonesi et al. [7] has argued that the recommender system which focuses on retrieving the most relevant entities should be evaluated by means of accuracy metrics and not error metrics. Hence, in this work, we implemented the evaluation approach proposed by Cremonesi et al. to assess the performance of personalized, content-based, movie recommendations of our approach against the baseline explained in Section 4.2. The evaluation approach is as follows:

After applying the filtering discussed in Section 4.1 on the original Movielens data set, we randomly selected 1.4% of the entire data set and used it as test set $T(|T| = 9805)$. The remaining 98.6% items are used as training set M. For each of the test item i with 5-stars in T, we performed the following.

- We randomly select the 1,000 items unrated by the user with an assumption that those were uninteresting to her.
- A ranked list of 1,001 items (i.e. 1,000 unrated plus the test item i) is formed by our algorithm.
- Then, we evaluate by picking the top N items. If the test item i is in the top N list, we have a hit, or a miss otherwise.

The probability of hit increases as N increases. We consider recall as the ratio of total hits and the size of the 5* test items from T. The set of 5* test items in test set T are denoted by K. Formally, recall is as follows:

$$recall(N) = \frac{\#Hits}{|K|} \tag{7}$$

4.4 Results

	Summing	**DegreeNorm**	**PriorityDegree**
BellLog	0.15	0.18	0.21
BellLogIntersect	0.09	0.20	**0.24**
BaseLine	0.002		

Table 1. Recall at Top 20 recommendation evaluation of various spreading and reverse spreading functions.

In Table 1, we report the recall at top 20 achieved on the Movielens dataset. Each cell represents the recall of the spreading (row) and reverse spreading functions (column) we described in 3. The final row of the Table 1 shows the results using the baseline approach. It is evident that all the variations of our approach beats the baseline. Plausible reasons for the poor performance of the baseline, based on our analysis of the results are: (1) it scores abstract categories

higher due to the structure of DBpedia taxonomy. There wasn't significant decay of spreading and the normalization is insufficient; and (2) the categories in the higher level of the taxonomy were common for all the users and entities, hence a pre-existing bias existed during the calculation of pearson correlation.

It is not surprising that Bell Log Intersect is the best performing spreading activation function since the parameters used in the activation function also had significant impact in our previous work [12]. Table 1 proves our intuition of adding new parameters to the reverse spreading activation function. *Summing* is the most simplest reverse spreading activation function used, and carries two major drawbacks. It: (1) ignores the number of entities associated with the category, which is important since some categories are generic and subsume many entities; and (2) spreads the same activation value to all the entities in the category. However, it is surprising to see that Summing with Bell Log performs better than Summing with Bell Log Intersect. Analyzing this unexpected performance is a part of our future work.

Degree Normalization function normalizes the spread of values from the category as a function of its out-degree. This hinders the importance provided to generic categories from the spreading activation functions. This shows that although, these activation functions achieved significant results in generating hierarchical interests from tweets, there are further areas of improvement to adapt it to recommendations. As we introduce the significance factor among relationships in taxonomy using priority parameter in reverse spreading, we observed the improvements in recommendations. This variation of the reverse spreading activation function performs the best among others we have experimented. The priority that users collaboratively add on Wikipedia for categories of each article does provide enough information to make an impact on recommendations of entities.

5 Conclusion and Future work

In this paper we have presented a content-based recommendation algorithm, building upon our previous work, that utilizes the hierarchical structure of a crowd sourced knowledge base. Utilizing spreading activation on a taxonomy, we experimented with different activation functions that leverages various features of the hierarchical structure. In our evaluation we have shown that we perform significantly better than other approaches that harness item taxonomies derived from crowd-sourced knowledge bases. In future, we intend to explore the category graph of DBpedia better for recommendations. Removal of noisy categories can make a significant impact in recommendations, we intend to develop techniques that can help us ignore categories that are not helpful for recommendations. As we have an initial result about the effect of the hierarchical level of categories (abstractness), we plan to optimize the activation function based on the hierarchical levels of the categories. Finally, by experimenting with more features of the hierarchy, we will develop a hybrid recommendation algorithm.

