Knowledge-Based Systems 000 (2017) 1–14

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Semantics-aware Recommender Systems exploiting Linked Open Data and graph-based features

Cataldo Musto*, Pasquale Lops, Marco de Gemmis, Giovanni Semeraro

Universita degli Studi di Bari "Aldo Moro", Department of Computer Science, Italy

ARTICLE INFO

Article history:
Received 10 April 2017
Revised 7 August 2017
Accepted 22 August 2017
Available online xxx

Keywords:
Recommender Systems
Linked Open Data
Semantics
Machine learning
Classifiers

ABSTRACT

The recent spread of Linked Open Data (LOD) fueled the research in the area of Recommender Systems, since the (semantic) data points available in the LOD cloud can be exploited to improve the performance of recommendation algorithms by enriching item representations with new and relevant features.

In this article we investigate the impact of the features gathered from the LOD cloud on a hybrid recommendation framework based on three classification algorithms, Random Forests, Naïve Bayes and Logistic Regression. Specifically, we extend the representation of the items by introducing two new types of features: *LOD-based features*, structured data extracted from the LOD cloud, as the *genre* of a movie or the *writer* of a book, and *graph-based features*, computed on the ground of the topological characteristics of both the *bipartite* graph-based representation connecting users and items, and the *tripartite* representation connecting users, items and properties in the LOD cloud.

In the experimental session we assess the effectiveness of these novel features; results show that the use of information coming from the LOD cloud could improve the overall accuracy of our recommendation framework. Finally, our approach outperform several state-of-the-art recommendation techniques, thus confirming the insights behind this research.

1. Introduction

According to its original vision [7], the goal of the *Semantic Web* was to make machine-readable the whole knowledge available on the Web. This enormous effort, that should have been carried out by stimulating the adoption of shared languages as RDF[1] or OWL[2] and protocols as URI, would have enabled a common framework allowing data to be shared and reused across applications, enterprises, and communities.

Unfortunately, more than fifteen years later the full vision of the Semantic Web has yet to be fully accomplished. Some considerable progress towards this direction has been obtained after the recent spread of the Linked Open Data (LOD) initiative [8], whose goal is to stress and emphasize the importance of publishing and making data *publicly* available and *linked* one to each other.

According to recent statistics,[3] thanks to the collaborative effort behind the LOD initiative, 150 billions of RDF triples and almost 10,000 linked datasets are now available in the so-called LOD cloud, a huge set of interconnected semantic datasets whose *nucleus* is commonly represented by DBpedia [1], the RDF mapping of Wikipedia that acts as a *hub* for most of the RDF triples made available in the LOD cloud. Such RDF triples represent, in a structured form, semantic information covering many topical domains, such as geographical locations, people, companies, books, scientific publications, films, music, TV and radio programs, genes, proteins, drugs, online communities, statistical data, and so on.

As an example for the musical domain, Fig. 1 shows a tiny portion of the properties, available in the LOD cloud, that describe the band *The Coldplay*. Such features range from very basic information, such as the fact that the band has its hometown in *London*, or that *Chris Martin, Jonny Buckland, Guy Berryman*, and *Will Champion* are their members, to more interesting and less trivial data points, as the fact that the group won a *Grammy Award* or plays *Pop music*. All these properties are freely available and can be easily gathered by using the SPARQL query language.[4]

* Corresponding author.
E-mail addresses: cataldomusto@gmail.com, cataldo.musto@uniba.it (C. Musto), pasquale.lops@uniba.it (P. Lops), marco.degemmis@uniba.it (M. de Gemmis), giovanni.semeraro@uniba.it (G. Semeraro).

[1] https://www.w3.org/RDF/.
[2] https://www.w3.org/OWL/.
[3] http://stats.lod2.eu/.
[4] https://www.w3.org/TR/rdf-sparql-query/.

http://dx.doi.org/10.1016/j.knosys.2017.08.015
0950-7051/

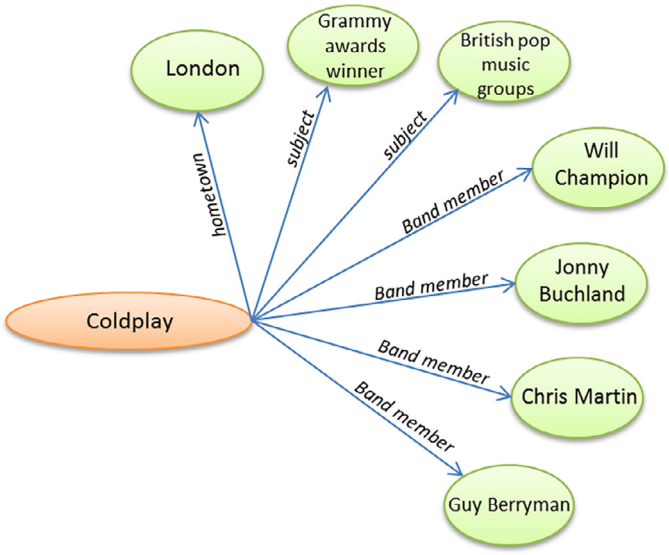


Fig. 1. A (tiny) portion of the properties, available in the LOD cloud, that describe the band *The Coldplay*.

