An evaluation of SimRank and Personalized PageRank to build a recommender system for the Web of Data

Phuong T. Nguyen, Paolo Tomeo, Tommaso Di Noia, Eugenio Di Sciascio
SisInf Lab, Polytechnic University of Bari
Via Orabona 4 – 70125 Bari, Italy
{phuong.nguyen, paolo.tomeo, tommaso.dinoia, eugenio.disciascio}@poliba.it

ABSTRACT

The `Web of Data` is the natural evolution of the World Wide Web from a set of interlinked documents to a set of interlinked entities. It is a graph of information resources interconnected by semantic relations, thereby yielding the name `Linked Data`. The proliferation of `Linked Data` is for sure an opportunity to create a new family of data-intensive applications such as recommender systems. In particular, since content-based recommender systems base on the notion of similarity between items, the selection of the right graph-based similarity metric is of paramount importance to build an effective recommendation engine. In this paper, we review two existing metrics, SimRank and PageRank, and investigate their suitability and performance for computing similarity between resources in `RDF` graphs and investigate their usage to feed a content-based recommender system. Finally, we conduct experimental evaluations on a dataset for musical artists and bands recommendations thus comparing our results with two other content-based baselines measuring their performance with precision and recall, catalog coverage, items distribution and novelty metrics.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval - *Information Filtering*

Keywords

SimRank; Personalized PageRank; Recommender Systems; Web of Data

1. INTRODUCTION

In the last year we are witnessing the evolution of the Web we used to know towards a huge distributed knowledge base. The *World Wide Web* is moving fast from a network of documents to a network of interconnected data (entities) thus creating the so called `Web of Data`. This latter has been introduced as a new scheme for bringing structured data into the Web. Data in the `Web of Data` is represented as a graph of semantically connected resources by means of `RDF`[1] thus allowing the structured data provided by `Linked Data` to be not only graspable by human beings but also processable by computers. This paves the way for automatic processing of Web contents, thus helping to mine existing data and deduce new knowledge. To date, information sharing, information retrieval [7, 8], community detection, recommendation systems [5, 18, 19] - to name a few - are the noteworthy applications that successfully leverage `Linked Data`. Nonetheless, to fully take advantage of `Linked Data`, the need for algorithms being capable of effectively handling `RDF` graphs is immense. One of the main tasks for `Linked Data` applications is to find similar entities with a given resource or to evaluate to which extent two resources are alike. This leads to the problem of computing similarity between semantically related resources. Semantic similarity measurement, therefore, is considered as a building block for `Linked Data` applications.

In this paper, we investigate the effectiveness of SimRank and Personalized PageRank as a means for measuring semantic similarity in `RDF` graph to build a content-based recommendation engine. SimRank has been employed effectively for measuring similarity in homogeneous graphs [27] and similarly, PageRank has also been successfully exploited in calculating similarity for WordNet [2, 3]. We investigate how effectively the two metrics work with `Linked Data`. We perform experiments on data retrieved from `DBpedia`[2], the cornerstone of the whole Linked Open Data cloud[3], to feed a content-based recommender system (RS) with the aim of evaluating the effectiveness of the two metrics. Indeed, the knowledge encoded in semantic datasets of the Linked Open Data project can be exploited to improve the performance of content-based (CB) approaches to recommendation [5]. Such systems try to recommend items similar to those a given user has liked in the past, matching up the attributes of a user profile in which preferences and interests are stored, with the attributes of the items descriptive content [15]. We use the values computed with SimRank and Personalized PageRank, to find similarities between items and use them to produce the final recommendation list. Our experimental evaluation has been conducted by using the well

[1]`http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140225/`
[2]`http://dbpedia.org`
[3]`http://lod-cloud.net/`

known dataset of Last.fm[4] for musical artists recommendation. We compare our recommendation results with two other content-based baselines measuring their accuracy with precision and recall metrics as well as measuring catalog coverage, items distribution and novelty.

The main contributions of the paper are: (i) the evaluation of the effectiveness in the adoption of two well-established graph-based ranking metrics in a *pure content-based RS* scenario; (ii) the analysis of their performances in terms not only of precision and recall but also of diversity, items coverage and novelty.

