Introducing Linked Open Data in Graph-based Recommender Systems

Cataldo Musto[a,*], Pierpaolo Basile[a], Pasquale Lops[a], Marco de Gemmis[a], Giovanni Semeraro[a]

[a] *Universita degli Studi di Bari "Aldo Moro", Department of Computer Science*

Abstract

Thanks to the recent spread of the Linked Open Data (LOD) initiative, a huge amount of machine-readable knowledge encoded as RDF statements is today available in the so-called LOD *cloud*. Accordingly, a big effort is now spent to investigate to what extent such information can be exploited to develop new knowledge-based services or to improve the effectiveness of knowledge-intensive platforms as Recommender Systems (RS).

To this end, in this article we study the impact of the *exogenous knowledge* coming from the LOD cloud on the overall performance of a graph-based recommendation framework. Specifically, we propose a methodology to automatically feed a graph-based RS with features gathered from the LOD cloud and we analyze the impact of several widespread feature selection techniques in such recommendation settings.

The experimental evaluation, performed on three state-of-the-art datasets, provided several outcomes: first, information extracted from the LOD cloud can significantly improve the performance of a graph-based RS. Next, experiments showed a clear correlation between the choice of the feature selection technique and the ability of the algorithm to maximize specific evaluation metrics, as *accuracy* or *diversity* of the recommendations. Moreover, our graph-based al-

*Corresponding author

Email addresses: cataldo.musto@uniba.it (Cataldo Musto), pierpaolo.basile@uniba.it (Pierpaolo Basile), pasquale.lops@uniba.it (Pasquale Lops), marco.degemmis@uniba.it (Marco de Gemmis), giovanni.semeraro@uniba.it (Giovanni Semeraro)

DOI: 10.1016/j.ipm.2016.12.003

gorithm fed with LOD-based features was able to overcome several baselines, as collaborative filtering and matrix factorization.

Keywords: Recommender Systems; PageRank; Graphs; Linked Open Data; Feature Selection; Diversity

1. Introduction

In 2007, the Linked Open Data project [8] was launched to stimulate researchers and organizations publishing their data in RDF[1] format and adopting shared vocabularies, in order to express an agreed semantics and interlink the data to each other. Nine years later, 150 billions[2] of RDF triples and almost 10,000 linked datasets are available on the Web, thus representing a rapidly growing piece of the *big data* puzzle [30].

These interconnected RDF statements form a huge decentralized knowledge base, called Linked Open Data (LOD) cloud (see Figure 1). The LOD cloud covers many topical domains, ranging from government and geographical data to structured information about media (movies, books, etc.) and life sciences. The typical *entry point* to these data is DBpedia [3], the RDF mapping of Wikipedia which is commonly considered as the *nucleus* of the emerging *Web of Data*.

Due to the enormous availability of such machine-readable knowledge, a big effort is now spent to investigate whether and how knowledge-intensive services and applications, as Recommender Systems (RS) [27], may benefit of this *plethora* of data. By considering a typical pipeline carried out by a RS, a very straightforward use of the information encoded in the LOD cloud regards the enrichment of the representation of the *items* to be recommended as well as of the *preferences of the target user*. For example, the movie *The Matrix* is described in the LOD cloud by means of a huge set of properties (see Figure 2).

[1] http://www.w3.org/TR/rdf-concepts/

[2] http://stats.lod2.eu/

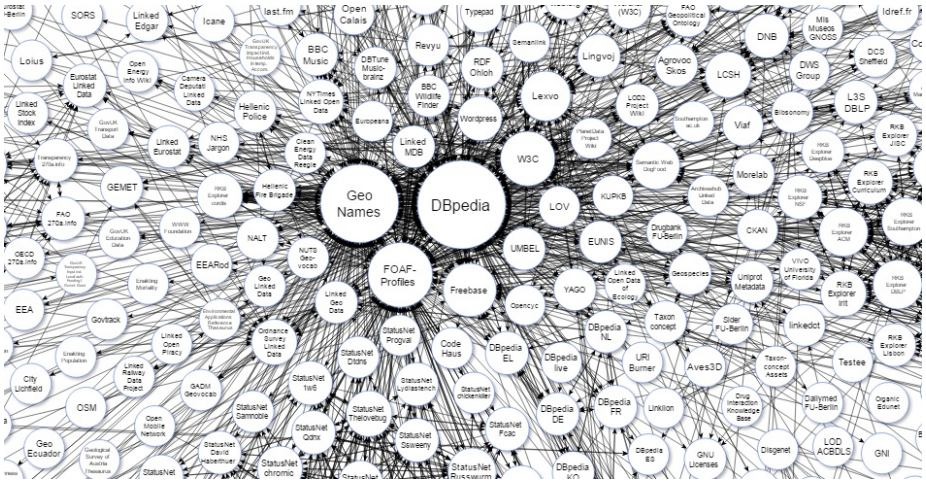


Figure 1: The core of the Linked Open Data Cloud

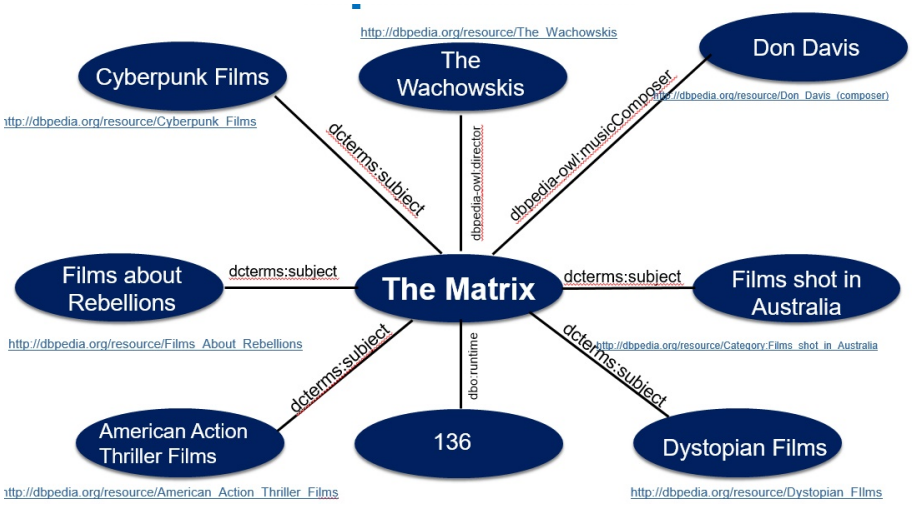


Figure 2: A (tiny) portion of the properties describing the movie "The Matrix"

Clearly, some of the properties (as the *director* of the movie, the *year* or the *composer*) are quite trivial, but many others are very fine-grained and can actually enrich the representation of the items by automatically injecting new and useful knowledge. Accordingly, thanks to these novel data points, user preferences and tastes can be better modeled: as an example, it is possible to infer that a user interested in *The Matrix* may be also interested in *Dystopian* or *Cyberpunk* movies.

Similarly, also recommendation algorithms can be boosted by exploiting the non-trivial connections encoded in the LOD cloud. As shown in Figure 3, the information encoded in DBpedia allows to discover that both *The Matrix* and *The Lost World: Jurassic Park* share some unexpected connections: indeed, by just sifting through the LOD cloud it emerges that both movies, mapped to the URIs HTTP://DBPEDIA.ORG/RESOURCE/THE_MATRIX and HTTP://DBPEDIA.ORG/RESOURCE/THE_LOST_WORLD:JURASSIC_PARK, respectively, share the common feature CATEGORY:FILMS_SHOT_IN_AUSTRALIA, which is encoded through the property DCTERMS:SUBJECT.

It immediately follows that RS technologies can benefit of the information stored in these novel *data silos*, since in a movie recommendation scenario Jurassic Park can be easily suggested to a user who already enjoyed the Matrix. However, the enormous plethora of connections emerging from the LOD cloud can be exploited to generate more daring and unexpected recommendations: as an example, *Minority Report* may be suggested to a user who enjoyed *Cloud Atlas*, given that both movies share the common characteristic of having a director who shot a movie in Australia.

Given the enormous potential of the data encoded in the LOD cloud, in this article we study the impact of such *exogenous knowledge* on the overall performance of a graph-based recommendation framework. We focused our attention on graph-based approaches since they use a *uniform* formalism to represent both *collaborative* and *LOD-based* features. Indeed, in the first case users and items are represented as *nodes* and preferences are represented as *edges*. Similarly, entities from the LOD cloud are represented as *nodes* while the connections

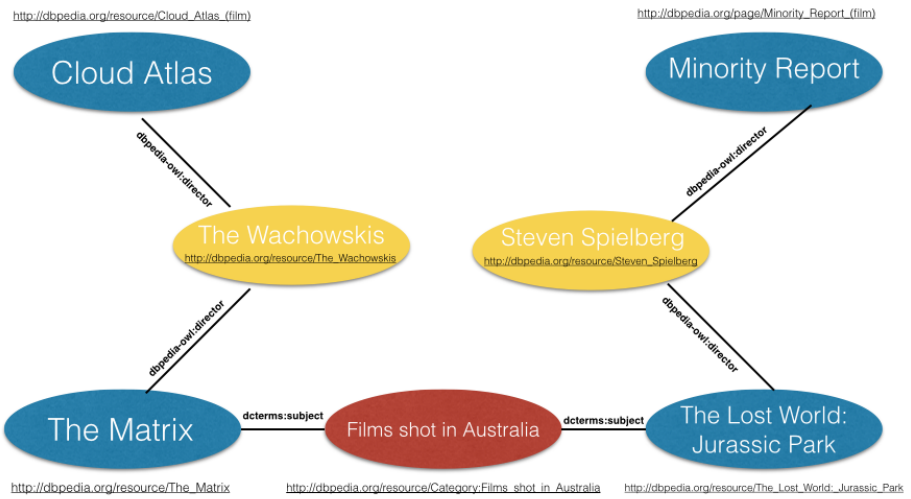


Figure 3: A (tiny) portion of the connections between entities which are encoded in the Linked Open Data cloud.

between them (expressed through RDF properties) are represented as *edges*. It is very straightforward that through a simple mapping of the items to be recommended with the URIs available in the LOD cloud, both representations can be connected and merged in a unique and powerful formalism. Given such a representation, we adopted PageRank with Priors [20] as graph-based recommendation algorithm.

In this work we also compared several techniques to automatically select the best subset of LOD-based properties (that is to say, the best subset of *edges* modeled in the resulting representation), with the aim to investigate to what extent a specific feature selection technique can influence the overall behavior of the algorithm and can *endogenously* lead to a higher *accuracy* or a higher *diversity* of the recommendations. In this case, the experimental evaluation showed a clear correlation between the choice of the feature selection algorithm and the ability of the RS to maximize a specific evaluation metric. Moreover, our graph-based algorithm fed with LOD-based features was able to overcome several state-of-the-art baselines, further confirming the insight that LOD-based

features can be effectively exploited in RS research.

To sum up, the contributions of the paper can be summarized as follows:

- We investigate the impact of the integration of the knowledge coming from the LOD cloud in a graph-based recommendation framework.
- We propose a methodology to automatically feed a graph-based recommendation algorithm with features coming from the LOD cloud.
- We give guidelines to drive the choice of the feature selection technique, according to the needs of a specific recommendation scenario (i.e., maximize accuracy, maximize diversity).
- We validate our methodology by evaluating its effectiveness with respect to several state-of-the-art datasets.

The rest of the paper is organized as follows: Section 2 analyzes related literature. The description of our recommendation methodology is the core of Section 3, while an overview of the feature selection techniques is provided in Section 4. Next, the details of the experimental evaluation we carried out are described in Section 5. Finally, Section 6 sketches conclusions and future work.

2. Related Work

This work investigates two different research lines: graph-based RSs and LOD-based RSs.

In this section we present the current literature in both areas.

2.1. Graph-based Recommender Systems

A very early work in the area of graph-based RSs is due to Aggarwal et al. In [2], they present Horting, an approach which models users as nodes in a graph and connects users according to their similarity. Given this representation, predictions are calculated by walking the graph to nearby nodes and combining the opinions of the nearby users. The adoption of a graph-based topology is

also the core of several work proposed by Huang et al. [22, 23]. In such articles the authors propose a two-layer graph structure which is applied in both an e-commerce and a book recommendation scenario. According to this model, items and users are represented as nodes on two separate layers. Next, transactional data (e.g. a user purchasing an item) are exploited to link nodes in different layers. Given such a representation, several recommendations methods as direct retrieval (e.g. retrieving products similar to the target customers previous purchases) or association rules mining are exploited to generate suggestions.

Next, graph-based representations have been largely adopted to model the concept of *trust* in RSs [39]. As an example, in [24] the authors defined a trust-based network among the users which is exploited to overcome the *cold start*, a typical issue of RSs. Indeed, in this setting ratings of trustful users (instead of *similar users*, as in classic *collaborative filtering* algorithms [16]) are exploited to generate recommendations. A similar attempt towards this research direction is due to Golbeck et al. [18], who modeled trust between users in a movie RS. In both cases the experimental evaluation supported the insight that trustful users (or, generally speaking, authoritative nodes of a social network) lead to better suggestions.

More recently, due to the large popularity of PageRank [41], many papers introduced approaches inspired by Random Walk in the area of RS as well. Indeed, several research showed that the analysis of the *social interactions* among users in a social network is very important, since *authoritative nodes* influence other users' choices [50], and this is tremendously important for recommender systems-related scenarios.

The main difference between the approaches inspired by Random Walk lies in the topology of the graph-based representation: for example, FolkRank has been proposed by Hotho et al. [21] for tag recommendation in social bookmarking systems. The model proposed an adaptation of PageRank relying on a graph-based representation of resources along the tags the community used to annotate them. Next, approaches inspired by PageRank gained a lot of interests in movie and video recommendation scenarios: in [9], Bogers proposed

ContextWalk, a movie RS relying on PageRank, which also models tags, genres and actors, while Baluja et al. [4] present a recommender system for YouTube videos based on random walks on the bipartite user-video graph. They evaluate their method on a three-month snapshot of live data from YouTube and they show that such a novel approach is able to outperform typical baselines for video recommendations. Random Walk was also applied to the problem of rater identification in recommender systems. In [48], a relational context-aware graph consisting of four types of nodes, i.e., user, item, rating and time, and the relations between them is used to distinguish common preferences from personal ones, which could have a great potential for improving group recommendation models. Similarly, in [29], the extra knowledge provided by Last.fm users' social activity was able to improve the performance of a recommendation system using the Random Walk method, and was also able to overcome a common memory-based collaborative filtering item recommendation method, augmented with the social knowledge as well. In the domain of music recommendation, more complex approaches have also been proposed to include multiple types of social media information and music acoustic-based content. Indeed, in [11], the recommendation problem is modeled as a ranking problem on a unified hypergraph which models the multitype objects in a music social community and their relations.

Recently, de Gemmis et al. [13] evaluated Random Walk with Restart and investigated to what extent such graph-based representation can lead to *serendipitous* recommendations. Other attempts are due to Gori [19], who defined the graph on the ground of co-viewing users behavior, and Abbassi et al. [1] who combines Personalized PageRank with Spectral Clustering techniques to recommend weblogs according to the underlying link structure. Moreover, a recent work by Noulas et al. [38] exploited personalized Random Walk for a place recommendation algorithm. In this setting, the technique is run over the user-places graph and combines information coming from social networks (connection between users) with venue visits frequency data. As stated by the authors, this novel approach obtained a significant improvement with respect to other mod-

els for place recommendation, thus confirming the effectiveness of approaches inspired by PageRank in a broad range of scenarios.

One of the distinguishing aspects of our work is that none of the current literature investigated the integration of LOD-based data in graph-based RSs. The only recent similar attempt has been presented by Ostuni et al. [40]. In this work the authors extract the paths connecting users to items by merging the information coming from user preferences with those extracted from the LOD cloud. Given this hybrid data model, the relevance of an item for a user is calculated by counting the number of paths connecting that user and that item: the more the paths, the more the relevance. However, differently from that work, we encoded LOD-based features in a graph-based representation and we exploited Page Rank with Priors as recommendation technique.