This huge availability of semantics-aware machine-readable data attracted researchers and practitioners willing to investigate how such information can be exploited to develop new services and platforms, or to improve the effectiveness of existing algorithms. A very trending research line investigates the exploitation of these novel (semantic) data points in the area of Recommender Systems (RS) [24], since LOD can be effectively used to handle several problems RSs typically suffer from. *Content-based Recommender Systems* [15] for example, suffer from the well-known problem of *limited content analysis*, i.e. when limited or no features that describe the items to be recommended are available. The knowledge encoded in the LOD cloud can help to deal with this problem, since several features which are relevant for a recommendation task, as the *director* of a movie or the *genre* played by a band, can be gathered from the LOD cloud. This is a largely investigated research line, as we will show in the review of the literature in the area. Similarly, graph-based Recommender Systems can also benefit from such semantic data points. A recent survey on graph-based Recommender Systems is provided in [37], which also presents the data model they are based on. Basically, graph-based Recommender Systems model users and items as nodes in a graph and connect them according to the preferences of users on specific items. This model makes simpler the use of additional information related to users or items. In Fig. 2, the classic bipartite user-item graph representation modeling user-item preferences, as in classical *collaborative filtering algorithms*, can be easily extended by injecting in the graph the properties available in the LOD cloud that describe the items. Besides classical properties, items can be represented by very specific ones, which also allow to discover surprising connections. For example, as shown in Fig. 2 for the movie domain, both the movies *The Matrix* and *Moulin Rouge!* are *Australian films*, and these new information can in turn help to generate better (and maybe *unexpected*) recommendations.

According to these insights, it immediately emerges that RSs may tremendously benefit from the data points available in the LOD cloud. To this end, in this article we investigate the impact of such *exogenous knowledge* on the performance of a *hybrid* recommendation framework based on three classification techniques, Random Forests, Naïve Bayes and Logistic Regression. In this work we followed the hybridization strategy which is typically referred to as *feature combination* [11], i.e. items are represented in terms of different heterogeneous groups of features and are used as training examples to feed the classifiers. Such a model is then exploited to classify new and unseen items as *relevant* or *not relevant* for the target user.

The features we used can be roughly classified in three families: *basic features* (Section 3.1), *content-based features* (Section 3.2) and *topological features* (Section 3.3). *Basic features* include *popularity-based* features, as well as *collaborative features* built on the ground of the user preferences; *content-based features* include features extracted by processing the textual content describing the items, and *LOD-based properties* gathered from the LOD cloud, such as the *genre* of a movie or the *writer* of a book; *topological features* include *bipartite graph-based features* obtained by mining the bipartite graph connecting users to items they liked, and *tripartite graph-based features*which take into account the graph connecting users, items and properties gathered from the LOD cloud.

In the experimental session we assess the effectiveness of our framework by varying the sets of features used to represent items; results provide several interesting insights. Indeed, it emerges that the overall accuracy of the recommendation framework benefits from the introduction of *LOD-based* and *tripartite* graph-based features, and the proposed framework is able to overcome several state-of-the-art recommendation algorithms.

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

- we developed a hybrid recommendation framework based on classification techniques, and we designed families of features to feed the framework. Novel types of features extracted from the LOD cloud have been taken into account besides classical ones, they have been properly combined, and tested on three different datasets;
- we investigated to what extent the injection of knowledge coming from the LOD cloud influences the performance of a recommendation framework based on classification techniques. We have contributed to shed more light on the influence of different item representations based on the knowledge coming from the LOD cloud on the accuracy of recommendations. We tested representations based on properties extracted from the LOD cloud and on topological characteristics of the graph connecting users, items and properties;
- we identified the subsets of features that maximize a specific evaluation metric in our recommendation setting. We tested the ability of specific features and their combination to identify the most relevant items and to correctly rank them in the recommendation list;
- we validated our methodology by evaluating its effectiveness with respect to several state-of-the-art baselines. We compared our approach with widespread and best-performing recommendation algorithms, and with approaches introduced in our previous research as well.