The paper is structured in the following sections. Section 2 introduces the metrics SimRank and PageRank while Section 3 presents the implementation and experimental results. Section 4 brings an overview of related work. Finally, Section 5 concludes the paper.

2. SIMRANK AND PAGERANK FOR MEASURING GRAPH SIMILARITY

An `RDF` graph is defined as a directed graph `G=(V,E,R)`, where `V` is the set of vertices, `E` is the set of edges and `R` represents the relationship among the nodes. An `RDF` graph consists of enormous nodes and oriented links with semantic relationships. Its building block is the triple <`subject, predicate, object`> stating that the node `subject` is connected to the node `object` by means of the edge labelled with `predicate`. To evaluate how similar two given resources within an `RDF` graph are, it is necessary to incorporate their intrinsic characteristics into the similarity calculation. To be precise, nodes, links, and the mutual relationships among subjects and objects could be considered as input for the calculation. Although originally developed for homogeneous graphs, Personalized PageRank and SimRank can be two interesting candidates to compute similarity between `RDF` resources. For the sake of completeness, in the following we briefly recall the way they are computed.

SimRank.

SimRank has been proposed to compute similarity between nodes in a graph using the structural context [12]. Similarity is calculated according to object-to-object relationships: the similarity between two nodes is dependent on their neighbors. Two nodes are considered to be similar if they are referenced by similar nodes. The similarity value for two nodes α and β is computed by SimRank using a fixed-point function. Given $k \geq 0$ we have $R^{(k)}(\alpha,\beta) = 1$ with $\alpha = \beta$. Dually, we have $R^{(k)}(\alpha,\beta) = 0$ with $k = 0$ and $\alpha \neq \beta$. In all the other cases the general formula is

$$R^{(k+1)}(\alpha,\beta) = \frac{d}{|I(\alpha)| \cdot |I(\beta)|} \sum_{i=1}^{|I(\alpha)|} \sum_{j=1}^{|I(\beta)|} R^{(k)}(I_i(\alpha), I_j(\beta)) \tag{1}$$

where d is a damping factor $(0 \leq d < 1)$; $I(\alpha)$ and $I(\beta)$ are the set of inbound neighbors of α and β, respectively. $|I(\alpha)| \cdot |I(\beta)|$ represents a normalization factor to have $R^{(k)}(\alpha,\beta) \in [0,1]$. Equation (1) implies that the similarity for two nodes is computed by aggregating the similarity of all possible pairs of their neighbors. SimRank has been originally designed for homogeneous graphs.

[4]`http://sisinflab.poliba.it/semanticweb/lod/recsys/datasets/`

Personalized PageRank.

By PageRank, the rank of a node is calculated according to the relationship with other nodes. A node receives an amount of rank from every node which points to it and in turn transfers an amount of its rank to the nodes it refers to. In this sense, a node will have a high rank if it is referenced by nodes with high rank. To compute the rank of n nodes, an $n \times n$ transition matrix G is built from the link relationship between the nodes. In this matrix, row i represents the rank that node α_i transfers to other nodes that it has links to. Given the set $O(\alpha)$ representing the set of outbound links of α, it transfers the amount of rank $rank(\alpha) = \frac{1}{|O(\alpha)|}$ to all of its neighborhood nodes. In the transition matrix, the cell at row i and column j has the value of $rank(\alpha)$ if there is a link from node α_i to node α_j, otherwise it has the value of 0. From this definition, a problem arises with *dangling nodes*, i.e. nodes with no outgoing links. By these nodes, the PageRank vectors degrade very quickly and produce inappropriate ranks. To circumvent dangling nodes, some amendment is made to the transition matrix. This is done by introducing two vectors thus redefining the transition matrix as $G' = G + \vec{\delta} \cdot \vec{\omega}$, where $\|\vec{\omega}\|_1 = 1$ and δ is a column vector with $\delta_i = 1$ if i is a dangling node and $\delta_i = 0$ otherwise. Usually, all entries ω_i are set to $\frac{1}{n}$. Based on the original transition matrix, the Google matrix used for PageRank is defined as follows [26] as $G'' = d \cdot G' + (1-d) \cdot \vec{v}$ where d is the damping factor $(0 \leq d < 1)$ and $\vec{v}$ is the personalization vector. The outcome vector represents the ranks of all nodes, i.e. entry i holds the rank of the graph node α_i. Similar to SimRank, the PageRank vector is obtained after a finite number of iterations. The complete formulation is:

$$\vec{\pi}^{(k+1)} = d \cdot \vec{\pi}^{(k)} \cdot G + d \cdot (\vec{\pi}^{(k)} \cdot \vec{\delta}) \cdot \vec{\omega} + (1-d) \cdot \vec{v} \tag{2}$$