2.2. LOD-based Recommender Systems

In this section we focus on the exploitation of knowledge specifically encoded in the Linked Open Data cloud to improve recommendations, while the authors in [49] proposed a survey on recommendation scenarios adopting alternative forms of information concerning users and items, such as those extracted from social networks, folksonomies, tags, or reviews and comments.

Research in the area of LOD-based RSs takes its root in the field of ontology-based recommender systems, introduced by Middleton et al. [33]. The preliminary attempts towards the exploitation of Linked Open Data in RS area are due to Passant [42], who proposed a music recommender system based on semantic similarity calculations involving DBpedia properties. The use of DBpedia for similarity calculation is also the core of the work presented by Musto et al. [37]: in that paper music preferences are extracted from Facebook pages of users and are used as input to find other relevant artists and to build a personalized music playlist. Another DBpedia-based similarity measure has been recently proposed by Meymandpour et al. in [32], where the authors adopt the Partitioned Information Content (PIC) [31], a similarity measure inspired by Information Theory and adapted to the scenario of Linked Open Data, to assess about the similarity

of two resources described through LOD properties. Next, such semantic similarity measure is used as backbone of a collaborative recommendation approach to identify items similar to those the target user already voted. Experiments reported by the authors demonstrated how such approach overcomes all the baselines taken into account.

A relevant research line investigated the adoption of data sources belonging to the Linked Open Data cloud as a *feature generation* tool, since they have been largely used to introduce new relevant features to better model user preferences and better represent the items. For example, in [10] the authors present *TasteWeight*, a recommender systems relying on Facebook-based music preferences. In this work DBpedia is exploited to gather one or more labels describing the genre played by each artist extracted from Facebook. Recommendations are generated by querying a SPARQL endpoint with such labels, in order to obtain new artists playing (most of) the genres liked by the target user. Similarly, Baumann et al. [7] extract features from Freebase[3], a large collaborative knowledge base, to describe artists. A similar approach is also proposed by Schmachtenberg et al. [47], who query LinkedGeoData[4] to collect features describing points of interests. In all these papers Linked Open Data are used to avoid the problem of *limited content analysis* [12], which is typical of content-based recommendation approaches.

Subsequent papers have focused on the impact of LOD-based features on the overall accuracy of a recommender system. A relevant paper has been presented by Di Noia et al. [14], who performed a preliminary comparison of different manually-selected LOD properties in a movie recommender system. Similarly, an empirical evaluation of the impact of LOD-based features on several recommendation techniques (as PageRank and text classifiers) has been presented in [35]. Such studies confirmed the usefulness of injecting Linked Open Data into recommender systems since, regardless the specific technique adopteed to gen-

[3] https://www.freebase.com/

[4] http://linkedgeodata.org

erate recommendations, the performance of LOD-enabled RSs often overcome that obtained by several widespread recommendation techniques as collaborative filtering and matrix factorization. This has been further confirmed by several studies performed in many different domains, as book recommendation [44], e-learning resources recommendation [15] and event recommendation [28].

More recently, the use of LOD-based data sources has been the core of the ESWC 2014 Recommender Systems Challenge[5]: in that setting, the best-performing approaches were based on ensembles of several widespread algorithms. Specifically, the winning approach proposed by Basile et al. [5, 6] used Borda Count to aggregate the output coming from Random Forests, Logistic Regression and PageRank with Priors, running on diverse sets of features gathered from the LOD cloud. Similarly, the second best-performing configuration [46] was based on the same ingredients, since it combined different base recommenders, such as collaborative and content-based ones, with a non-personalized recommender based on popularity.

However, differently from this work, none of the above described approaches tackles the issue of identifying and selecting the most promising subset of LOD-based features, since all analysis are performed by using trivial sets of properties, as the *most popular* ones within the LOD cloud or a *manually selected* subset. To this end, in this paper we perform an extensive study aiming to understand to what extent the introduction of a methodology to automatically select the (best) set of features extracted from the LOD cloud can contribute to the overall effectiveness of the graph-based recommendation framework. Finally, another distinguishing aspect of this article is the thorough analysis of the impact of feature selection techniques on the *diversity* of recommendations. Indeed, none of the work presented in literature addresses the problem of investigating whether specific feature selection techniques are endogenously able to maximize specific evaluation metrics as the diversity or the novelty of the recommendations.

It is worth to state that this article significantly extends the results presented

[5] http://2014.eswc-conferences.org/important-dates/call-RecSys

in [34] and [36], where we confirmed the insight that feature selection techniques can be useful to automatically select a subset of LOD-based properties. In this article we further validated those results by extending the experimental protocol to three different (and bigger) datasets and by evaluating the results at different sparsity levels. Moreover, we introduced a thorough analysis of the scalability issues raised by the massive introduction of data points coming from the Linked Open Data cloud. Finally, more challenging baselines as a matrix factorization algorithm including side information were introduced, in order to further emphasize the effectiveness of our graph-based recommendation approach.

3. Methodology

In this section we describe our graph-based recommendation methodology. In the first part some basics of the PageRank with Priors are provided, while the second part describes how we intend to exploit data available in the LOD cloud for our goals. Finally, we motivate the need to adopt feature selection techniques to automatically detect the most relevant features gathered from the LOD cloud.

3.1. Basics of Graph-based Recommendations

The main idea behind our graph-based model is to represent *users* and *items* as *nodes* in a graph. Formally, given a set of users $U = \{u_1, u_2, \dots u_n\}$ and a set of items $I = \{i_1, i_2, \dots i_m\}$, a graph $G = \langle V, E \rangle$ is instantiated. It is worth to note that G is a bipartite graph, since it models two different kind of entities (that is to say, *users* and *items*). Next, an edge connecting a user u_i to an item i_j is created for each positive feedback expressed by that user ($likes(u_i, i_j)$), thus $E = \{(u_i, i_j) | likes(u_i, i_j) = true\}$. Clearly, if each user and each item have at least a *positive* rating, then $|V| = |U| + |I|$.

Given this basic formulation, built on the ground of simple *collaborative*[6]

[6] We just modeled user-item couples, as in collaborative filtering algorithms

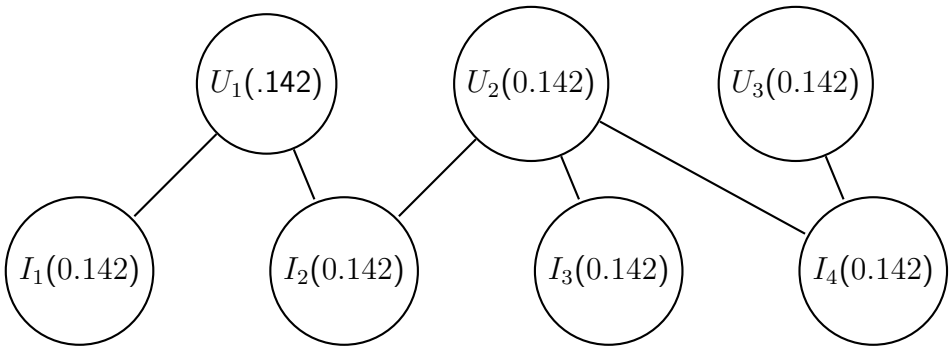


Figure 4: PageRank with users and items.

data points, each item $i \in I$ can be provided with a relevance score. To calculate the relevance of each item, we used a well-known variant of the PageRank called *PageRank with Priors* [20].

The behavior of the classic PageRank is sketched in Figure 4: in the original formulation an evenly distributed prior probability is assigned to each node ($\frac{1}{N}$, where N is the number of nodes). Differently from PageRank, *PageRank with Priors* adopts a non-uniform personalization vector assigning different weights to different nodes to get a bias towards some nodes (specifically, the preferences of a specific user). As shown by Figure 5, in our algorithm the probability was distributed by defining a simple heuristics, set after a rough tuning: 80% of the total weight is evenly distributed among items liked by the users (0% assigned to items disliked by the users), while 20% is evenly distributed among the remaining nodes. Damping factor was set equal to 0.85, as in [41].

Given this setting, the PageRank with Priors is executed for each user (this is mandatory, since the prior probabilities change according to user's feedbacks), and nodes are ranked according to their PageRank score which is in turn calculated on the ground of the connectivity in the graph. The output of the PageRank is a list of nodes ranked according to PageRank scores, labeled as L. Given L, recommendations are built by extracting from L only *non-voted* nodes $i_1 \dots i_n \in I$. An example of the basic behavior of PageRank in a recommendation setting is sketched in Figure 6: in this case user U_3 is provided

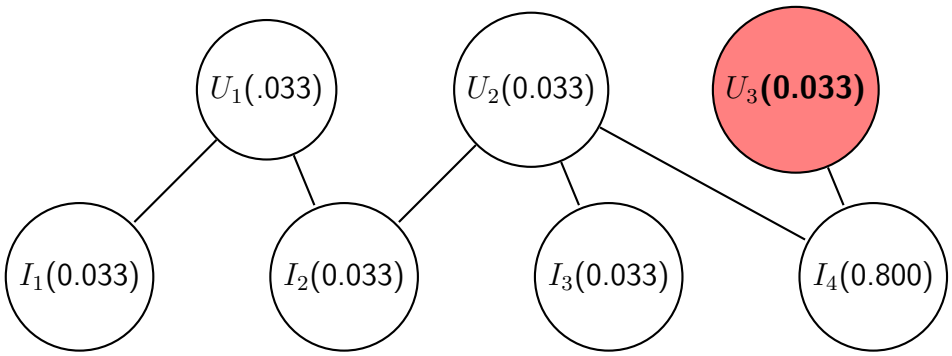


Figure 5: Personalized PageRank on U_3, who previously liked item I_4.

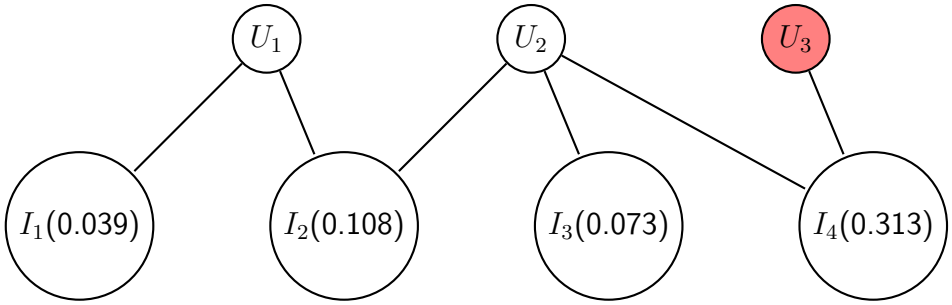


Figure 6: Results of the Personalized PageRank on U_3. List of ranked items: $\{I_2, I_3, I_1\}$.

with I_2 as recommendation, since it is the *non-voted* item node with the highest PageRank score, which is calculated according to the connectivity of the resulting users-items graph.

3.2. Introducing LOD-based features

As stated above, our basic formulation does not take into account any data point different from users' ratings. The insight behind this work is to enrich the above described graph by introducing some *extra* nodes and edges, according to the information available in the LOD cloud. Formally, we want to define an extended graph $G_{LOD} = \langle V_{LOD-ALL}, E_{LOD-ALL} \rangle$, where $V_{LOD-ALL} = V \cup V_{LOD}$ and $E_{LOD-ALL} = E \cup E_{LOD}$. In this case, E_{LOD} is the set of the new connections resulting from the properties encoded in the LOD cloud

(e.g. *director*, *subject*, *genre*, etc.), while V_{LOD} is the new set of nodes (e.g. *Wachowski brothers* or *Keanu Reeves*) connected to the items $i_1 \dots i_m \in I$ through the properties of the LOD cloud.

It is worth to note that we only included nodes and properties which are *directly* connected to the items to be recommended. This choice is due to the results already presented in [35], where it is shown that the introduction of non-direct relationships and nodes leads to an exponential growth of the PageRank running time, but it does not produce a significant improvement in the precision of the recommendation process.

Clearly, in this setting G_{LOD} is a tripartite graph, since beyond users and items the properties gathered from the LOD cloud describing the items are now modeled as well.

By considering again the movie *The Matrix*, as previously explained the information about the director of the movie is encoded by the property modeled in the LOD cloud as HTTP://DBPEDIA.ORG/PROPERTY/DIRECTOR. In this case, an extra node *The Wachowski Brothers* is added to V_{LOD} and an extra edge, labeled with the name of the property, is instantiated in E_{LOD} to connect the movie with its director. Similarly, if we want to model in the new resulting graph also the information about the cast of the movie (encoded by the property HTTP://DBPEDIA.ORG/PROPERTY/STARRING), many new nodes and new edges are defined, for example between *The Matrix* and its main actors, as *Keanu Reeves*. In turn, given that *Keanu Reeves* acted in several movies, many new edges are added to the graph and many new paths now connect different movies: it is worth to note that these paths would not have been available if the only *collaborative data points* were instantiated.

It immediately emerges that, due to this novel enriched representation, the structure of the graph tremendously changes since many new nodes and many new edges are added to the model. As shown in Figure 7, it may happen that many new resources are connected to the items: in our case, each novel node V_i represents a resource gathered from the LOD which is connected to the items in the dataset.

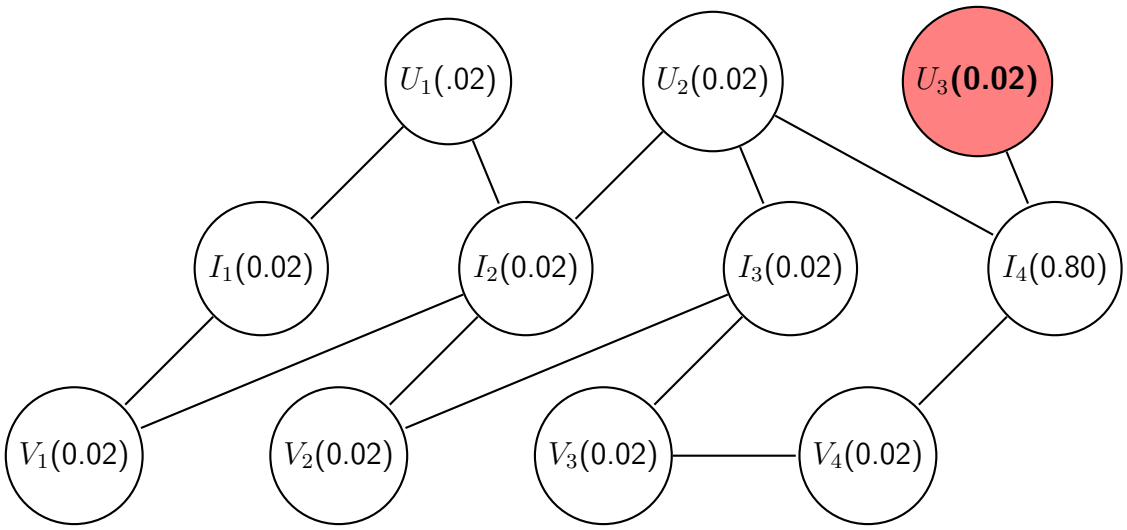


Figure 7: PageRank with Priors on U_3 with users, items and resources gathered from the LOD.

As a consequence, given that relevance scores are calculated on the ground of graph topology, the recommendations generated for a user u_i also change when *LOD-based data points* are taken into account. As shown in Figure 8, it may happen that the ranking of the items changes due to the novel information gathered from the LOD cloud. In this example item I_3 is now ranked as first (instead of second). The first goal of our experimental session will be to investigate whether graph-based RSs can benefit from the introduction of novel LOD-based features.