The rest of the paper is organized as follows: Section 2 analyzes related literature. The description of the different features we adopted in our recommendation framework is provided in Section 3, while the details of the experimental evaluation we carried out are described in Section 4. Finally, Section 5 sketches conclusions and future work.

2. Related work

This work investigates the use of features gathered from the LOD cloud in a recommendation framework based on classification techniques. The idea of *casting* the recommendation task to a classification one dates back to the late 90s and is due to Pazzani et al. [49], who proposed a news recommender system that adopted a Naïve Bayes classifier to learn user profiles. After that, the use of such techniques has been largely investigated, especially

Please cite this article as: C. Musto et al., Semantics-aware Recommender Systems exploiting Linked Open Data and graph-based features, Knowledge-Based Systems (2017), http://dx.doi.org/10.1016/j.knosys.2017.08.015

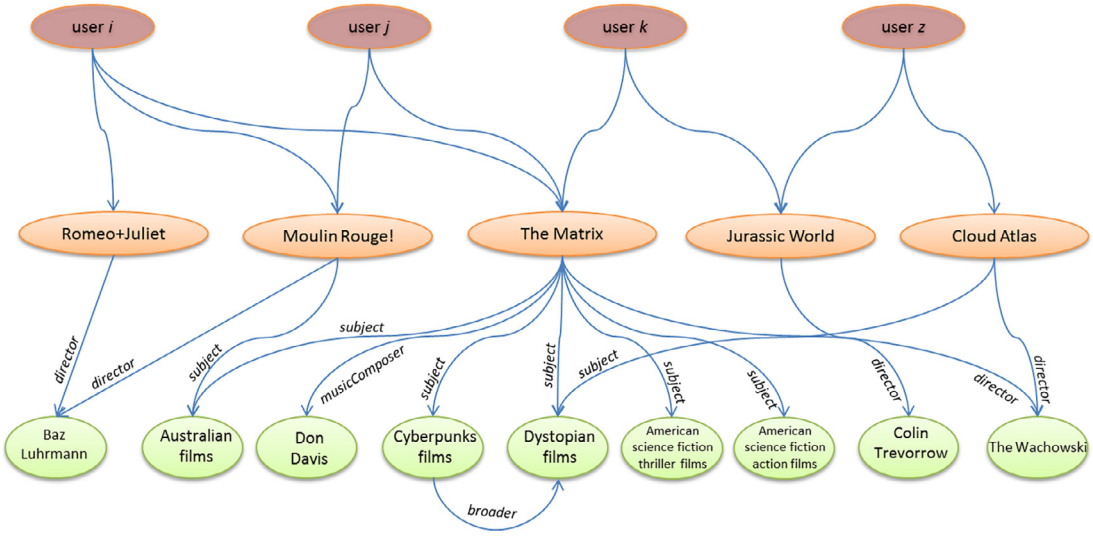


Fig. 2. A tiny portion of the graph connections between users, items and entities encoded in the LOD cloud.

for content-based recommendation algorithms [27]. Several work gave evidence of the good performance of Random Forests [63], Naïve Bayes [35] and Linear Classifiers [64] in a wide set of domains, ranging from movies [30] and cultural heritage [28,56], to the recommendation of interesting reviews [44].

By following the classification presented in [11], our framework falls in the category of *hybrid Recommender Systems*, since we performed a *feature combination* process that merges different types of features, ranging from collaborative and content-based ones to those gathered from the LOD cloud. The use of features directly extracted from the LOD cloud is one of the distinguishing aspects of this work. Research in this area takes its root in the field of ontology-based Recommender Systems, introduced by Middleton et al. [33]. However, in most of the current literature, properties gathered from `DBpedia` are only exploited to define new similarity measures, as by Passant [48] and more recently by Piao and Breslin [51]. Another similarity measure is proposed in [32], where the authors introduced the Partitioned Information Content (PIC), a measure inspired by the Information Theory and adapted to the scenario of Linked Open Data. In their work such semantic similarity measure is used as a backbone of a collaborative recommendation approach to identify items similar to those the target user already voted. Experiments demonstrated how such an approach overcomes all the baselines taken into account. The use of `DBpedia` for similarity calculation is also the core of the work presented by Musto et al. [41]: in that paper music preferences are extracted from Facebook and similarity measures are exploited to build personalized music playlists. In the current work, we do not define any similarity measure, rather we use the features gathered from the LOD cloud to build a more comprehensive model for representing user preferences [14].