A variant of the original PageRank algorithm, the Personalized PageRank algorithm was derived to measure the similarity between topics [11]. The main idea of this approach is to exploit PageRank vector to characterize a topic which is comprised of a set of words. The topic is characterized by concentrating probability mass to its constituent words, represented as nodes in a graph. This is done by modifying the personalization vector $\vec{v}$ in Equation (2). The corresponding entries in $\vec{v}$ are assigned the value of 1, whilst all the other entries of $\vec{v}$ are assigned the value of 0. By doing this the biased topic will earn a high rank. The PageRank vector obtained is considered as the features of the topic and helps distinguish the topic from others. The similarity between two topics α and β represented by vectors $\alpha = \{a_i\}_{i=1,..,n}$ and $\beta = \{b_i\}_{i=1,..,n}$ is computed as the inner product space between the two vectors [24].

$$p.PageRank(\alpha,\beta) = \frac{\sum_{i=1}^{n} a_i \times b_i}{\sqrt{\sum_{i=1}^{n}(a_i)^2} \times \sqrt{\sum_{i=1}^{n}(b_i)^2}}$$

Personalized PageRank has been applied to measure similarity between words in WordNet [2, 3]. Following the same line of reasoning, we believe that Personalized PageRank can be used for computing similarity between resources in an `RDF` graph. In the preceding sections, we are going to employ SimRank and Personalized PageRank to measure similarity between resources in `Linked Data`.

3. EVALUATION

A question that might arise at any time is how effective the two metrics are with regards to a recommendation task? In this section we present our attempt to analyze the performance of SimRank and PageRank. Such a study is of particular importance since it not only provides an insight into the suitability but also casts light on the effectiveness of measurement techniques for `Linked Data`. In our experiments, we re-implemented the two metrics, SimRank and PageRank and we conducted experiments on `RDF` graphs. In particular, as input for the calculation, we retrieved data from `DBpedia` via `SPARQL` queries and extracted a subgraph containing only the information related to a specific domain. In particular, we considered data from the music domain. We retrieved resources that are instances of the two classes `dbo:MusicalArtist` and `dbo:Band`. For each resource we also got the `RDF` triples that are involved in as subject or object. The final outcome is then used in a content-based recommendation engine to evaluate its performance in terms of accuracy, novelty, items distribution and sales diversity. The latter is considered a relevant quality dimension for both business and user perspective: the user may receive less obvious recommendations, comply with the objective to help users discover new content [25] and the business may increase the sales [6].

From a computational point of view, evaluating both SimRank and Personalized PageRank value is a high demanding task that strongly depends on the number of edges connecting nodes within the graph. In order to reduce the computational load we downsized the extracted subgraph by selecting a set of `RDF` properties that result meaningful for the domain of interest (music in our case). We selected those properties belonging to the `DBpedia` ontology (plus `dcterms:subject`) that occur most frequently in the dataset in relation to musical artists and bands. In particular we selected incoming and outgoing properties as shown in Table 1.

Inbound	*Outbound*
`dbo:producer`	`dcterms:subject`
`dbo:artist`	`dbo:genre`
`dbo:writer`	`dbo:associatedBand`
`dbo:associatedBand`	`dbo:associatedMusicalArtist`
`dbo:associatedMusicalArtist`	`dbo:instrument`
`dbo:musicalArtist`	`dbo:occupation`
`dbo:musicComposer`	`dbo:birthPlace`
`dbo:bandMember`	`dbo:background`
`dbo:formerBandMember`	
`dbo:starring`	
`dbo:composer`	

Table 1: The set of properties used to compute similarity values between pairs of resources.