3.3. Selecting LOD-based features

Thanks to the data points available in the LOD cloud, many new information are encoded in our graph. It is likely that such new data lead our recommendation algorithm to better suggestions, since user preferences as well as items characteristics are better represented thanks to the information available in the LOD cloud. However, as the number of extra nodes and extra edges grows, the computational load of the PageRank with Priors significantly grows as well. As a consequence, it is necessary to reduce the overall load of the algorithm. A

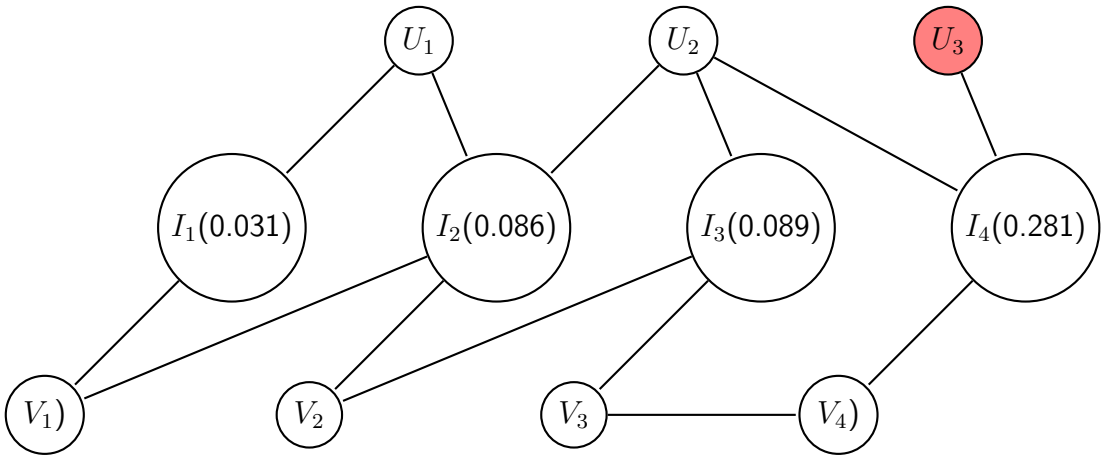


Figure 8: Results of the PageRank with Priorsk on U_3 with users, items and resources gathered from the LOD. List of ranked items: $\{I_3, I_2, I_1\}$.

promising strategy is to investigate to what extent (if any) each of the properties modeled in the graph improves the accuracy of our recommendation strategy, in order to filter out unuseful connections and maintain only the subset of the most useful properties.

A very *naive* approach may be to manually select the most relevant LOD-based features, according to simple heuristics or to domain knowledge (e.g. properties as *director*, *starring*, *composer* may be considered as relevant for the Movie domain, whereas properties as *runtime* or *country* may be not). This basic approach has several drawbacks: first, it requires a manual effort, but it is also strictly domain-dependent, and it is not possible to arbitrarily state that a certain property (as *director*) can be useful and another one can not without an extensive experimental evaluation.

To this aim, in this work we proposed a different methodology, based on the exploitation of *features selection techniques* to automatically select the most promising LOD-based features. Formally, our idea is to take as input E_{LOD}, the overall set of LOD-based properties, and to produce as output $E_{LOD-FS_T} \subseteq E_{LOD}$, the set of properties a specific feature selection technique T returned as relevant. Clearly, the exploitation of a feature selection technique

T also produces a set $V_{LOD-FS_T} \subseteq V_{LOD}$, containing all the LOD-based nodes connected to the properties in E_{LOD-FS_T}.

Given that the application of such techniques causes a change in the topology of the graph, the PageRank with Priors needs to be executed again for each user. In this setting, given a FS technique T, PageRank will be executed against the graph $G_{LOD-T} = \langle V_{LOD-T}, E_{LOD-T} \rangle$, where $V_{LOD-T} = V \cup V_{LOD-FS_T}$ and $E_{LOD-T} = E \cup E_{LOD-FS_T}$.

In the experimental session the effectiveness of all these topologies will be evaluated, by comparing also different techniques for LOD-based features selection.

4. Overview of the feature selection techniques

In this section we provide an overview of the features selection (FS) techniques exploited in this work to identify the most promising subset of LOD-based properties.

Clearly, an exhaustive description of each technique is out of the scope of the paper: we will just provide the insight behind each technique and we will give some methodological details about its behavior. It is worth to state that all the techniques have been exploited in a *top-K selection* setting. That is to say, given a set of features $F = \{f_1 \dots f_r\}$, a FS technique T produces a ranked features list $F_T = \{f_1 \dots f_k\} \subseteq F$, where each feature f_i is ranked according to descending relevance order. Next, the first K features[7] returned by the algorithm are used to feed the graph-based recommendation algorithm. Clearly, in the experimental setting different values for K have been evaluated. Hereafter, a brief description of the seven FS techniques we taken into account follows:

PageRank (PR): PR itself can be used as FS technique. In this setting, differently from the process described in Section 3.1, for each property gathered

[7]Hereafter, the concepts of *features* and *properties* can be considered as synonyms, since in this setting each property gathered from the LOD is a feature of our graph-based model.

from the LOD cloud and for each item to be recommended *a new node* is created. Next, an edge between an *item* node i_j and a *property* node p_k is instantiated whenever an item i_j is described by the property p_k. Finally, classical PageRank is run and the relevance score associated to each property node is calculated. Properties are ranked according to their PR score and the first K are returned. Given that the relevance score of each node is calculated on the ground of its connectivity in the graph, the insight behind this technique is that the more a *property node* in the graph is connected with other nodes, the more is the likelihood that it will be labeled as *relevant*.

Principal Component Analysis (PCA): PCA [26] is a popular technique for FS. This strategy aims at identifying the subset of features (called *Principal Components*, PC) which are relevant, mutually uncorrelated and able to maintain most of the information conveyed by the whole set of features. Operationally, PC are extracted in three steps: first, an item-property matrix X encoding the distribution of the properties along the items is built. Next, the matrix Σ, calculated as the covariance matrix of X, is obtained and its eigenvectors are calculated. Finally, the K eigenvectors of matrix Σ with the highest eigenvalues are labeled as PC and are returned as output of the processing.

Support Vector Machines (SVM): even if SVM [25] has been largely used as classification framework, it can be adopted for FS as well. Indeed, given the previously described set of features F, for each item i a classification hypothesis $H = \theta_0 f_0 + \theta_1 f_1 + \ldots + \theta_r f_r$ is learned by SVM in a *one-vs-all* fashion (that is to say, the item itself along with its features is labeled as *positive* example, while all the other items are labeled as *negative*). The variables $\theta_0 \ldots \theta_r$ are the parameters of the hyphothesis learned by SVM, and their values (typically called *magnitude*) describe the importance of the corresponding feature in the model. When SVM is used as FS technique, features are ranked according to the average magnitude learned over the classes $i_1 \ldots i_m$, and the first K are returned as output of the process.

Chi-Squared Test (CHI): CHI[8] is a statistical test largely adopted to evaluate the dependency between a feature and a *class attribute*. As for SVM, we considered each item as a different *class* and we calculated to what extent a specific feature is relevant for that class. As described in [52], the overall score of each feature is the average value returned by Chi-Squared tests run throughout the available items (classes). The K features obtaining the higher CHI scores are the output of the FS process.

Information Gain (IG): IG measures the decrease of entropy when a feature is given versus when it is absent. Formally, given a feature f_i, IG is calculated as follows:

$$IG(f_i) = E(I) - \sum_{v \in dom(f_i)} \frac{|I_v|}{|I|} * E(I_v) \quad (1)$$

where $E(I)$ is the overall entropy on the data, I_v is the number of items in which feature f_i assumes a value equal to v, and $E(I_v)$ is the entropy of the data calculated only on data where feature f_i has value v. Intuitively, IG of a specific feature is high when the overall sum is high, so a specific feature has a high IG if $E(I_v)$ is low. By exploiting this FS technique, features are ranked according to their IG and the top-K are returned as output.

Information Gain Ratio (GR): the goal of GR is to extend the classical IG to penalize the attributes that assume a broad range of different values throughout the data. This is done by introducing a normalization term calculated as follows:

$$Norm(I, f) = - \sum_{v \in dom(f)} \frac{|I_v|}{|I|} * log \frac{|I_v|}{|I|} \quad (2)$$

Next, GR is calculated as the ratio between $IG(f)$ and $Norm(I, f)$. As for IG, the K features with the highest GR scores are returned as output.

[8] http://en.wikipedia.org/wiki/Chi-squared_test

Mininum Redundancy Maximum Relevance (MRMR): MRMR [43] is a feature selection technique which aims to identify the subset of features which are highly relevant but enough *diverse* from each other. To this end, two functions based on Mutual Information are defined. Given a subset of features, the first one tries to minimize the average mutual information calculated between all the possible couples of features, while the second one tries to maximize their correlation with the target class. In our setting, we extracted our K features by adopting a Greedy strategy to build a set of features which was gradually larger.

5. Experimental Evaluation

Our experiments were designed on the ground of four different research questions:

1. Do graph-based recommender systems benefit of the introduction of LOD-based features?
2. Do graph-based recommender systems exploiting LOD features benefit of the adoption of FS techniques?
3. Is there any correlation between the choice of the FS technique and the behavior of the algorithm?
4. How does our methodology perform with respect to state-of-the-art techniques?

5.1. Experimental Design

Description of the datasets. Experiments were performed by exploiting three state-of-the-art datasets, i.e. MovieLens 1M[9], DBbook and Last.fm[10]. The first one is a widespread dataset for movie recommendations, the second comes from the previously mentioned Linked-Open Data-enabled Recommender

[9] http://grouplens.org/datasets/movielens/1m/
[10] http://grouplens.org/datasets/hetrec-2011/

Systems challenge and focuses on book recommendation, and the latter is a music recommendation dataset relying on Last.fm's users listening habits. Some statistics about the datasets are provided in Table 1.

A quick analysis of the data immediately shows the very different nature of the datasets: MovieLens 1M is the most suitable dataset for *collaborative filtering* algorithms, since both users and items is provided with a significant number of ratings (165.59 per user and 269.88 per item, on average), and this makes the identification of the neighborhood and similarity calculations easier. On the other side, DBbook and Last.fm resulted as more sparse. The first one has a small number of ratings per user (only 11.70 ratings with only 5 ratings as mode) while Last.fm has a very small number of ratings for each item (5.26, with a mode equal to 1). Due to these issues, it is likely that both these datasets will benefit of the integration of the new data points coming from the LOD cloud. Furthermore, DBbook has the peculiarity of being unbalanced towards negative ratings (only 45% of positive preferences), and this makes the recommendation task more challenging.

Table 1: Statistics of the Datasets

	MovieLens 1M	**DBbook**	**Last.fm**
Users	6,040	6,181	1,892
Items	3,883	6,733	17,632
Ratings	1,000,209	72,372	92,834
Positive Ratings	57.51%	45.86%	53.10%
Avg. Rat./User ± stdev	165.59 ± 192.74	11.70 ± 5.85	49.06 ± 5.84
Median/Mode per user	96 / 21	11 / 5	50 / 50
Avg. Rat./Item ± stdev	269.88 ± 384.04	10.74 ± 27.14	5.26 ± 20.62
Median/Mode per item	124 / 1	4 / 1	1 / 1

Experimental protocol. Experiments were performed by adopting different protocols: we used a 80%-20% training-test split for MovieLens 1M, and a 70%-30% one for Last.fm, performed on a *per-user* basis, in order to main-

tain in the training set enough ratings for each user. For DBbook we used the training-test split provided in literature.

Different protocols were also adopted to build user profiles. In MovieLens 1M, given that user preferences are expressed on a 5-point discrete scale, we decided to consider as *positive* only those ratings equal to 4 and 5, while for Last.fm we adopted the same protocol defined in [40]: given that each user was provided with the listening count for each artist, we calculated the average number of listening for that user and we considered as *positive* ratings all the artists whose listening count was over the average. On the other side, the DBbook dataset is already available as *binarized*, thus no further processing was needed.

As recommendation algorithms we used the previously described PageRank with Priors, set as explained in Section 3.1. As introduced in Section 3, we compared the effectiveness of our graph-based recommendation methodology by considering three different graph topologies: G, modeling the basics *collaborative* information about user ratings; G_{LOD}, which enriches G by introducing LOD-based features gathered from DBpedia, and G_{LOD-T} which lighten the load of PageRank with Priors by relying on the features selected by a FS technique T.

In order to evaluate also how our recommendation algorithm is able to deal with *cold-start* recommendation scenarios, we repeated the experiments by exploiting a gradually lower number of ratings for each user. We defined four different sparsity levels: *Given-All*, *Given-50*, *Given-20*, *Given-10*, when all, 50%, 20% and 10% of the ratings available for each user were exploited, respectively. Ratings for each user were randomly chosen, but the proportion between positive and negative ratings was maintained (e.g., a user expressing 90 positive and 10 negative ratings will be modeled in *Given-10* configuration with 9 positive and only one negative rating.) The sparsity of the datasets on varying the sparsity levels is presented in Table 2. Clearly, our expectation is that LOD-based features have a more significant impact on more sparse configurations. Indeed, our insight is that thanks to the injection of such knowledge, both user preferences and items representation are better modeled, and this can

Table 2: Sparsity of the datasets

	MovieLens 1M	**DBbook**	**Last.fm**
Given-All	96.42%	99.85%	99.80%
Given-50	98.21%	99.92%	99.90%
Given-20	99.29%	99.97%	99.96%
Given-10	99.64%	99.98%	99.98%

also overcome the shortage of ratings resulting in more sparse recommendation settings.

In order to enrich the graph G, each item in the dataset was mapped to a DBpedia entry. Specifically, 3,300 MovieLens 1M entries were successfully mapped (85% of the items) while 6,600 items (98.02%) from DBbook were associated to a DBpedia node. In the first case we automatically mapped the items by launching a SPARQL query based on the title of the movie against a DBpedia endpoint, while in the latter we used the mapping made available for the previously mentioned RecSys challenge. Finally, 8,175 Last.fm's artists (46.36%) were mapped to DBpedia. In this case we used the mapping made available in [40]. The items for which a DBpedia entry was not found were only represented by using *collaborative* data points. Overall, MovieLens 1M entries were described through 60 different DBpedia properties, while the number of properties describing books and artists was equal to 70 and 81, respectively.

As feature selection techniques all the approaches previously described in Section 4 were employed, while for the K parameter (the number of LOD-based features) three different values were compared: 10, 30 and 50. The performance of each graph topology was evaluated in terms of *F1-measure* and *Mean Average Precision (MAP)*, in order to also evaluate the goodness of the ranking. Moreover, we also calculated the overall running time[11] of each experiment with a specific graph topology. To answer the third research questions we also evalu-

[11] Experiments were run on an Intel-i7-3770 CPU3.40 gHZ, with 32GB RAM.

ated the *diversity* of the recommendations, calculated by exploiting the classical Intra-List Diversity (ILD) [53], which is calculated as the opposite of the average similarity between all the couples of items in the recommendation set. Formally, let R be a recommendation set, let I_j and I_k be a couple of items in R, let $sim(I_j, I_k)$ be a similarity measure, as the classic *cosine similarity*, the ILD of a recommendation set R can be calculated as follows.

$$ILD(R) = 1 - \frac{\sum_{j,k=1(j \neq k)}^{N} sim(I_j, I_k)}{|R|} \quad (3)$$

Given that *cosine similarity* needs a vector-space representation of the items to be compared, we represented each item relying on features encoded in the Linked Open Data cloud (e.g. subject, genre, director), and we calculated the similarity on the ground of the matching between those values.

Statistical significance was assessed by exploiting Wilcoxon and Friedman tests, chosen after running the Shapiro-Wilk test[12], which revealed the non-normal distribution of the data. Finally, it is worth to note that the source code of our graph-based recommendation framework has been published on GitHub[13]. The code extends the Jung framework[14], a Java library to manage graph-based data, by introducing new code to extract features from the LOD cloud and exploit them in a recommendation setting.