References

1. Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. *Dbpedia: A nucleus for a web of open data*. Springer, 2007.
2. Stephan Baumann and Rafael Schirru. Using linked open data for novel artist recommendations. In *13th Internal Society for Music Information Retrival Conference, Porto*, 2012.
3. Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far, 2009.
4. Roi Blanco, BerkantBarla Cambazoglu, Peter Mika, and Nicolas Torzec. Entity recommendations in web search. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, JosianeXavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, and Krzysztof Janowicz, editors, *The Semantic Web ISWC 2013*, volume 8219 of *Lecture Notes in Computer Science*, pages 33–48. Springer Berlin Heidelberg, 2013.
5. J. Bobadilla, F. Ortega, A. Hernando, and A. GutiéRrez. Recommender systems survey. *Know.-Based Syst.*, 46:109–132, July 2013.
6. Laurent Candillier, Frank Meyer, and Marc Boull. Comparing state-of-the-art collaborative filtering systems. In Petra Perner, editor, *Machine Learning and Data Mining in Pattern Recognition*, volume 4571 of *Lecture Notes in Computer Science*, pages 548–562. Springer Berlin Heidelberg, 2007.
7. Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM.
8. Rubn Gonzlez Crespo, Oscar Sanjun Martnez, Juan Manuel Cueva Lovelle, B. Cristina Pelayo Garca-Bustelo, Jos Emilio Labra Gayo, and Patricia Ordoez de Pablos. Recommendation system based on user interaction data applied to intelligent electronic books. *Computers in Human Behavior*, 27(4):1445 – 1449, 2011. Social and Humanistic Computing for the Knowledge Society.
9. Tommaso Di Noia, Roberto Mirizzi, Vito Claudio Ostuni, Davide Romito, and Markus Zanker. Linked open data to support content-based recommender systems. In *Proceedings of the 8th International Conference on Semantic Systems*, I-SEMANTICS '12, pages 1–8, New York, NY, USA, 2012. ACM.
10. Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, CSCW '00, pages 241–250, New York, NY, USA, 2000. ACM.
11. Bhargav Kanagal, Amr Ahmed, Sandeep Pandey, Vanja Josifovski, Jeff Yuan, and Lluis Garcia-Pueyo. Supercharging recommender systems using taxonomies for learning user purchase behavior. *Proc. VLDB Endow.*, 5(10):956–967, June 2012.
12. Pavan Kapanipathi, Prateek Jain, Chitra Venkataramani, and Amit Sheth. User interests identification on twitter using a hierarchical knowledge base. In Valentina Presutti, Claudia dAmato, Fabien Gandon, Mathieu dAquin, Steffen Staab, and Anna Tordai, editors, *The Semantic Web: Trends and Challenges*, volume 8465 of *Lecture Notes in Computer Science*, pages 99–113. Springer International Publishing, 2014.
13. Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th international conference on machine learning*, pages 331–339, 1995.

14. Elizabeth F. Loftus and Ronald W. Scheff. Categorization norms for fifty representative instances. *Journal of Experimental Psychology*, 91(2):355, 1971.
15. Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer US, 2011.
16. Aditya Krishna Menon, Krishna-Prasad Chitrapura, Sachin Garg, Deepak Agarwal, and Nagaraj Kota. Response prediction using collaborative filtering with hierarchies and side-information. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 141–149, New York, NY, USA, 2011. ACM.
17. Vito Claudio Ostuni, Tommaso Di Noia, Eugenio Di Sciascio, and Roberto Mirizzi. Top-n recommendations from implicit feedback leveraging linked open data. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 85–92. ACM, 2013.
18. Alexandre Passant. dbrec–music recommendations using dbpedia. In *The Semantic Web–ISWC 2010*, pages 209–224. Springer, 2010.
19. Simone Paolo Ponzetto and Michael Strube. Deriving a large scale taxonomy from wikipedia. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2*, AAAI'07, pages 1440–1445. AAAI Press, 2007.
20. Jeffrey Pound, Peter Mika, and Hugo Zaragoza. Ad-hoc object retrieval in the web of data. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 771–780, New York, NY, USA, 2010. ACM.
21. M Ross Quillan. Semantic memory. Technical report, DTIC Document, 1966.
22. Marcin Szczuka, Andrzej Janusz, and Kamil Herba. Clustering of rough set related documents with use of knowledge from dbpedia. In JingTao Yao, Sheela Ramanna, Guoyin Wang, and Zbigniew Suraj, editors, *Rough Sets and Knowledge Technology*, volume 6954 of *Lecture Notes in Computer Science*, pages 394–403. Springer Berlin Heidelberg, 2011.
23. Li-Tung Weng, Yue Xu, Yuefeng Li, and Richi Nayak. Exploiting item taxonomy for solving cold-start problem in recommendation making. In *Tools with Artificial Intelligence, 2008. ICTAI'08. 20th IEEE International Conference on*, volume 2, pages 113–120. IEEE, 2008.
24. Yuchen Zhang, Amr Ahmed, Vanja Josifovski, and Alexander Smola. Taxonomy discovery for personalized recommendation. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 243–252, New York, NY, USA, 2014. ACM.
25. Cai-Nicolas Ziegler, Georg Lausen, and Lars Schmidt-Thieme. Taxonomy-driven computation of product recommendations. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pages 406–415, New York, NY, USA, 2004. ACM.