3.1 The Recommendation Engine

In order to show the quality of similarities computed with SimRank and PageRank for a content-based recommender system, we have carried out experiments to evaluate the proposed recommendations. We selected the well known dataset `Last.fm hetrec-2011`[5].

Last.fm contains 92835 implicit feedbacks on 17632 artists by 1892 users. For computational reasons we downsized the number of artists and bands to the 700 most popular ones. Then a mapping of each item to the corresponding DBpedia URI has been used by exploiting the data available at `http://sisinflab.poliba.it/semanticweb/lod/recsys/datasets/`. For the music domain we then extracted the corresponding subgraph composed by a total of 113,386 triples. In Table 2 we summarize some statistics of the data we used. We used for our experiment the k-nearest neighbors algorithm, which finds the set $neighbors(\alpha)$ containing the k most similar entities β to a given item α using a similarity function $sim(\alpha, \beta)$ [15]. The recommendations are computed using a weighted sum as in [21], where the weights are the similarities between the items, to produce the prediction score P for a given user-item pair (u, α). The formula takes into account the items belonging to the user profile $profile(u)$, in our case the artists/bands already listened to, and the score $r(u, \alpha)$ assigned to the item α by the user u.

$$P(u,\alpha) = \frac{\sum_{\beta \in neighbors(\alpha) \cap profile(u)} sim(\alpha,\beta) \cdot r(u,\beta)}{\sum_{\beta \in neighbors(\alpha) \cap profile(u)} sim(\alpha,\beta)}$$

The experiments have been carried out by considering as $sim(\alpha, \beta)$ the values computed via Equation (1) for SimRank and Equation (3) for Personalized PageRank. In order to compare our results, we also selected for our experiments two more similarity functions as baselines: a pure vector space model with cosine similarity (VSM) and a simplified variation of the algorithm proposed in [5] (isemantics).

	Last.fm
Number of users	1,867
Number of items	700
Number of ratings	47,330
Data sparsity	0.963%
Avg number of users per item	98.19
Avg number of items per user	25.52
Number of extracted triples	113,386
Avg number of Inbound links	109.25
Avg number of Outbound links	31.44

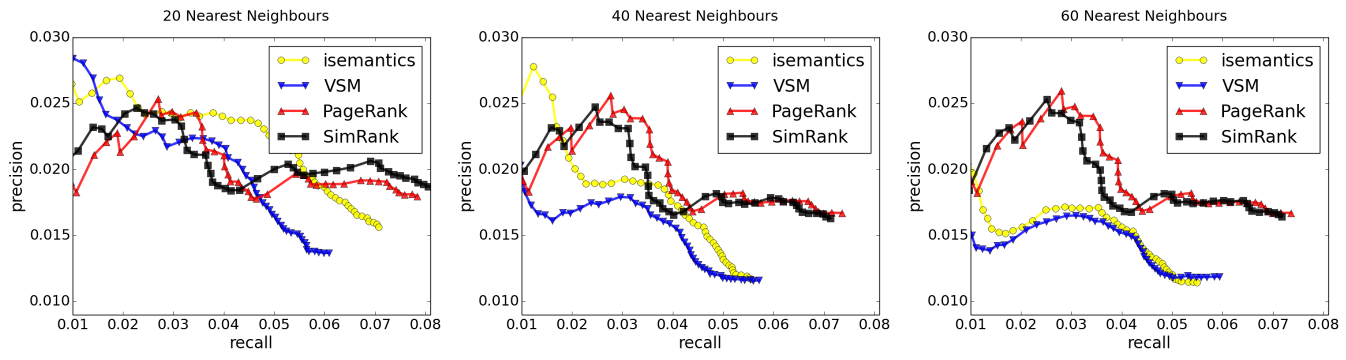
Table 2: Statistics about the Last.fm dataset.