5.2. Discussion of the Results

Experiment 1. In the first experiment we evaluated the introduction of LOD-based features in graph-based recommender systems. Results are presented in Figure 9 and 10.

At first glance, it emerged a very dataset-dependant behavior, since the only dataset which always benefits of the introduction of LOD is Last.fm. With this dataset, the LOD-boosted recommender system always overcame the baseline

[12] http://en.wikipedia.org/wiki/Shapiro-Wilk_test
[13] https://github.com/cataldomusto/lod-recsys/
[14] http://jung.sourceforge.net/

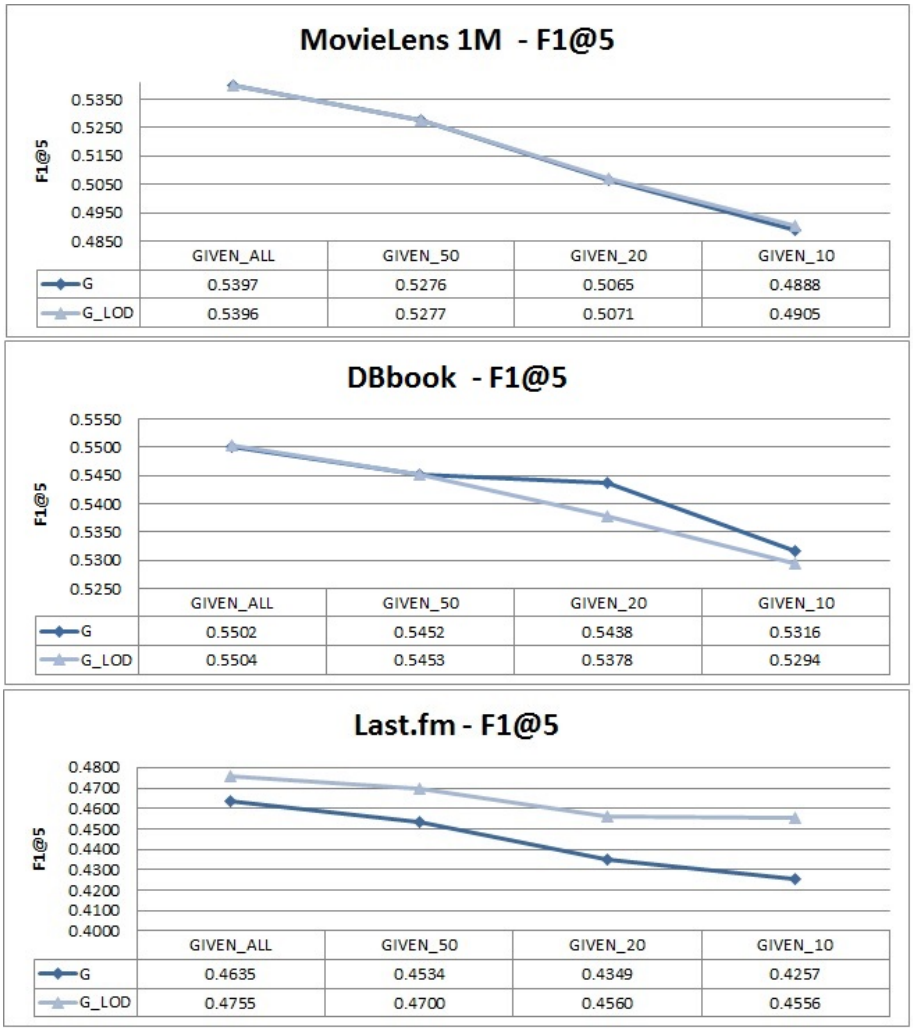


Figure 9: Impact of Linked Open Data on graph-based recommender systems (F1@5)

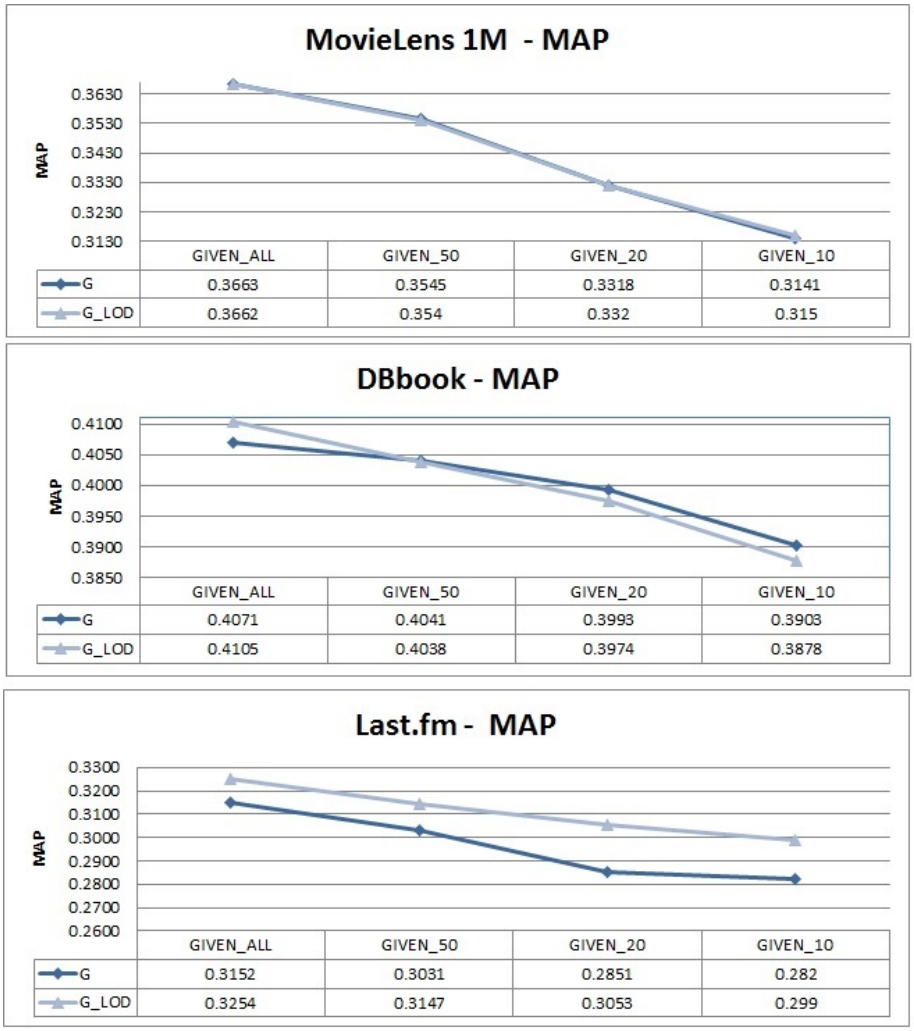


Figure 10: Impact of Linked Open Data on graph-based recommender systems (MAP)

with a statistically significant improvement ($p << 0.0001$) on both F1-measure and MAP, for all the sparsity levels.

Next, the behavior on DBbook dataset was as challenging as expected. The only significant improvement was obtained by considering MAP on *Given_All* configuration. In all the other cases, the G_{LOD} configuration never overcame the baseline. It is likely that this behavior is due to the negatively unbalanced ratio between positive and negative ratings, which made very difficult to model user preferences in a proper way, especially on more sparse configurations. Moreover, the decrease in performance may be also justified by the fact that some of the properties encoded in the LOD cloud (only those about *books*, clearly) are noisy, thus the injection of *all* the knowledge available in the LOD cloud does not provide any significant improvement.

Finally, by considering MovieLens 1M data, results provided interesting outcomes: as expected, our graph-based recommendation methodology *particularly benefits* of the information available in the LOD cloud on *more sparse* configurations. Indeed, on *Given_10* configuration we noted a significant improvement (from 0.4888 to 0.4905 in terms of F1-measure) between G and G_{LOD} configurations. An improvement was also noted on the same configuration in terms of MAP (from 0.3141 to 0.315). This means that the features coming from the LOD cloud are useful when our data model does not hold enough knowledge about user preferences. Indeed, when the number of available ratings increases, the importance of LOD-based features progressively decreases. As reported in Figure 9, on more dense configurations the differences are really tiny and non-significant for both F1-measure and MAP.

Moreover, the computational load requested by the LOD-aware version of PageRank with Priors requests a deeper investigation of the benefits provided by the Linked Open Data. Indeed, as shown in Figure 11, the introduction of Linked Open Data features causes a growth of both nodes and edges in the graph. The growth is almost exponential on DBbook and Last.fm, since many new connections are due to the information available in the LOD are injected, but is less significant for MovieLens 1M, since also the original bipartite user-

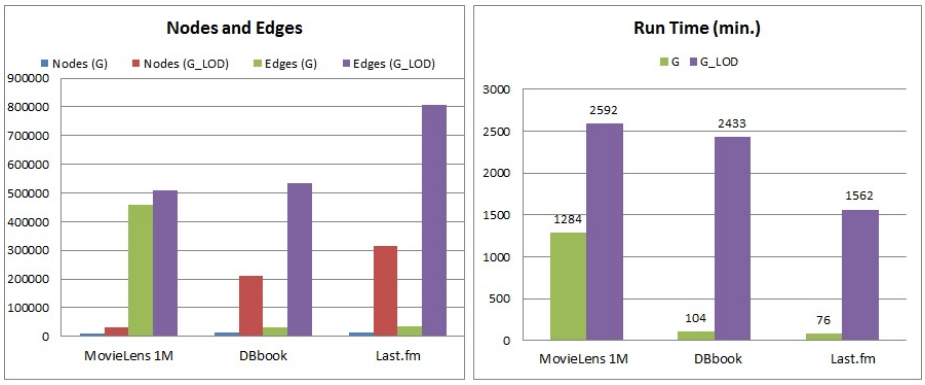


Figure 11: Statistics about Run Time and Graph Topologies - Data report the total run time over all the set of users

item graph already encodes more than 400,000 user-item connections.

However, regardless the peculiarities of each dataset, the run time of the algorithms grows as well. By the way, it is worth to note that the data in Figure 11 refer to the whole set of users. By calculating the running time for each user the results are still acceptable (around 50 seconds for Last.fm, around 20 seconds for DBbook and MovieLens 1M). For the sake of simplicity we only report the data obtained by the *Given_All* configuration, but the trend is similar with more sparse versions of the datasets.

To sum up, the main outcome of this first experiment is that the injection of LOD features needs to be carefully evaluated. Although a global merge of the *collaborative data points* with the information available in the LOD cloud tends to improve the overall F1 on several recommendation settings, the strong increase in terms of run time and computational load may be of hindrance for a complete exploitation of such a representation.

Moreover, we noted that this behavior has a strict connection with the topology of the graph: as the number of available (positive) ratings drops down, most of the connections modeled in the graph depend only on the properties encoded in the LOD. As a consequence, the recommendation process is much more influenced by similarities and connections between items instead of being driven

by user preferences. This can lead a twofold behavior: when the properties gathered from the LOD cloud are particularly significant (as for MovieLens 1M and Last.fm), they can successfully overcome the lack of ratings and can provide accurate recommendations in *cold-start* scenarios, as well. On the other side, when properties tend to be noisy (as for DBbook), overall results drop down.

These issues further underline the need to adopt feature selection techniques: on one hand, they can be useful to filter out non-useful connections and lighten the load of the algorithm. On the other, given that only relevant LOD-based features are maintained in the graph, this can lead to an overall improvement of both F1-measure and MAP.

Experiment 2. In the second experiment we evaluated the impact of all the previously presented feature selection techniques in such recommendation setting. Results of the experiments are provided in Figures 12-13 (for F1@5) and in Figures 14-15 (for MAP). In this case we only show the results of F1@5 and MAP for *Given_All* and *Given_10* configurations. In both cases we avoided to report more results since *Given_50* and *Given_20* obtained result similar *Given_All* and *Given_10*, respectively.

In the plots we show the results obtained by the algorithm fed with the first K properties returned by each of the seven feature selection techniques previously presented, with $K = 10, 30$ and 50. In order to easily read the outcomes of the experiment, we also reported an horizontal line showing the result obtained by the baseline G_{LOD}. The overall best-performing configuration is further highlighted with a star.

The first outcome emerging from the results is that feature selection is able to improve the overall F1-measure obtained by our graph-based recommendation strategy. For example, by considering MovieLens 1M data, on varying the value of the K parameter all the techniques overcome the baseline at least once. The only exception to this trend are represented by F1@5 calculated on the *Given_10* configuration. Indeed, in this case only four out seven techniques overcome the baseline (even with a small, but significant gap). However, by considering MAP, all the techniques are able to improve the overall effectiveness of the algorithm.

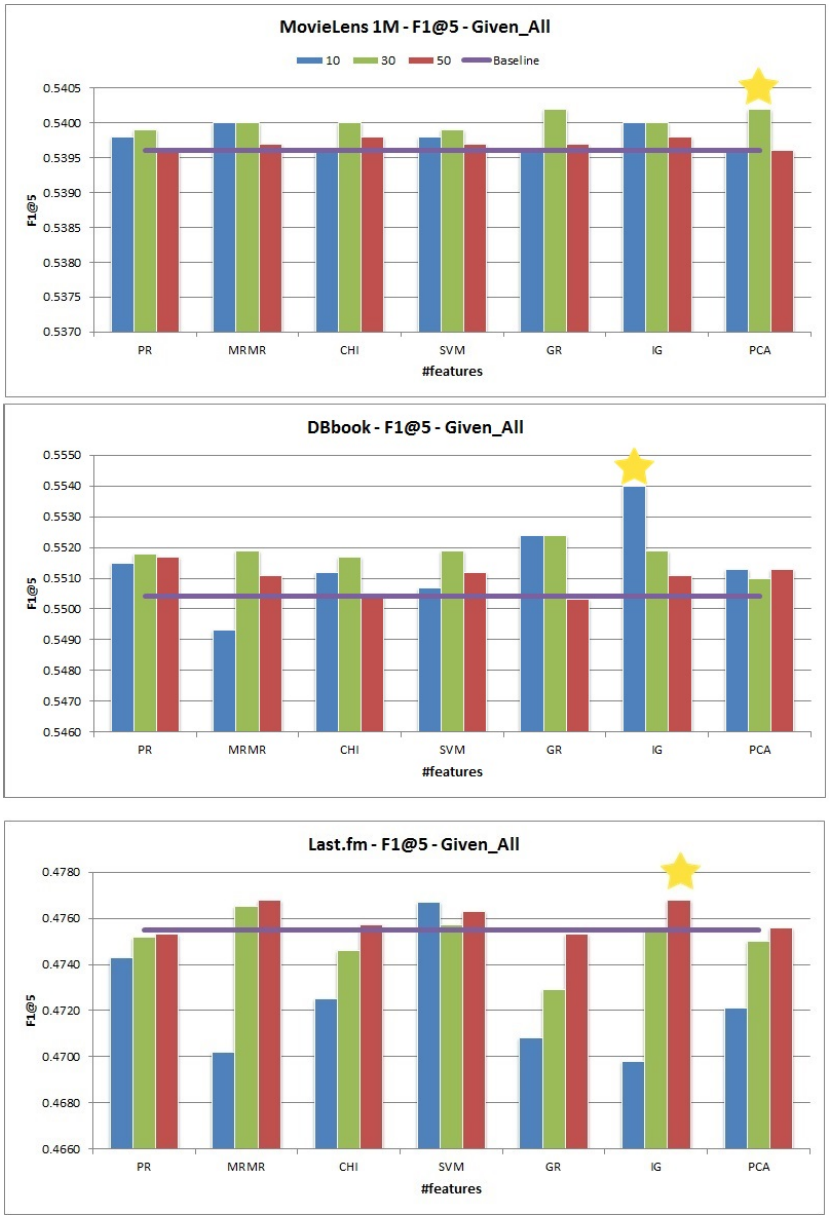


Figure 12: Impact of feature selection techniques on LOD-aware graph-based recommender systems - F1@5 - Given_All configuration