Furthermore, Linked Open Data have been largely adopted as a mean to tackle the previously mentioned problem of *limited content analysis*. This is done in [9], where the authors present *TasteWeights*, a recommender system relying on music preferences extracted from Facebook. In that work `DBpedia` is exploited to gather one or more labels describing the genre played by each artist the user liked. Next, recommendations are generated by querying a `SPARQL` endpoint, looking for new artists playing (most of) the genres liked by the target user. Similarly, Baumann et al. [6] extracted features from Freebase[5] to describe artists. A similar approach is also proposed by Schmachtenberg et al. [55], who query `LinkedGeoData`[6] to collect features describing points of interests. Recently, the use of LOD-based data sources has been the core of the ESWC 2014 Recommender Systems Challenge[7] [16]: in that setting, the best-performing approach in the *top-N* recommendation task [4,5] was based on ensembles of several algorithms, such as Random Forests, Logistic Regression and Personalized PageRank [22], running on diverse sets of features gathered from the LOD cloud.

Differently from such literature, we tried to assess the effectiveness of LOD-based features on the overall accuracy of a recommender system. A similar attempt has been presented by Di Noia et al. [17], who performed a preliminary comparison of the accuracy of a movie recommender system, fed with different properties extracted from the LOD cloud. Similarly, Musto et al. [36] carried out an empirical analysis of the impact of LOD-based features on several recommendation techniques, based on PageRank and text classifiers. In both cases, LOD-based features included in the model are manually selected by exploiting trivial heuristics, such as their popularity. The problem of automatically selecting the best subset of LOD-based features has been recently tackled by Musto et al. [37,38], who presented an empirical comparison of several feature selection techniques to identify the best set of features among those available in the LOD cloud. Results showed a 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.

Most of the previous work used features directly extracted from the LOD cloud (as the genre of a movie or the writer of a book), while in the current work we design a more comprehensive *hybrid recommendation model* merging different features in a unique representation, following a similar attempt [45], where the authors presented preliminary results of a music recommender system merging textual data with graph-based and collaborative features.

Finally, the predictive power of graph-based features for recommendation tasks was investigated by several researchers, who showed the effectiveness of these features for the suggestion of the best citations of a research paper [58], for cross-domain recommendation based on features extracted from social networks

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

[6] http://linkedgeodata.org.

[7] http://challenges.2014.eswc-conferences.org/index.php/RecSys.

Please cite this article as: C. Musto et al., Semantics-aware Recommender Systems exploiting Linked Open Data and graph-based features, Knowledge-Based Systems (2017), http://dx.doi.org/10.1016/j.knosys.2017.08.015

Table 1
Family of features.

Family of features	Features
Basic	Popularity (P)
	Collaborative (C)
Content-based	Textual Content (T)
	LOD-based (L)
Topological	Bipartite graph-based (BP)
	Tripartite graph-based (TP)

[59] and for online dating systems [62]. These results are in line with the outcomes obtained by Tiroshi et al. [60], who showed that graph-based topology measures such as the number of neighbors, node centrality, PageRank and so on, can significantly improve the prediction accuracy of a recommender system. An important distinguishing aspect of our work is that in most of the literature the authors build their measures by only exploiting the *bipartite* user-item graph. In our case, we further extend this insight by also taking into account the *tripartite* user-item-properties graph, thus including also the information coming from the LOD cloud. Another approach that exploits features based on the tripartite graph-based representation is presented in [18,46]. In this case, the authors generate *path-based features* obtained by calculating the paths connecting users and items, and exploit them to feed a recommendation model based on a Learning to Rank framework.

In general, all the studies we mentioned confirmed the usefulness of injecting Linked Open Data into Recommender Systems. Indeed, regardless of the specific technique adopted to generate recommendations, the performance of LOD-based Recommender Systems tend to overcome that obtained by widespread recommendation techniques, as collaborative filtering or matrix factorization. This has been further confirmed by several studies performed in many different domains, as book recommendation [50], e-learning resources recommendation [19] and event recommendation [25].

To sum up, the goal of this work is to take the best out of the current research in the area of LOD-enabled Recommender Systems: our idea is to define a comprehensive *hybrid* representation model merging different families of features, i.e. *content-based, collaborative*, and *graph-based* ones, with the novel data points gathered from the LOD cloud. We carried out an extensive *feature engineering* process in order to assess their effectiveness in a recommendation task and to identify the subset of features which maximizes specific evaluation metrics. Through this work the aimed to make one step forward with respect our previous research in the area of Semantics-aware Recommender Systems [36–38]. Specifically, we enhanced the approach presented in [36] by introducing also graph-based features calculated on the ground of the tripartite graph representation connecting users, items and properties gathered from the LOD cloud, and we also extended the experimental evaluation by adding more evaluation metrics and more state-of-the-art datasets. Similarly, we also significantly extended the methodology presented in [37,38], since in the current work we defined a comprehensive hybrid item representation which includes several different families of features, while in our previous attempt the whole recommendation process was only based on PageRank scores.