3.2 Experimental Results and Discussion

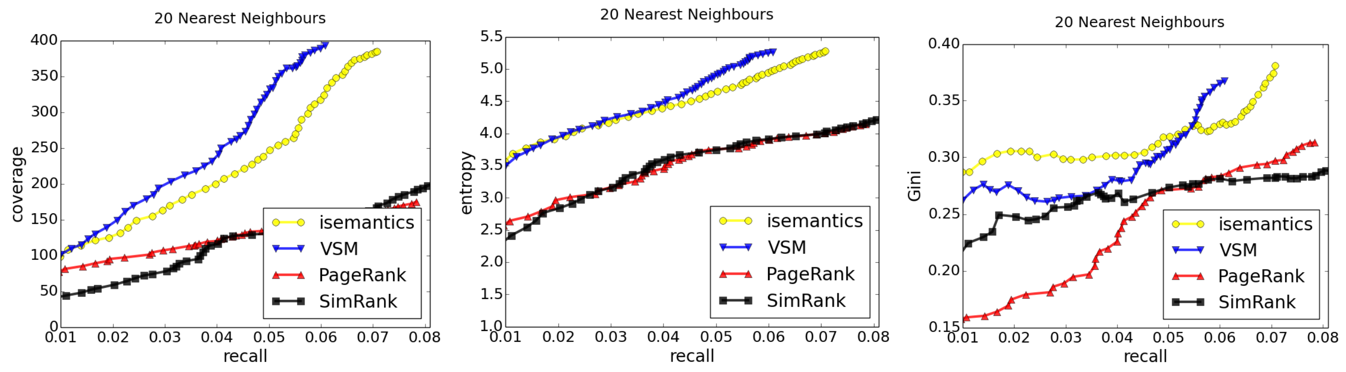
In order to estimate the quality of the recommendations, we used the holdout method splitting a dataset into two parts: training set and test set. We built the training set by using, for each user, the first 60% of the ratings and the remaining 40% to build the test set. Last.fm has implicit feedback, therefore each item in the test set was perceived as relevant. For evaluating recommendation ranking accuracy we used the *TestItems* evaluation methodology presented in [4]. Considering only the top N results, for measuring accuracy we used precision (P@N) and recall (R@N). The former is defined as the fraction of the top-N recommended items that are relevant to the user u; the latter as the fraction of relevant items from the test set that appear in the N predicted items.

To measure the sales diversity, we considered three other important metrics: catalog coverage [9] (the percentage of items in the catalog that are ever recommended to users), Entropy and Gini coefficient to measure the distribution of recommended items [1, 6, 25] (the degree to which recommendations are concentrated on a few items or are more

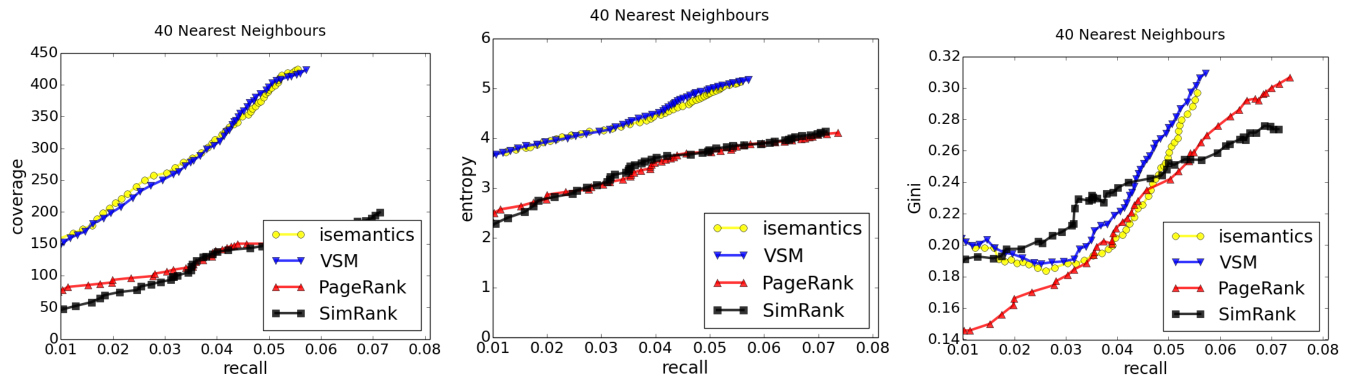
[5]Available at `http://ir.ii.uam.es/hetrec2011/datasets.html`



(a)



(b)



(c)