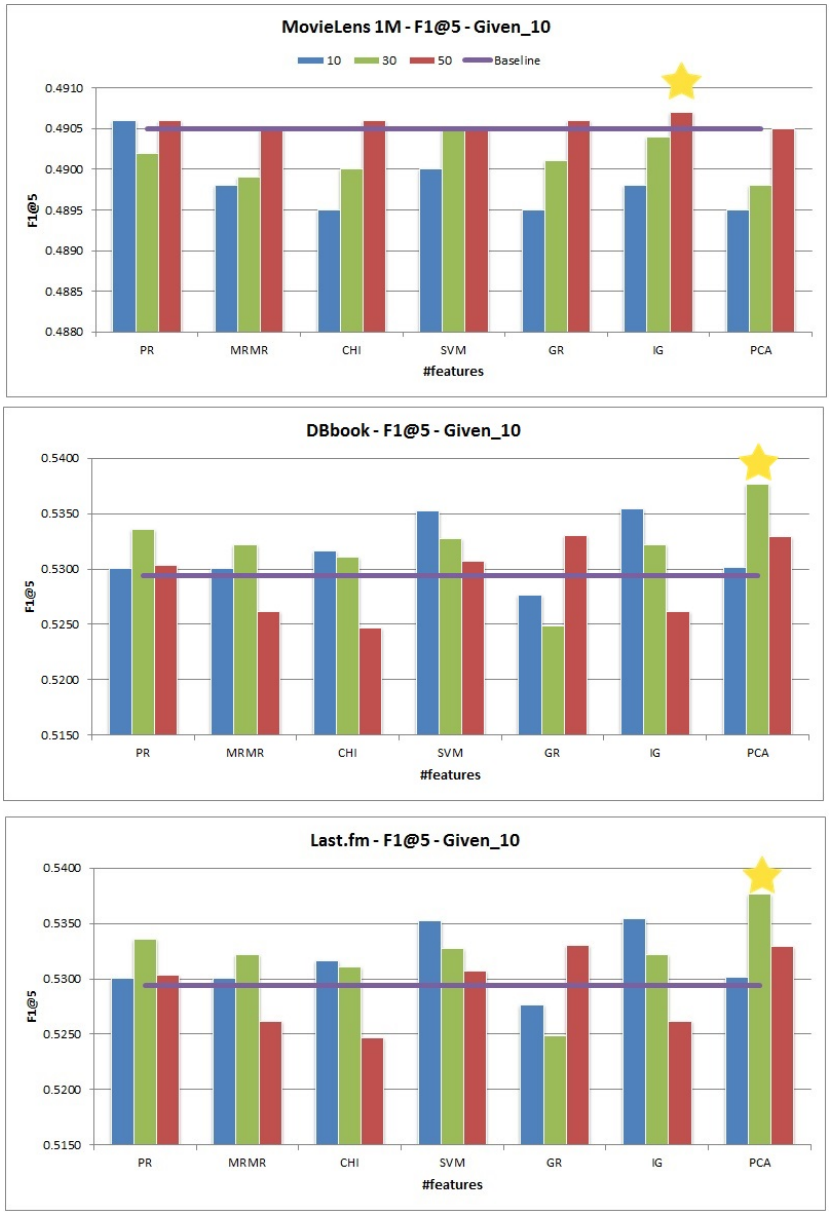


Figure 13: Impact of feature selection techniques on LOD-aware graph-based recommender systems - F1@5 - Given_10 configuration

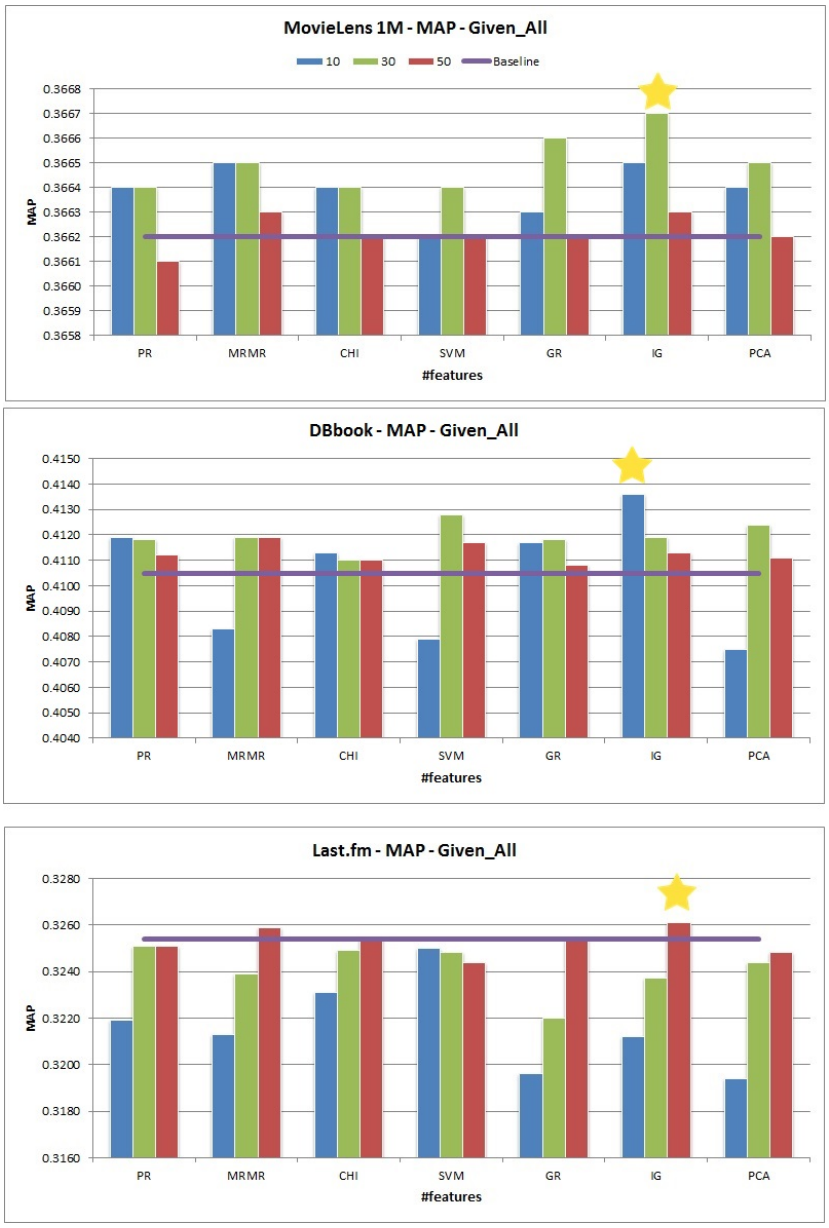


Figure 14: Impact of feature selection techniques on LOD-aware graph-based recommender systems - MAP - Given_All configuration

Figure 15: Impact of feature selection techniques on LOD-aware graph-based recommender systems - MAP - Given_10 configuration

A similar behavior is also noted for DBbook and Last.fm, since the horizontal line representing the baseline is often overcame by the configurations based on features selection. This confirms one of the insight behind this work.

Moreover, by deeply analyzing the results, a strong connection between the topology of the graph and the feature selection strategy to be adopted was noted. Indeed, when the dataset is more dense (as MovieLens 1M, in the *Given_All* configuration) enough collaborative data points are modeled in the graph, thus it is not necessary to perform a strong feature selection. It is not by chance that the best performing configuration is obtained with $K = 30$ for both F1-measure and MAP. This means that by injecting half of the properties available in the LOD cloud we can obtain the highest accuracy.

Alternatively, when the data gets more sparse, we noted a twofold behavior: when LOD-based features are meaningful as for Last.fm (as emerged in Experiment 1), it is a good choice to model all the data points available. Indeed, if we consider the *Given_All* configuration on Last.fm data it immediately emerges that configurations run with $K = 50$ overcome the others for all the techniques on both F1@5 and MAP. This was someway expected since, as shown in Experiment 1, even by injecting all the 81 available features gathered from the LOD the algorithm was able to overcome the baseline, thus it was not necessary to strongly reduce the number of features modeled in the graph.

On the other side, when LOD-based features are noisy as for DBbook, a more aggressive feature selection is necessary to obtain better results. Indeed, in this case we obtained the best results with $K = 10$ and $K = 30$. It is also worth to note that thanks to feature selection our strategy always overcome both G and G_{LOD} baselines.

It is also worth to note that the results emerging from the experiments are confirmed regardless the specific evaluation metrics. Indeed, for DBbook the same configuration obtained the best results on both F1@5 and MAP (IG with 10 features for the Given_All configuration and PCA with 30 features for the Given_10 configuration). Similarly, IG with 50 features and PCA with 30 features were the best-performing configuration on Last.fm for both the the

metrics on Given_All and Given_10 configurations, respectively. Little variations were noted on MovieLens 1M: PCA with 30 features was the best-performing configuration in terms of F1@5 while IG (with the same number of features) obtained the best results in terms of MAP. The unique controversial behavior was noted on the more sparse version of MovieLens 1M dataset: while on one side IG with 50 features obtained the higher F1-measure, only 10 features were enough to get the best results in terms of MAP. This behavior is probably due to the fact that an higher number of features can improve the recall of the algorithms (which influences the F1@5), while a very small subset of features is good enough when only the precision has to be considered.

However, regardless this consideration, the overall behavior for both the metrics confirmed the soundness of the approach, since features selection techniques significantly overcame all the baselines took into account in this experiment.

To sum up, some methodological outcomes emerged from this experiment: first, results showed a clear connection between the sparsity of the data and the need for feature selection. A preliminary comparison between the F1-measure and the MAP obtained by G and those obtained by G_{LOD} give the information about how strong the feature selection should be. When G_{LOD} performs worse than the baseline, it is likely that most of the properties gathered from the LOD are noisy, thus it is necessary to filter them out before running the algorithm. Moreover, results let clearly identify the best-performing features selection techniques, since PCA and IG obtained the best results in most of the comparisons.

Next, we compared the best-performing configurations based on feature selection with the results shown in Experiment 1. Results are presented in Figures 16 and 17. These plots give a complete comprehension of the usefulness of injecting LOD-based knowledge into graph-based recommender systems, since regardless the specific dataset or the specific sparsity level, the LOD-aware configurations boosted by feature selection overcome the baseline with significant gaps. Gaps are significant also on MovieLens 1M data: even if differences are very tiny (due to the fact that most of the PageRank calculations are due to the

original bipartite user-item graph, which is very huge), the improvement was constant over all the users, and this led to the significance of the results. Specifically, Last.fm obtained an improvement which tended to get larger on varying the sparsity level for both the metrics. Moreover, a notable improvement on more challenging configurations (that is so say, when the data gets more sparse) was noted, i.e. on MovieLens 1M data. This further underlined the effectiveness of the approach, since thanks to feature selection techniques even the configurations which did not overcome the baseline in Experiment 1 here obtained a higher F1-measure. This final outcome is also confirmed on DBbook data, where the application of feature selection techniques on the features gathered from the LOD cloud led to results which are better than the baseline relying on simple collaborative data points, for both F1@5 and MAP.

Beyond the significant increase in the overall F1 obtained by the algorithm, the adoption of feature selection also leads to a significant decrease in the overall load of the PageRank algorithm. As shown in Figure 18, the exploitation of feature selection strongly reduced the running time of the algorithms as well as the number of nodes and edges modeled in the graph. Specifically, the number of nodes and edges was reduced on all the datasets (-23.5% nodes and -3.55% edges on MovieLens 1M, -58.10% nodes and -73.39% edges on DBbook, and -22.48% nodes and -39.00% edges on Last.fm). This led to a decrease in terms of running time on the three datasets equal to -20.05 .%, -95.27%, and -20.93%, respectively.

To sum up, Experiment 2 totally confirmed the insight behind this work, since we showed that the injection of the knowledge gathered from the LOD cloud can improve the overall effectiveness of a graph-based recommender system. Moreover, our methodology to automatically feed the recommender system with LOD-based properties relying on feature selection technique leads to both improvement in F1-measure and a lighten in the overall computation load of the recommendation algorithm.

Experiment 3. In the third experiment we shifted the attention from the accuracy to different evaluation metrics, and we investigated whether the

Figure 16: Comparison of the best-performing configuration based on feature selection with both the baselines G and G_{LOD}. (F1@5)

Figure 17: Comparison of the best-performing configuration based on feature selection with both the baselines G and G_{LOD}. (MAP)

Figure 18: Statistics about Run Time and Graph Topologies with Feature Selection

adoption of a specific FS technique can *endogenously* induce a higher diversity at the expense of a little decrease of F1. In this case we avoid also the discuss the impact of the diversity on the MAP since Experiment 1 and Experiment 2 showed that both the metrics had a comparable behavior.

Results of the experiments are provided in Figure 19. In this case we provide only the results for F1@5 with *Given_All* configuration. The baseline G_{LOD} is identified in all the plots by a red diamond.

By analyzing the results, data showed a very technique-dependent trend: features selection techniques as PR, PCA and SVM tend to improve the F1 measure of the algorithm system without a significant impact on the diversity of the recommendations. Specifically, SVM improved the F1 in several comparisons, while it did not provide any benefit in terms of diversity, with the exception of DBbook data. Similarly, PR led to a significant improvement over the baseline but provides a (little) improvement in terms of diversity over the baseline in almost all the comparisons.

It is worth to note that PCA, which performed as one the best techniques in Experiment 2, here always decreases the diversity of the recommendations on MovieLens 1M and DBbook data. Thus, these results suggest to adopt PCA only in those recommendation settings where the diversity of the recommendations is not a primary requirement. Similarly, the use of GR as features selection

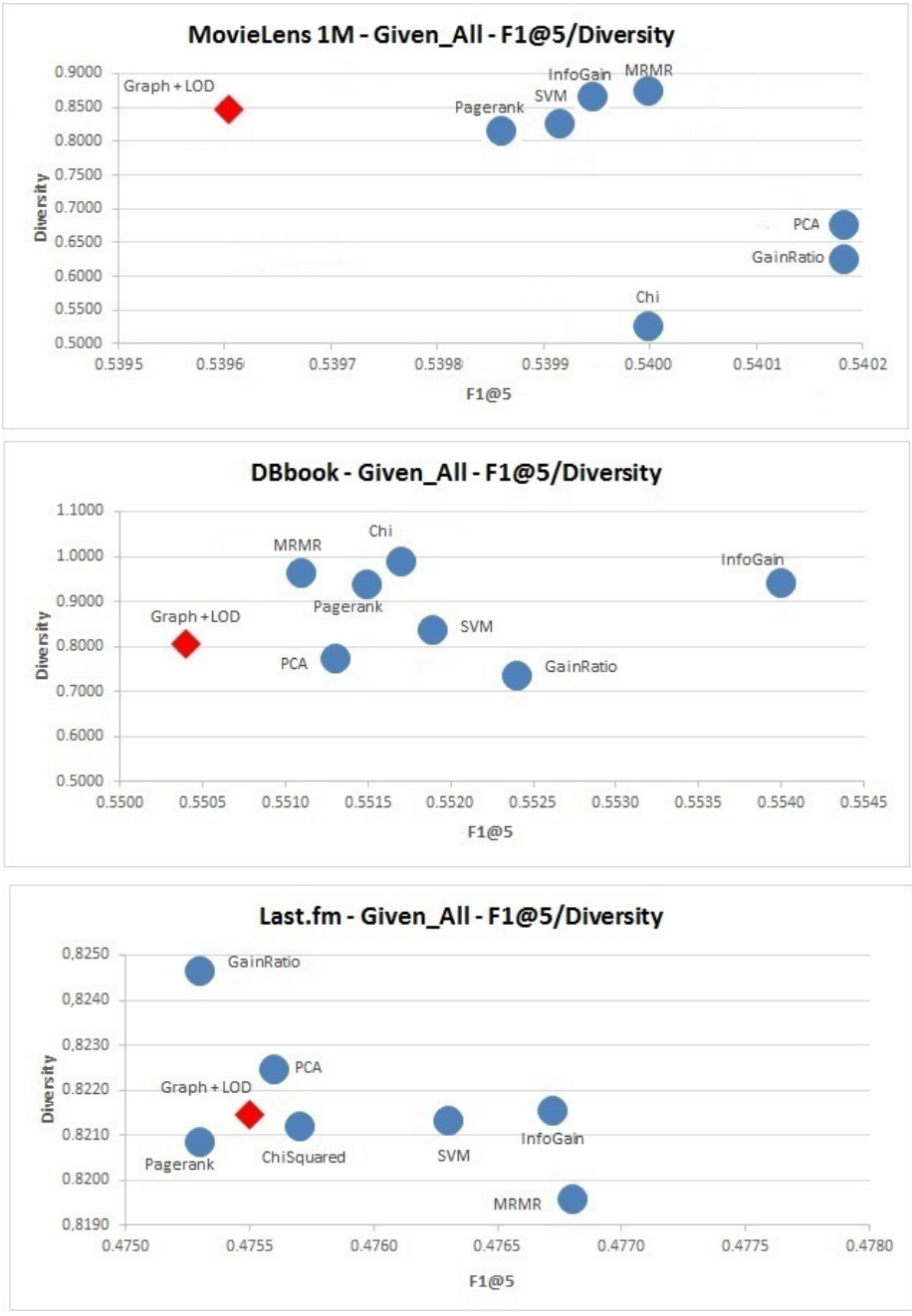


Figure 19: Trade-off between F1 and Diversity, compared to the baseline G_{LOD}. (F1@5)

technique led to a significant decrease in diversity in all the comparisons, with the only exception of Last.fm in the *Given_All* configuration.