3. Families of features

In this section we describe three families of features to represent items (Table 1). Features are extracted by analyzing the basic usage data, the content of items and the information coming from the LOD cloud. Different combinations of those features are adopted to train different classifiers aimed at the prediction of the most interesting items for a user.

We will use the movie *The Matrix* as a running example and we will show how each family of features contributes to the overall representation of the items.

Table 2
Toy example of the popularity features encoded for the movie *The Matrix*.

Popularity features	The Matrix
Number of ratings	2571
Number of positive ratings	2174
Ratio of positive ratings	0.845

Table 3
Toy example of a matrix modeling users' like and dislike. Each item is modeled by extracting the corresponding column vector.

	The Matrix	Cloud Atlas	Donnie Darko	...	Inception
U_1	1		1	...	1
U_2	0	1	0	...	1
U_3		0	0	...	1
...	...	...	...	...	...
U_n	1	1	1	...	0

3.1. Basic features

3.1.1. Popularity features (P)

This group of features include basic popularity-based information about the items, such as:

- **Number of ratings** - *overall* number of ratings received by an item
- **Number of positive ratings** - number of *positive* ratings received by an item
- **Ratio of positive ratings** - ratio between positive and overall number of ratings.

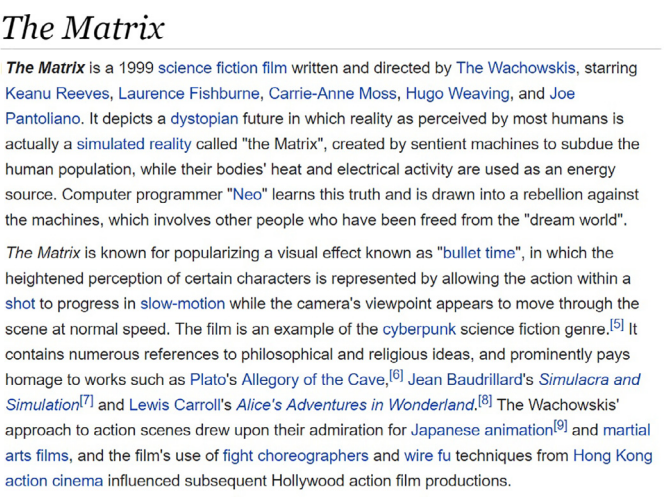
This tiny group of features may seem trivial, but it is typically very useful for a recommendation task, since it provides information about how popular is a certain item among the users and how positive is their general opinion about it. Moreover, as Cremonesi et al. already showed [13], on some datasets non-personalized algorithms based on simple popularity measures can obtain performance comparable to that of more sophisticated techniques, thus we decided to include also those features in a comprehensive *hybrid* representation of the items.

Table 2 shows an example of the popularity features encoded for the movie *The Matrix* in the MovieLens1M dataset [21]. The first two features are *not normalized*, while the third one ranges between 0 and 1.

3.1.2. Collaborative features (C)

This class of features models the information encoded in the *user-item matrix* typically exploited in collaborative filtering (CF) algorithms [23]. As shown in Table 3, rows of the matrix represent the preferences expressed by users – each *like* is encoded as 1, while each *dislike* is encoded as 0 – and columns represent the ratings received by each item. The empty cells, typically referred to as *missing values*, represent the items the user did not rate, yet.

Differently from classical CF algorithms, that leverage the *whole* user-item matrix to calculate the *neighbors* of the target user and to predict the items the user may be interested into, in our approach we are only interested in extracting the *column vector* modeling the ratings received by an item in order to include them in our *hybrid* item representation. Accordingly, the number of *collaborative features* we encoded for each item corresponds to the numbers of the *rows* in the matrix, i.e. to the number of *users* in the



The Matrix

The Matrix is a 1999 science fiction film written and directed by The Wachowskis, starring Keanu Reeves, Laurence Fishburne, Carrie-Anne Moss, Hugo Weaving, and Joe Pantoliano. It depicts a dystopian future in which reality as perceived by most humans is actually a simulated reality called "the Matrix", created by sentient machines to subdue the human population, while their bodies' heat and electrical activity are used as an energy source. Computer programmer "Neo" learns this truth and is drawn into a rebellion against the machines, which involves other people who have been freed from the "dream world".