Figure 1: (a) Precision and Recall curves obtained by varying the length of the recommendations list from 1 to 50 on the Last.fm dataset. Catalog coverage, Entropy and Gini index curves obtained by varying the length of the recommendations list from 1 to 50 on Last.fm, with 20 (b) and 40 (c) neighbors. The results with 60 neighbors are very similar to those with 40 neighbors, therefore they are omitted due to space limitations.

equally distributed across the items). As in [1], we reversed the scale for Gini coefficient so that smaller values represent lower distributional equity and larger values correspond to higher equity.
Furthermore, we used two metrics to measure the novelty of the recommendations: long-tail percentage [1] and Expected Popularity Complement (EPC@N) [25]. The former measures the percentage of long-tail items among the recommendations across all users. The Expected Popularity Complement metric [25] measures the novelty of a recommendation list as the average novelty of the recommended items L_u. More formally, $EPC(L_u) = \frac{\sum_{\alpha \in L_u} nov(\alpha)}{|L_u|}$ where $nov(\alpha)$ measures the probability of not to being known by a random user $nov(\alpha) = 1 - \frac{\sum_{u \in U} 1_{r(u,\alpha)>0}}{\sum_{u \in U} |profile(u)|}$, being U the set of all the users.

The results for the four algorithms were computed with 10, 20, 40, 60, 80 neighbors for different number of top-N recommendations (from top-*1* to top-*50*). Due to space limitations, only the results with 20, 40, 60 neighbors are shown in Figure 1[6]. Figure 1(a) shows the results on Last.fm in terms of precision and recall. SimRank and Personalized

[6]Results with 10 and 80 neighbors are very similar to the ones with 20 and 60 neighbors respectively.

	EPC@10	Long-tail%@10	EPC@20	Long-tail%@20	EPC@30	Long-tail%@30	EPC@40	Long-tail%@40	EPC@50	Long-tail%@50
isemantics	0.869	0.695	0.844	0.648	0.841	0.657	0.839	0.658	0.836	0.643
VSM	0.865	0.677	0.848	0.657	0.847	0.675	0.848	0.673	0.843	0.658
PageRank	**0.913**	**0.780**	**0.913**	**0.823**	**0.893**	**0.767**	0.867	**0.743**	0.860	0.710
SimRank	0.911	0.796	0.906	0.793	0.881	0.747	**0.874**	0.742	**0.867**	**0.717**

Table 3: Comparison of novelty results in terms of EPC@N and Long-tail%@N for the four algorithms with 60 neighbors.

PageRank can produce comparable results and they outperform the two baselines when the number of selected neighbors increases. This means that they are not affected by possible noise if more neighbors are considered. Moreover, we see that starting from 40 neighbors, their results tend to stabilize. They obtain higher recall values in each configuration, thus strongly surpassing the baselines with the higher top-N (particularly over top-30). This means that by increasing the number of recommendations, the two metrics are able to suggest other relevant items, in this sense they perform better than the two baselines. We also observe that they obtain lower values for catalog coverage and dispersion compared to the two baseline. However in Table 3 we see that they outperform the baseline in terms of EPC and long-tail percentage. This means that they tend to suggest always a small subset of items and the suggestions are not equally distributed, but these items do not necessarily belong to the most popular ones.