Next, the behavior shown by CHI is worth to be more investigated, since this technique performed the worst on MovieLens 1M data (in the *Given_All* configuration it obtained the highest decrease in Diversity) but it improved the metrics on both DBbook and Last.fm data. Finally, the techniques able to provide users with most diverse recommendation sets was IG, which was able to improve the baseline in all the comparisons. Also mRMR obtained good results, since it overcame the baseline in two out of three datasets. As shown in Figure 19, such techniques are often put in the upper right corner of the plots: this means that the adoption of such techniques led to both an improvement in terms of F1 and diversity of the recommendation.

To sum up, these results show that the choice of a particular FS technique has a significant impact on the overall behavior of the recommendation algorithm. As shown in both experiments, some techniques have the ability of inducing a higher diversity (or F1) at the expense of a little of a little decrease in F1 (or diversity, respectively), whereas others can provide a good compromise between both metrics. Clearly, more investigation is needed to deeply analyze the behavior of each technique, but these results already give some general guidelines which can drive the choice of the FS technique which best fits the requirements of a specific recommendation scenario.

By considering the results emerging from this experimental session, it is possible to draw some general guidelines to drive the choice of the feature selection technique:

- PCA and PR are the best FS technique when diversity is not important, since they provide an improvement in terms of F1.
- mRMR is the technique able to maximize the diversity, at the expense sometimes of a little of F1.
- IG is the technique generally emerging as the best-performing one, since

it provides the best compromise between improvement an F1 and increase in diversity.

Experiment 4. In the last experiment we compared the effectiveness of our graph-based recommendation methodology with several state of the art recommendation algorithms. This experiment is further split in two parts: in the first we compare our methodology to some state-of-the-art baseline which did not use any information coming from the LOD cloud, i.e. User-to-User (U2U-KNN) and Item-to-Item Collaborative Filtering (I2I-KNN), a popularity-based approach, the Bayesian Personalized Ranking (BPRMF) which uses Matrix Factorization as the learning model with Bayesian Personalized Ranking (BPR) optimization criterion [45].

Next, we evaluated the impact of LOD-based features also on a hybrid extension of BPR. Specifically, we used the features gathered from the LOD as side information on items using an item-attributes matrix [17]. We built the item-attribute matrix using the same content data used for our graph-based recommendation algorithm. The computation of the recommendations for the previous baselines has been done with the publicly available software library MyMediaLite[15].

For U2U and I2I, experiments were carried out by setting the neighborhood size to 50, 80 and 100 and by using cosine similarity as similarity measure, while BPRMF was run by setting the factor parameter equal to 10, 20, 50, 100 and adopting 0.05 as learning rate. For brevity, we only report the results obtained by the best-performing configurations (80 neighbors for U2U and I2I, 100 factors for BPRMF, 50 factors for BPRMF with side information).

Results showing the comparison between our methodology and the baselines which do not exploit information coming from the LOD are depicted in Figures 20 and 21. As shown in the plots, our graph-based RS significantly outperforms all the baselines for both F1@5 and MAP. Our approach obtained

[15] http://www.mymedialite.net/

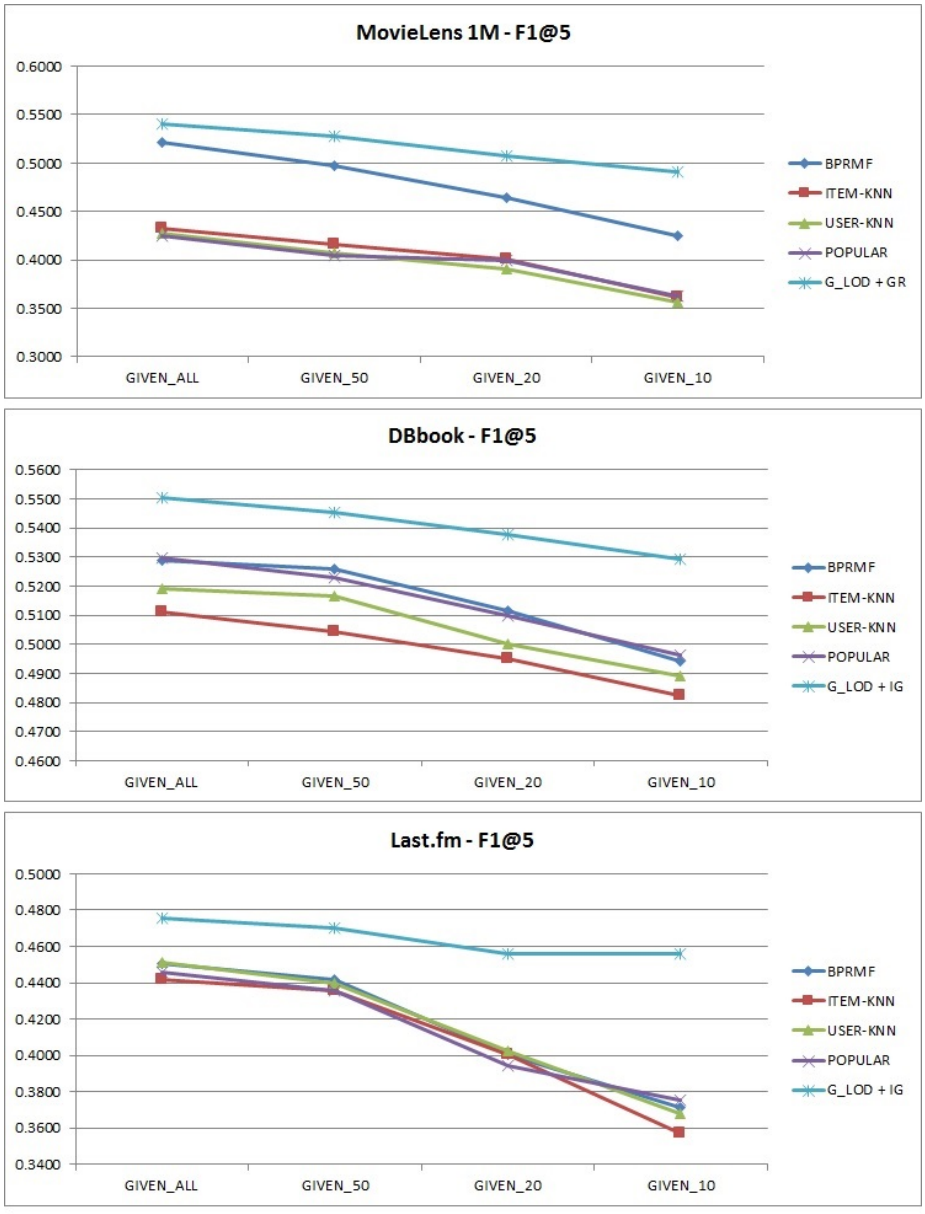


Figure 20: Comparison to baselines. (F1@5)

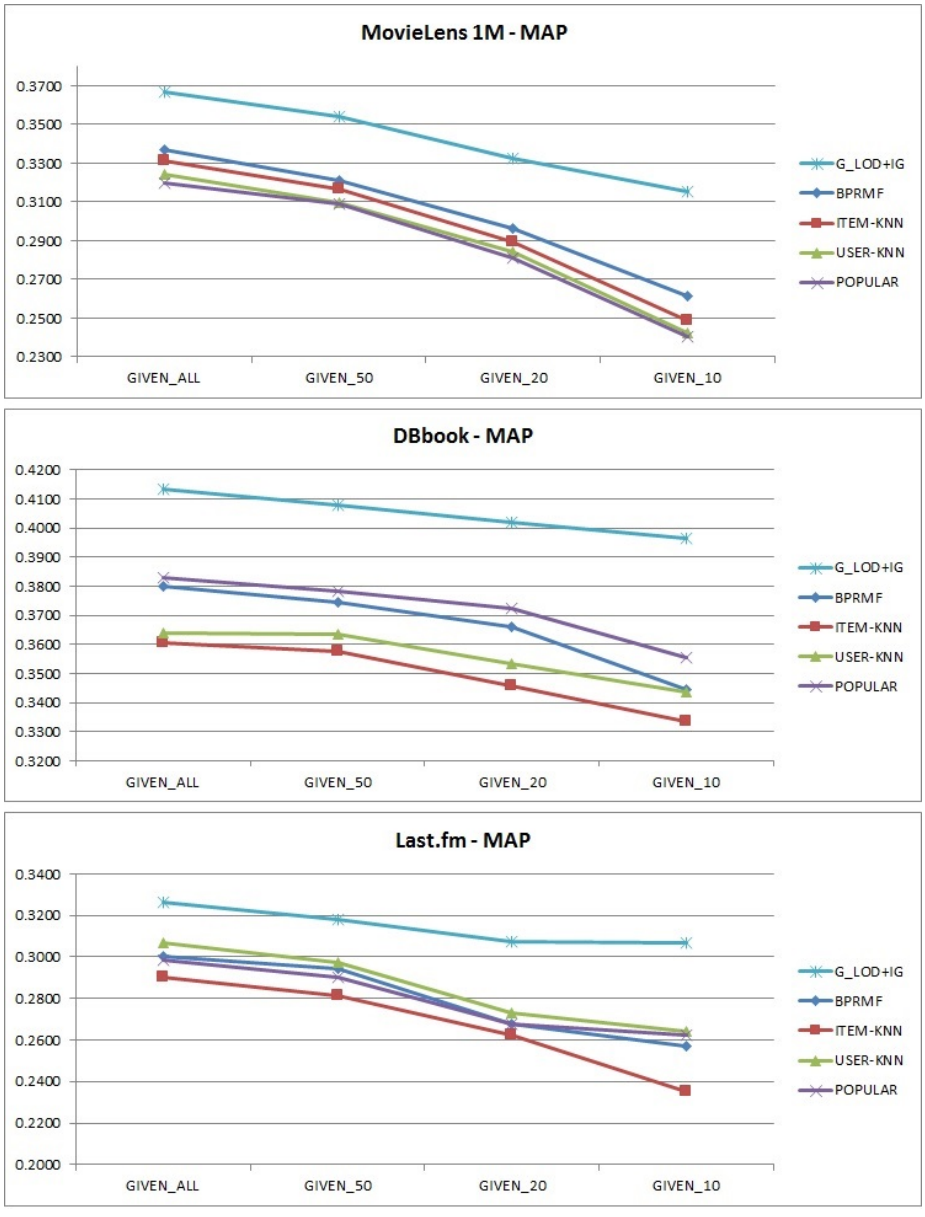


Figure 21: Comparison to baselines. (MAP)

higher results on both the metrics also when compared to a well-performing matrix factorization algorithm as BPRMF. It is worth to note that the gap gets significantly larger when the sparsity gets higher. As expected, our conjecture regarding the adoption of content-based features gathered from the LOD to overcome the shortage of ratings is here empirically confirmed. It is not by chance that, differently from collaborative recommendation strategies, our graph-based framework tends to have more stable values for F1-measure and MAP, even when the number of ratings gets dramatically down.

Finally, results further confirmed how challenging has been the recommendation task on all the dataset we exploited in the experimental sessions: indeed, as the data got more sparse (and negatively unbalanced, as in the case of DBbook) the best-performing configuration was the simple popularity-based baseline. However, also in this setting, our graph-based RS boosted with LOD-based features obtained the best results and definitely confirmed the effectiveness of our approach.

Next, we also evaluated the impact of LOD-based features on a different recommendation algorithm. As previously explained, we extended the original BPRMF by introducing side information extracted from the LOD cloud. In Figures 22 and 23, we showed the impact of side information on BPRMF. respectively. For both the metrics, the outcomes emerging from the experiments are similar to those emerged by discussing Experiment 1, since the impact of side information positively influences both F1@5 and MAP only on Last.fm. As regards DBbook, it seems that the integration of the information coming from the LOD cloud can improve only the MAP, while by analyzing MovieLens 1M data it emerged that side information have a positive impact only on F1@5 on more sparse configurations. These outcomes underlined the usefulness of features selection techniques to automatically process the features gathered from the LOD cloud also for BPRMF.