The Matrix is known for popularizing a visual effect known as "bullet time", in which the heightened perception of certain characters is represented by allowing the action within a shot to progress in slow-motion while the camera's viewpoint appears to move through the scene at normal speed. The film is an example of the cyberpunk science fiction genre.[5] It contains numerous references to philosophical and religious ideas, and prominently pays homage to works such as Plato's Allegory of the Cave,[6] Jean Baudrillard's *Simulacra and Simulation*[7] and Lewis Carroll's *Alice's Adventures in Wonderland*.[8] The Wachowskis' approach to action scenes drew upon their admiration for Japanese animation[9] and martial arts films, and the film's use of fight choreographers and wire fu techniques from Hong Kong action cinema influenced subsequent Hollywood action film productions.

Fig. 3. Plot of the movie *The Matrix* extracted from Wikipedia.

Table 4
Partial representation of the vector modeling the content-based features extracted from the description of the movie *The Matrix*. Features are scored by using binary values.

Content-based features	The Matrix
The Matrix	1
1998	0
1999	1
science fiction	1
adventure	0
movie	1
film	0
The Wachowski	1
starring	1
...	...
Harrison Ford	0
Keanu Reeves	1
cyberpunk	1

dataset. The insight behind this choice is to emphasize the presence of users' behavioral patterns that emerge from the data, in order to obtain a similar representation for items liked by similar users. In our case, the collaborative features give the information that *The Matrix* is a movie similar to *Donnie Darko* since they obtained the same ratings from the community.

The choice of including this set of features in our hybrid representation is quite straightforward, since CF algorithms and matrix factorization techniques [26] tend to obtain very good performance especially when the *sparsity* of the original matrix is not high, thus we decided to encode also this information in our model. It is worth to note that we did not employ any particular technique to deal with missing values. Even if some work showed that processing missing values can lead to better performance [29], in the current work we preferred to exploit the whole matrix and to model missing values as *special* values without replacing them with synthetic scores.

3.2. Content-based features

3.2.1. Textual content features (T)

Textual content can be exploited to provide items with useful and descriptive features. As an example, the textual description of the movie *The Matrix* (Fig. 3) gathered from Wikipedia immediately shows that several distinctive characteristics of the movie can be extracted from such data.

However, *textual descriptions* of the items are typically full of *noisy* features, thus it is necessary to properly process such data by adopting Natural Language Processing (NLP) techniques [31] before including them in our item representations. By analyzing the content in Fig. 3, several features, as the *articles* or the *conjunctions*, are definitely not needed to represent items and should be removed. This step is referred to as *stopwords removal* and is performed by using standard lists of non-relevant terms which are filtered out from the content. Given that such lists are often language-dependent, this process is typically coupled with a *language detection* step that allows to identify the most correct list of stopwords to be used. Moreover, it is sometimes necessary to further process the content with more sophisticated techniques, in order to: (1) correctly identify the entities mentioned in the text, (2) detect more complex concepts as bigrams or trigrams (as *computer science* or *science fiction* in the plot of *The Matrix*), (3) reduce word inflections to their *word stem* through stemming algorithms [52], thus making the representation of words more uniform.

An extensive description of the algorithms exploited in an NLP pipeline is out of the scope of this paper. In this section we just want to emphasize that we provided our item representation with a set of descriptive features directly extracted from textual content.

To sum up, in our pipeline the original content was first tokenized, then stopwords were removed and entities occurring in the text were identified; the remaining tokens were stemmed. In this case, the amount of features added to the model corresponds to the size of the *vocabulary*, i.e. the number of different tokens occurring in the description of *all the items* in the dataset. Given such a representation, each feature can be scored by exploiting very simple weighting strategies, such as a boolean one which assigns 1 or 0 to each feature occurring or not in an item description, respectively, or a simple counting of occurrences of each feature in an item description, or a strategy based on the TF-IDF [2], which takes into account both the frequency of a feature (TF) in an item description and its rarity (IDF) in the whole set of item descriptions.

In Table 4 we provide a partial representation of the feature vector encoding content-based information for the movie *The Matrix*, extracted from the plot in Fig. 3. For the sake of readability, words were not stemmed and a binary score was adopted to indicate whether a certain feature occurs or not in the textual description. It is worth to note that in this representation *synonymous* terms (as *film* and *movie*) were treated as different tokens. In order to deal with this issue more sophisticated techniques for Word Sense Disambiguation (WSD) [42] are needed; the exploitation of a WSD based on Distributional Semantics [3] is one of the activities we already planned to perform in the next future.