4. RELATED WORK

In this section we briefly recall some related work adopting a content-based (or hybrid) approach to recommendation by exploiting a semantic-based approach. A system for recommending musical artist and bands based on `DBpedia` is presented in [19]. The systems extracts data about bands and artists from `DBpedia`. Instances, namely resources instantiating the classes `dbo:MusicalArtist` and `dbo:Band`[7], are analyzed to supply input data for the recommendation algorithm. A similarity measurement algorithm, called LDSD, is designed to calculate similarity between a piece of music or an artist and the elements of a candidate set. This semantic similarity metric is computed on the basis of links between resources. Based on the filtering results, the system produces a list of songs or artists that can then be presented to users. In [23], a framework for evaluating artist similarity is built. Given an artist, the system searches for alike artists according to two factors: style and mood. First, the information for style and mood is collected from the All Music Guide website. Afterwards, a co-clustering algorithm is employed to build hierarchical taxonomies describing the two factors. With respect to the taxonomies, the similarity between each pair of terms is computed by means of the existing similarity metrics: Resnik [20], Jiang-Conrath [13], Lin [14] and Schlicker [22]. From the similarity calculation, a list of similar artists for a selected artist is eventually generated. A key function for content-based recommender systems is calculating similarity between items, so that possible items can be recommended to a user based on his past selected preferences. The authors in [17] exploit `Linked Data` to compute similarity between resources for a content-based recommender system. They propose a neighborhood-based graph kernel to compute semantic similarity. In this approach, items are represented as nodes in a neighborhood graph. Starting from two resources, a graph is extended by using a set of selected properties. Each node involved in the similarity calculation is assigned a weight corresponding to its relationship with other neighborhood nodes. Afterwards, a kernel function is devised to calculate similarity between them. The effectiveness of the approach has been proven by the experimental results on the Movielens dataset. A content-based system leveraging `DBpedia` for recommending movies is introduced in [5]. Based on the set of movies that have been preferred by a user, the system engine extracts movie information from `DBpedia` and computes the similarity between a new item and current rated items. A movie is characterized by a set of features corresponding to the neighbor movies in the graph via a property. The set of features is represented as a vector. The cosine function is used to measure semantic similarity. By calculating similarity between two movies, the system can provide a user with a list of similar movies according to his previously selected preferences. For computing *top-N* item recommendations from implicit feedback, a hybrid algorithm - named SPrank - is proposed in [16]. The algorithm extracts path-based features from `DBpedia` and mines semantic graph to detect subtle relationships among items. It incorporates ontological knowledge with collaborative user preferences to produce recommendations. Experimental results on two datasets from MovieLens and Last.fm demonstrate that the proposed algorithm gains a high prediction accuracy even when the experimental data is sparse. In [10], a hybrid recommender system has been proposed to overcome the problems of cold-start and lower accuracy in collaborative and content-based recommender systems. The proposed approach models item interactions using unified Boltzmann machines. By integrating collaborative and content information, the system learns weights representing the importance of different pairwise interactions. Based on the probabilistic models, the system can predict whether a user will act on a specific item. By doing that, more appropriate recommendations can be made.

5. CONCLUSION AND FUTURE WORK

In this paper we have presented an initial investigation on the usage of SimRank and Personalized PageRank for measuring similarity based on the structural context of a graph in order to feed a content-based recommender system. In particular, we analyzed the potential of using these metrics for automatically measuring similarity when the data describing the content are available as `Linked Data`. To validate the outcomes, we conduct experimental evaluations on the Last.fm dataset thus comparing our recommendation results with some other standard content-based baseline measuring their accuracy with precision and recall metrics. Moreover we measured the performance of the recommender system in terms of catalog coverage, items distribution and novelty of results. Experimental results show that, in the

[7]In the rest of the paper, for the sake of clarity and compactness, we will use CURIEs instead of URIs. Moreover, the prefixes for CURIEs are the ones available on `http://prefix.cc`.

given circumstances, SimRank and Personalized PageRank can produce interesting results compared to the two baselines in terms of precision, recall an novelty even though their performance decrease when we evaluate catalog coverage, items distribution.

We are currently in the process of performing the same experiments also on two more domains via the Movielens (movies) and TheLibraryThing (books) datasets. The aim is also to evaluate how much the number of triples available in the Linked Data cloud, and related to a specific domain, may affect the performance of a recommendation algorithm. We are also implementing enhanced versions of SimRank and Personalized PageRank to take into account paths of length greater than two to compute the similarity between resources.

Acknowledgements. The authors acknowledge partial support of PON04a2_E RES NOVAE and PON01_00850 ASK-Health

6. REFERENCES

[1] G. Adomavicius and Y. Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Trans. on Knowl. and Data Eng.*, 24(5):896–911, 2012.

[2] E. Agirre, M. Cuadros, G. Rigau, and A. Soroa. Exploring knowledge bases for similarity. In *Proceedings of LREC'10*, 2010.

[3] E. Agirre and A. Soroa. Personalizing pagerank for word sense disambiguation. In *Proceedings of EACL '09*, pages 33–41, 2009.