Similar outcomes also emerged by evaluating the impact of features selection techniques on BPRMF including side information. As shown in Figures 24 and 25, BPRMF can benefit of such techniques since in most of the experiments

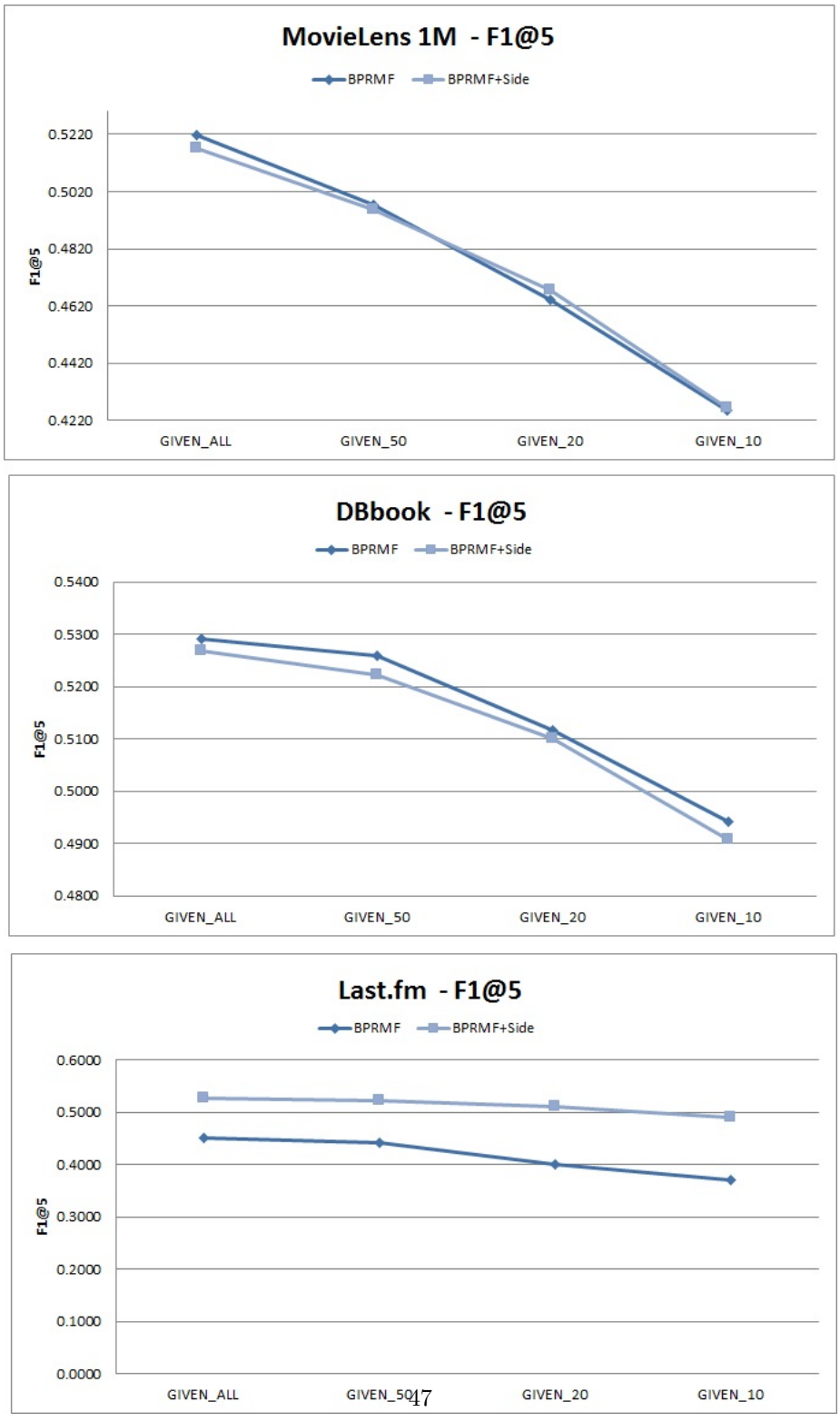
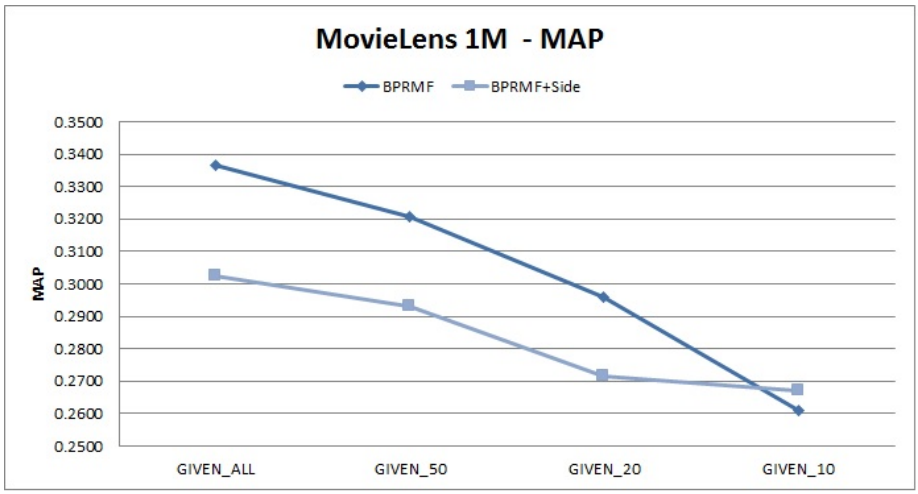
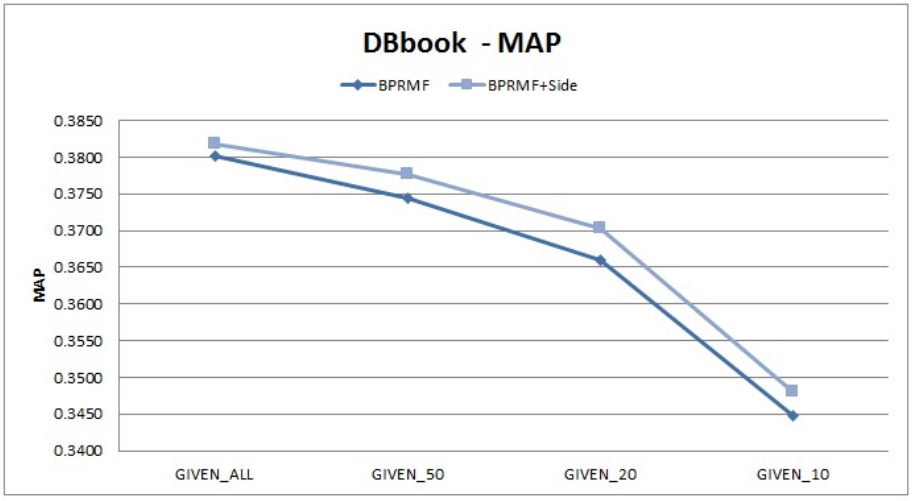


Figure 22: Comparison of BPRMF with BPRMF including side information. (F1@5)





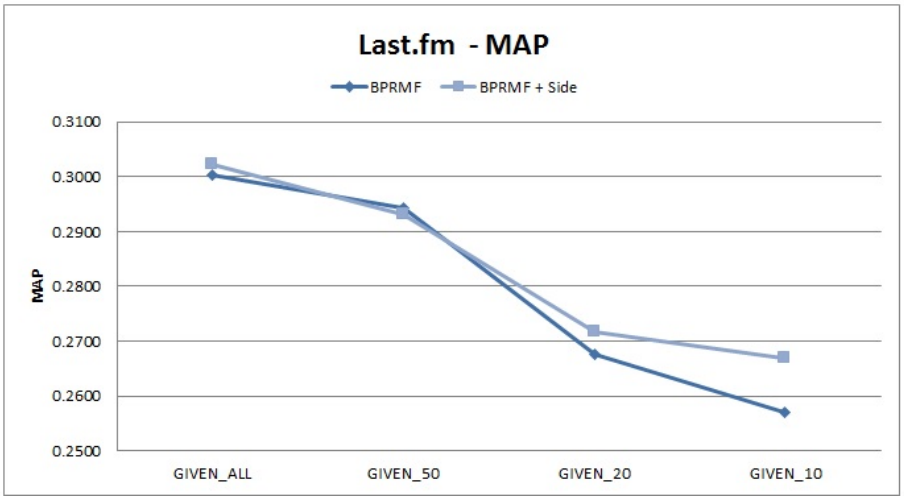


Figure 23: Comparison of BPRMF with BPRMF including side information. (MAP)

settings it is possible to get an improvement of both F1@5 or MAP, as previously reported for PageRank with Priors.

Finally, we also compared the best-performing configuration based on BPRMF with side information with the best-performing configuration based on our graph-based methodology based on PageRank with Priors. Results are presented in Figure 26 and 27, and show how our recommendation algorithm always overcomes BPRMF, even when side information are included. It is also worth to note that the improvement is particularly significant on cold-start recommendation settings (as $Given\_10$ and $Given\_20$ configurations).

These results finally confirmed the effectiveness of our methodology and gave empirical evidence of the soundness of the insights behind our work.

6. Conclusions and Future Work

In this work we proposed a graph-based recommendation methodology based on PageRank with Priors, and we evaluated different techniques to automatically feed such a graph-based representation with features extracted from the LOD cloud. In the experimental session we investigated the impact of LOD-based features and results showed that graph-based RSs can benefit of the infusion of novel knowledge coming from the LOD cloud. Moreover, the adoption of FS techniques further improved the results obtained by our graph-based recommendation methodology, especially in scenarios with high data sparsity. Further investigations showed a clear correlation between the adoption of a specific FS technique with the overall results of the recommender, since some techniques *endogenously* showed the ability of increasing also the diversity of the recommendations generated by the algorithm. We also showed that our methodology was able to overcome several state-of-the-art baselines on all the datasets. A publicly available implementation of the framework as well as of the splits used for the evaluation guaranteed the reproducibility of the experimental results.

To sum up, the main lesson learned from these experiments is that recommender systems can benefit of the information encoded in the LOD cloud, but

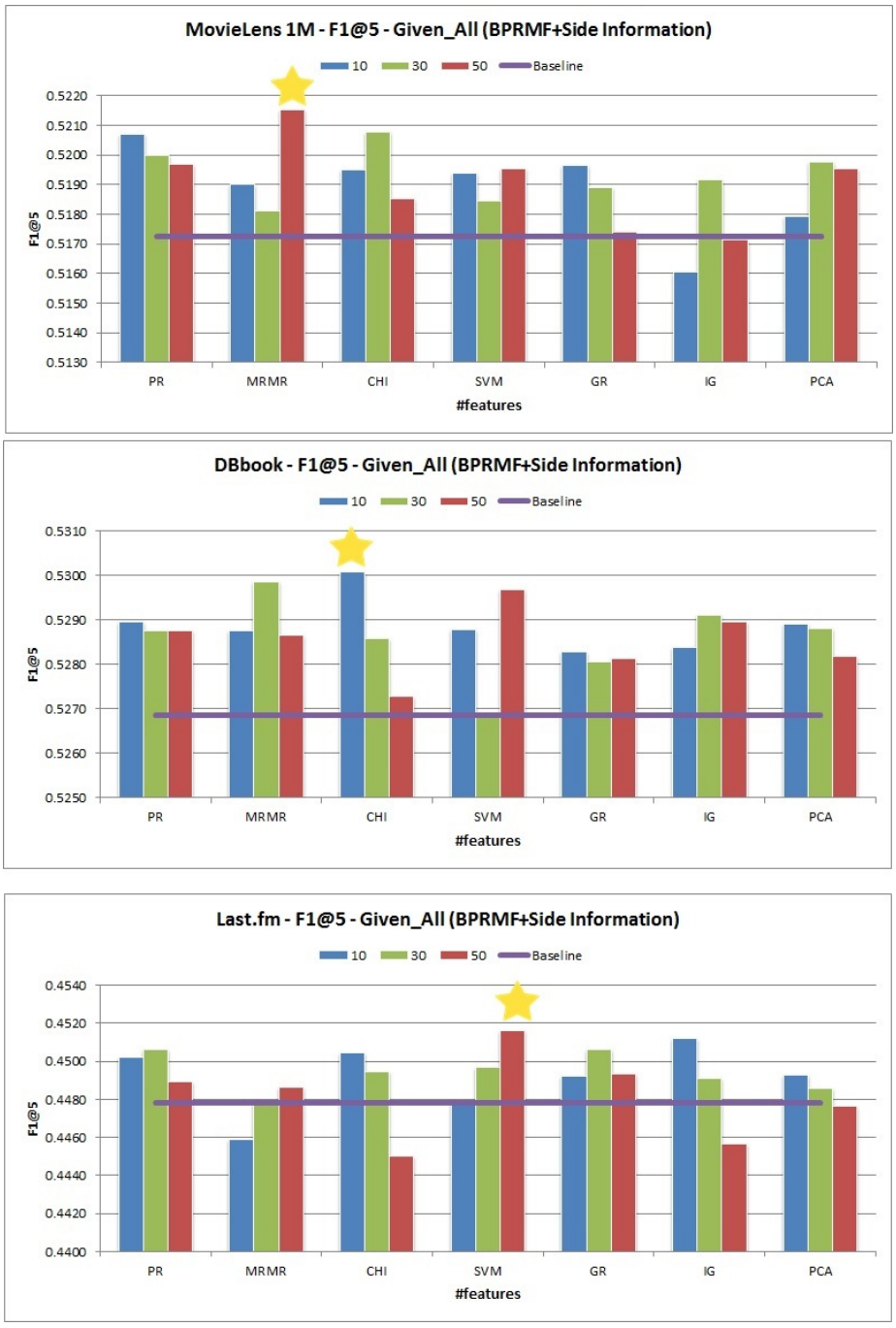


Figure 24: Impact of feature selection techniques on BPRMF with side information. (F1@5)

Figure 25: Impact of feature selection techniques on BPRMF with side information. (MAP)

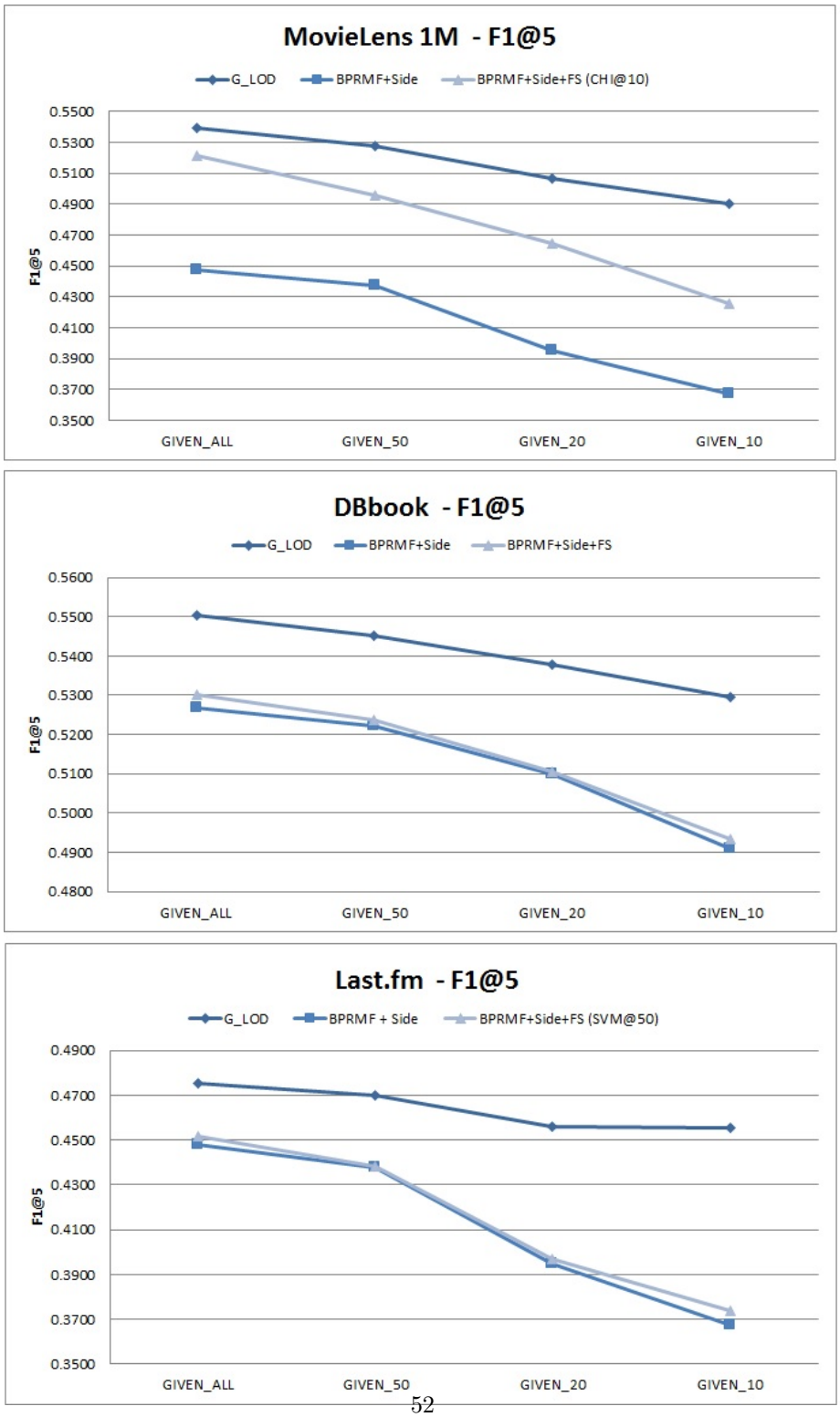
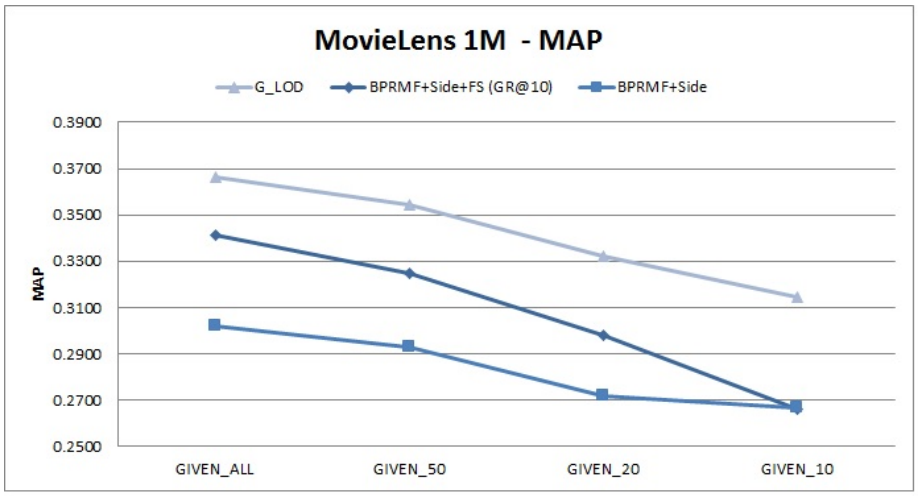
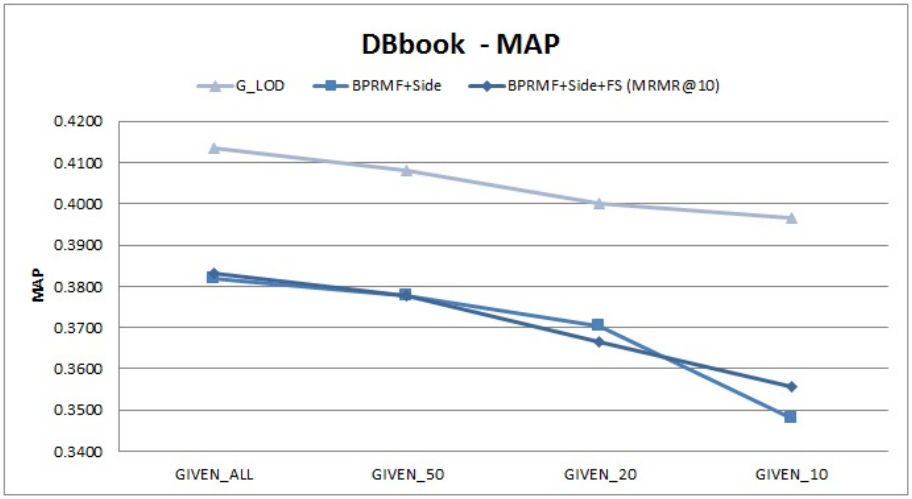