3.2.2. LOD-based features (L)

All the features we described contribute to represent several facets of the items, since information coming from the community as well as textual characteristics extracted from the description are encoded in the same vector. However, one of the goal of this work is to further extend such a representation by introducing novel data points.

The LOD cloud is an important source to get more descriptive features to model the items. As previously stated, several information about the items are freely available in the LOD cloud in RDF format, and can be gathered by simply querying a SPARQL endpoint. In order to query a SPARQL endpoint we need two types of information: the URI of the resource to query and the name of the properties to gather.

Before obtaining the URI of the resource we are interested in, it is mandatory to carry out a preliminary step that is typically re-

Please cite this article as: C. Musto et al., Semantics-aware Recommender Systems exploiting Linked Open Data and graph-based features, Knowledge-Based Systems (2017), http://dx.doi.org/10.1016/j.knosys.2017.08.015

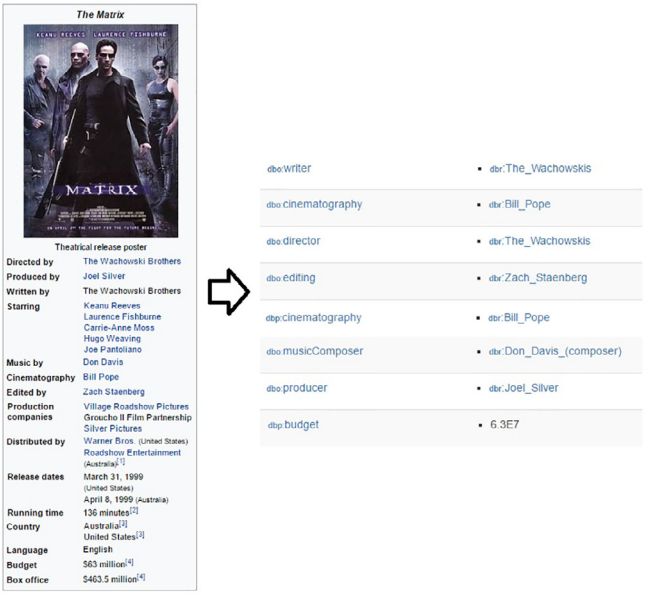


Fig. 4. Wikipedia infobox for the movie The Matrix and some of the corresponding properties encoded in RDF available in DBpedia.

ferred to as *mapping*. The goal of the mapping procedure is to identify, for each available item, the corresponding element in the LOD cloud the item refers to. As an example, we associate the movie *The Matrix* with its corresponding URI in the LOD cloud.[8] It is worth to emphasize that the mapping is a necessary and mandatory step to get an *entry point* to the LOD cloud. Without the mapping, it is not possible to have access to the LOD cloud and gather the information we need. Once this procedure is completed, it is possible to extract all the extra features describing our items.

In our case, we only focused on the data available in DBpedia, since it encodes in RDF format all the information contained in the Wikipedia infoboxes (Fig. 4), and we considered these data points as adequate to our purposes. As regards the properties to gather, we exploited the outcomes of our previous research [37,38] and we fed our model with the most relevant properties selected through feature selection algorithms.

To sum up, in order to gather LOD-based features we preliminarily carried out a mapping procedure to obtain the corresponding URI for each item in the dataset. Next, for each domain we defined a subset of relevant properties, and finally we used SPARQL to extract such data. Similarly to what we did for content-based features, we built a *vocabulary* of LOD-based properties and we used these features to represent each item. The score of each feature was set to 1 if the item is described through that RDF property, 0 otherwise. Table 5 reports some of the properties describing *The Matrix* gathered from the LOD cloud. Each feature is represented through the couple ⟨*property,value*⟩, since each entity can have different roles in the same movie or in distinct ones, as well.

A quick analysis of the features extracted from the LOD cloud shows a large overlap with those extracted from the simple textual description of items since some characteristics of the movie, as its director or starring, may appear both in DBpedia and in the plot of the movie. However, one of the advantages of merging structured LOD-based features with *unstructured* textual features is to couple the high precision of structured data (as the *genre* or the *director* of a movie) with the richness of the features extracted from the whole textual description. This aspect further confirms

Table 5
Partial representation of the vector modeling the LOD-based features extracted from DBpedia for the movie *The Matrix*.