[4] A. Bellogin, P. Castells, and I. Cantador. Precision-oriented evaluation of recommender systems: An algorithmic comparison. In *ACM RecSys '11*, pages 333–336, 2011.

[5] T. Di Noia, R. Mirizzi, V. C. Ostuni, D. Romito, and M. Zanker. Linked open data to support content-based recommender systems. In *ACM I-SEMANTICS '12*, pages 1–8, 2012. ACM.

[6] D.Fleder and K.Hosanagar. Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. *Management science*, 55(5):697–712, 2009.

[7] A. Freitas, J. a. Oliveira, S. O'Riain, E. Curry, and J. a. Pereira da Silva. Treo: Best-Effort Natural Language Queries over Linked Data. In *Proceedings of NLDB 2011 (poster)*, pages 286–289, 2011. Springer Berlin Heidelberg.

[8] A. Freitas, J. a. G. Oliveira, S. O'Riain, E. Curry, and J. a. C. P. Da Silva. Querying linked data using semantic relatedness: A vocabulary independent approach. In *Proceedings of NLDB'11*, pages 40–51, 2011. Springer-Verlag.

[9] M. Ge, C. Delgado-Battenfeld, and D. Jannach. Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In *ACM RecSys '10*, pages 257–260, 2010. ACM.

[10] A. Gunawardana and C. Meek. A unified approach to building hybrid recommender systems. In *ACM RecSys '09*, pages 117–124, 2009. ACM.

[11] T. H. Haveliwala. Topic-sensitive pagerank. In *ACM WWW '02*, pages 517–526, 2002. ACM.

[12] G. Jeh and J. Widom. Simrank: A measure of structural-context similarity. In *ACM KDD '02*, pages 538–543, 2002.

[13] J. Jiang and D. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proc. of the Int'l. Conf. on Research in Computational Linguistics*, pages 19–33, 1997.

[14] D. Lin. An information-theoretic definition of similarity. In *Proceedings of ICML '98*, pages 296–304, 1998. Morgan Kaufmann Publishers Inc.

[15] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer, 2011.

[16] V. C. Ostuni, T. Di Noia, E. Di Sciascio, and R. Mirizzi. Top-n recommendations from implicit feedback leveraging linked open data. In *ACM RecSys '13*, pages 85–92, 2013. ACM.

[17] V. C. Ostuni, T. Di Noia, R. Mirizzi, and E. Di Sciascio. A linked data recommender system using a neighborhood-based graph kernel. In *Proceedings of EC-Web'14*, Lecture Notes in Business Information Processing. Springer, 2014.

[18] A. Passant. Dbrec: Music recommendations using dbpedia. In *Proceedings of ISWC'10*, pages 209–224, 2010. Springer-Verlag.

[19] A. Passant. Measuring semantic distance on linking data and using it for resources recommendations. In *AAAI Spring Symposium: Linked Data Meets Artificial Intelligence*. AAAI, 2010.

[20] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of IJCAI'95*, pages 448–453, 1995.

[21] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *ACM WWW '01*, pages 285–295, 2001.

[22] A. Schlicker, F. S. Domingues, J. RahnenfÃijhrer, and T. Lengauer. A new measure for functional similarity of gene products based on gene ontology. *BMC Bioinformatics*, 7:302, 2006.

[23] B. Shao, T. Li, and M. Ogihara. Quantify music artist similarity based on style and mood. In *ACM WIDM '08*, pages 119–124, 2008. ACM.

[24] P. D. Turney and P. Pantel. From frequency to meaning: Vector space models of semantics. *J. Artif. Int. Res.*, 37(1):141–188, 2010.

[25] S. Vargas and P. Castells. Improving sales diversity by recommending users to items. In *ACM RecSys '14*, pages 145–152, 2014.

[26] R. S. Wills. Google's pagerank: The math behind the search engine. *Math. Intelligencer*, pages 6–10, 2006.

[27] W. Yu, X. Lin, and W. Zhang. Towards efficient simrank computation on large networks. In C. S. Jensen, C. M. Jermaine, and X. Zhou, editors, *ICDE*, pages 601–612. IEEE Computer Society, 2013.