Figure 26: Comparison of BPRMF with side information and features selection to our graph-based methodology (G_{LOD}) on F1@5





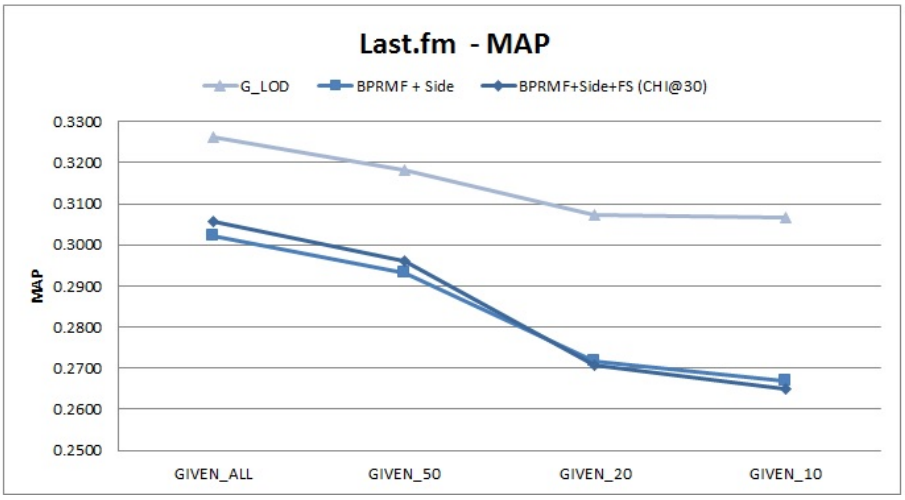


Figure 27: Comparison of BPRMF with side information and features selection to our graph-based methodology (G_{LOD}) on MAP

the overall effectiveness of the algorithm strictly depends on the choice of the parameters of the model, as the number of properties extracted from the LOD cloud as well as the choice of the feature selection algorithm. As expected, experiments did not provide a best overall feature selection techniques neither an optimal number of features to be chosen, since both of them are strictly dependent on the characteristics of the data the recommendation algorithm relies on.

As future work, we will investigate the impact of LOD-based features with different learning approaches as RS relying on text classification techniques (as Random Forests or SVM) or on semantics Vector Space models [51]. Finally, we will also analyze the quality of the recommendations by exploiting different evaluation metrics as *novelty* and *serendipity*.

7. References

[1] Z. Abbassi, V. S. Mirrokni, A recommender system based on local Random Walks and spectral methods, in: Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis, ACM, 2007, pp. 102–108.

[2] C. Aggarwal, J. Wolf, K. Wu, P. Yu, Horting hatches an egg: A new graph-theoretic approach to collaborative filtering, in: Proceedings of the 5th ACM SIGKDD Conference, ACM, 1999, pp. 201–212.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, DBpedia: A nucleus for a web of open data, Springer, 2007.

[4] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, M. Aly, Video suggestion and discovery for YouTube: taking Random Walks through the view graph, in: Proceedings of the 17th International Conference on World Wide Web, ACM, 2008, pp. 895–904.

[5] P. Basile, C. Musto, M. de Gemmis, P. Lops, F. Narducci, G. Semeraro, Ag-

gregation strategies for Linked Open Data-enabled Recommender Systems, in: European Semantic Web Conference (ESWC), 2014.

[6] P. Basile, C. Musto, M. de Gemmis, P. Lops, F. Narducci, G. Semeraro, Content-based Recommender Systems + DBpedia knowledge = Semantics-aware Recommender Systems, in: Semantic Web Evaluation Challenge - SemWebEval 2014 at ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014, Revised Selected Papers, vol. 475 of Communications in Computer and Information Science, Springer, 2014, pp. 163–169.

[7] S. Baumann, R. Schirru, Using Linked Open Data for novel artist recommendations, in: Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR, 2012.

[8] C. Bizer, The emerging web of Linked Data, IEEE Intelligent Systems 24 (5) (2009) 87–92.

[9] T. Bogers, Movie recommendation using Random Walks over the contextual graph, in: Proc. of the 2nd Intl. Workshop on Context-Aware Recommender Systems, 2010.

[10] S. Bostandjiev, J. O'Donovan, T. Höllerer, Tasteweights: a visual interactive hybrid recommender system, in: Proceedings of the sixth ACM conference on Recommender Systems, ACM, 2012, pp. 35–42.

[11] J. Bu, S. Tan, C. Chen, C. Wang, H. Wu, L. Zhang, X. He, Music recommendation by unified hypergraph: combining social media information and music content, in: A. D. Bimbo, S. Chang, A. W. M. Smeulders (eds.), Proceedings of the 18th International Conference on Multimedia 2010, Firenze, Italy, October 25-29, 2010, ACM, 2010, pp. 391–400.

[12] M. de Gemmis, P. Lops, C. Musto, F. Narducci, G. Semeraro, Semantics-aware content-based recommender systems, in: F. Ricci, L. Rokach, B. Shapira (eds.), Recommender Systems Handbook, Springer, 2015, pp. 119–159.

[13] M. de Gemmis, P. Lops, G. Semeraro, C. Musto, An investigation on the serendipity problem in Recommender Systems, Information Processing and Management 51 (5) (2015) 695 – 717.
URL http://www.sciencedirect.com/science/article/pii/S0306457315000837

[14] T. Di Noia, R. Mirizzi, V. C. Ostuni, D. Romito, Exploiting the web of data in model-based Recommender Systems, in: Proceedings of the ACM RecSys Conference, ACM, 2012, pp. 253–256.

[15] S. Dietze, H. Drachsler, D. Giordano, A survey on Linked Data and the social web as facilitators for tel Recommender Systems, in: Recommender Systems for Technology Enhanced Learning, Springer, 2014, pp. 47–75.

[16] M. D. Ekstrand, J. T. Riedl, J. A. Konstan, Collaborative filtering Recommender Systems, Foundations and Trends in Human-Computer Interaction 4 (2) (2011) 81–173.

[17] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, L. Schmidt-Thieme, Learning attribute-to-feature mappings for cold-start recommendations, in: G. I. Webb, B. Liu, C. Zhang, D. Gunopulos, X. Wu (eds.), ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010, IEEE Computer Society, 2010, pp. 176–185.

[18] J. Golbeck, J. Hendler, et al., Filmtrust: Movie recommendations using trust in web-based social networks, in: Proceedings of the IEEE Consumer communications and networking conference, vol. 96, 2006, pp. 282–286.

[19] M. Gori, A. Pucci, V. Roma, Itemrank: a random-walk based scoring algorithm for recommender engines., in: IJCAI, vol. 7, 2007, pp. 2766–2771.

[20] T. H. Haveliwala, Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search, IEEE Trans. Knowl. Data Eng. 15 (4) (2003) 784–796.

[21] A. Hotho, R. Jäschke, C. Schmitz, G. Stumme, K.-D. Althoff, Folkrank: A ranking algorithm for folksonomies, in: LWA, vol. 1, 2006, pp. 111–114.

[22] Z. Huang, W. Chung, H. Chen, A graph model for e-commerce Recommender Systems, Journal of the American Society for information science and technology 55 (3) (2004) 259–274.

[23] Z. Huang, W. Chung, T.-H. Ong, H. Chen, A graph-based recommender system for digital library, in: Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries, ACM, 2002, pp. 65–73.

[24] M. Jamali, M. Ester, Trustwalker: a Random Walk model for combining trust-based and item-based recommendation, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2009, pp. 397–406.

[25] T. Joachims, Text categorization with Support Vector Machines: Learning with many relevant features, Springer, 1998.

[26] I. Jolliffe, Principal Component Analysis, Wiley Online Library, 2002.

[27] P. B. Kantor, L. Rokach, F. Ricci, B. Shapira, Recommender Systems handbook, Springer, 2011.

[28] H. Khrouf, R. Troncy, Hybrid event recommendation using Linked Data and user diversity, in: Proceedings of the 7th ACM conference on Recommender Systems, ACM, 2013, pp. 185–192.

[29] I. Konstas, V. Stathopoulos, J. M. Jose, On social networks and collaborative recommendation, in: J. Allan, J. A. Aslam, M. Sanderson, C. Zhai, J. Zobel (eds.), Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009, ACM, 2009, pp. 195–202.

[30] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, A. H. Byers, Big data: The next frontier for innovation, competition, and productivity.

[31] R. Meymandpour, J. Davis, Linked Data informativeness, in: Y. Ishikawa, J. Li, W. Wang, R. Zhang, W. Zhang (eds.), Web Technologies and Applications, vol. 7808 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 629–637.
URL http://dx.doi.org/10.1007/978-3-642-37401-2_61

[32] R. Meymandpour, J. G. Davis, Enhancing Recommender Systems using Linked Open Data-based semantic analysis of items, in: Proceedings of the 3rd Australasian Web Conference (AWC 2015), vol. 27, 2015, p. 30.

[33] S. E. Middleton, D. De Roure, N. R. Shadbolt, Ontology-based Recommender Systems, in: Handbook on ontologies, Springer, 2004, pp. 477–498.

[34] C. Musto, P. Basile, M. de Gemmis, P. Lops, G. Semeraro, S. Rutigliano, Automatic selection of linked open data features in graph-based recommender systems, in: T. Bogers, M. Koolen (eds.), Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015., vol. 1448 of CEUR Workshop Proceedings, CEUR-WS.org, 2015, pp. 10–13.
URL http://ceur-ws.org/Vol-1448/paper3.pdf

[35] C. Musto, P. Basile, P. Lops, M. de Gemmis, G. Semeraro, Linked Open Data-enabled strategies for top-n recommendations, in: T. Bogers, M. Koolen, I. Cantador (eds.), Proceedings of the 1st Workshop on New Trends in Content-based Recommender Systems co-located with the 8th ACM Conference on Recommender Systems, CBRecSys@RecSys 2014, Foster City, Silicon Valley, California, USA, October 6, 2014., vol. 1245 of CEUR Workshop Proceedings, CEUR-WS.org, 2014, pp. 49–56.
URL http://ceur-ws.org/Vol-1245/cbrecsys2014-paper08.pdf

[36] C. Musto, P. Lops, P. Basile, M. de Gemmis, G. Semeraro, Semantics-aware graph-based recommender systems exploiting linked open data, in:

Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization, UMAP '16, ACM, New York, NY, USA, 2016, pp. 229–237. URL http://doi.acm.org/10.1145/2930238.2930249

[37] C. Musto, G. Semeraro, P. Lops, M. de Gemmis, F. Narducci, Leveraging social media sources to generate personalized music playlists, in: E-Commerce and Web Technologies - 13th International Conference, EC-Web 2012, vol. 123 of Lecture Notes in Business Information Processing, Springer, 2012, pp. 112–123.

[38] A. Noulas, S. Scellato, N. Lathia, C. Mascolo, A Random Walk around the city: New venue recommendation in location-based social networks, in: Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom), IEEE, 2012, pp. 144–153.

[39] J. O'Donovan, B. Smyth, Trust in Recommender Systems, in: Proceedings of the 10th international conference on Intelligent user interfaces, ACM, 2005, pp. 167–174.

[40] V. C. Ostuni, T. Di Noia, E. D. Sciascio, R. Mirizzi, Top-N recommendations from implicit feedback leveraging Linked Open Data, in: Proceedings of the ACM Conference on Recommender Systems, ACM, 2013, pp. 85–92.

[41] L. Page, S. Brin, R. Motwani, T. Winograd, The PageRank citation ranking: bringing order to the web.

[42] A. Passant, dbrec - Music Recommendations Using DBpedia, in: International Semantic Web Conference, Revised Papers, vol. 6497 of LNCS, Springer, 2010, pp. 209–224.

[43] H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy, Pattern Analysis and Machine Intelligence, IEEE Transactions on 27 (8) (2005) 1226–1238.

[44] L. Peska, P. Vojtás, Enhancing recommender system with Linked Open Data, in: Flexible Query Answering Systems, Springer, 2013, pp. 483–494.

[45] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, BPR: Bayesian personalized ranking from implicit feedback, in: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, AUAI Press, 2009, pp. 452–461.

[46] P. Ristoski, E. L. Mencia, H. Paulheim, A hybrid multi-strategy recommender system using Linked Open Data, in: European Semantic Web Conference, 2014.

[47] M. Schmachtenberg, T. Strufe, H. Paulheim, Enhancing a location-based recommendation system by enrichment with structured data from the web, in: Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14), ACM, 2014, p. 17.

[48] Y. Shi, M. Larson, A. Hanjalic, Mining relational context-aware graph for rater identification, in: Proceedings of the 2nd Challenge on Context-Aware Movie Recommendation, ACM, 2011, pp. 53–59.

[49] Y. Shi, M. Larson, A. Hanjalic, Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges, ACM Comput. Surv. 47 (1) (2014) 3:1–3:45.
URL http://doi.acm.org/10.1145/2556270

[50] A. Susarla, J.-H. Oh, Y. Tan, Social networks and the diffusion of user-generated content: Evidence from youtube, Information Systems Research 23 (1) (2012) 23–41.

[51] P. D. Turney, P. Pantel, et al., From frequency to meaning: Vector space models of semantics, Journal of artificial intelligence research 37 (1) (2010) 141–188.

[52] Y. Yang, J. O. Pedersen, A comparative study on feature selection in text categorization, in: ICML, vol. 97, 1997, pp. 412–420.

[53] C.-N. Ziegler, S. M. McNee, J. A. Konstan, G. Lausen, Improving recommendation lists through topic diversification, in: Proceedings of the 14th International Conference on World Wide Web, ACM, 2005, pp. 22–32.