Basic LOD-based features	The Matrix
⟨dbo:writer, dbr:The_Wachowski⟩	1
⟨dbo:writer, dbr:Quentin_Tarantino⟩	0
⟨dbo:director, dbr:The_Wachowski⟩	1
⟨dbo:director, dbr:Mel_Gibson⟩	0
⟨dbo:composer, dbr:Ennio_Morricone⟩	0
⟨dbo:composer, dbr:Don_Davis⟩	1
⟨dbo:editing, dbr:Zach_Staenberg⟩	1
⟨dct:subject, dbc:Artificial_uterus_in_fiction⟩	1
⟨dct:subject, dbc:Dystopian_films⟩	1
⟨dct:subject, dbc:American_Horror_movies⟩	0
...	...
⟨dbo:producer, dbr:Joel_Silver⟩	1

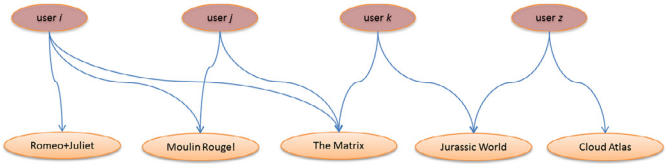


Fig. 5. A toy example of a bipartite graph, modeling users and items.

the insight behind our *hybrid data model*, since different knowledge sources are used to feed our representation with different facets of the items, giving rise to a comprehensive and likely more effective representation.

3.3. Topological features

3.3.1. Bipartite graph-based features (BP)

The whole set of preferences expressed by the users on the available items can be exploited to give rise to a *bipartite graph-based representation* of the available data. Specifically, we model each user and each item as *nodes*, and we connect them each time a user liked an item. It is worth to note that in this work we only model positive evidences (e.g., a user liked a specific item), even if in our previous research we showed that also negative evidences can improve the overall accuracy of the approach [40].

Formally, *bipartite graphs* are triplets $G_B = \langle U, I, E_B \rangle$, where U is the set of *top* nodes (e.g. the users), I is the set of *bottom* nodes (e.g. the items), and $E_B \subseteq U \times I$ the set of links between U and I, corresponding to a positive feedback expressed by a user on an item. Compared to standard graphs, nodes in a bipartite graph are separated in two disjoint sets, and links set a relation between a node in the first set and a node in the other set. An example of such a representation is provided in Fig. 5.

Given such a representation, we mined this graph to calculate some measures describing its *topological characteristics*, in order to encode this information in our item representation, as well. Since we are interested to enrich item representations, we define graph-based features specifically for item nodes and not for user nodes. We will denote by $N(i)$, $i \in I$ the *neighborhood* of a node item i, i.e. the set of users directly connected to i. Formally:

$N(i) : I \rightarrow 2^U \ni' N(i) = \{u_1, u_2, \ldots, u_k\}, (u_j, i) \in E_B, j = 1, 2, \ldots, k.$

We enriched item representations with the following five graph-based features calculated on the bipartite user-item graph:

Degree Centrality (Dc): it measures the importance of a node i through the number of nodes i is connected to. In the bipartite graph, *degree centrality* of an item corresponds to its *popularity*, i.e. the number of users who liked it. In Fig. 5, $Dc(MoulineRouge!) = 2$, while $Dc(TheMatrix) = 3$.

[8] http://dbpedia.org/resource/The_Matrix.

Average Neighbor Degree (ANd): it measures the average degree of nodes a node i is connected to. In the bipartite graph, *average neighbor degree* of an item corresponds to the average activity (number of liked items) of users who liked that item. In Fig. 5, $ANd(MoulineRouge!) = \frac{3+2}{2} = 2.5$, while $ANd(TheMatrix) = \frac{3+2+2}{3} = 2.33$.

PageRank score (PR): it is a widely-used recursive metric that quantifies the importance of nodes in a graph [47]. The core idea is to assign a score to any given node i which is derived from the links made to the node i from other nodes. Links from important nodes are worth more, and a node is important if it is pointed to by other important nodes. In the bipartite graph, *PageRank* of an item is computed through PageRank of users who liked that item.

Node Redundancy (Nr): it aims at capturing overlap in bipartite networks in a node-centered fashion. The redundancy coefficient of a node i (see Eq. 1) is the fraction of pairs of neighbors of i linked to another node than i.

$$Nr(i) = \frac{|\{\{u, w\} \subseteq N(i) : \exists i' \neq i, (u, i') \in E_B \wedge (w, i') \in E_B\}|}{\frac{|N(i)|(|N(i)|-1)}{2}} \quad (1)$$

In the bipartite graph, *node redundancy* of an item corresponds to the portion of pairs of users who liked that item and also liked another item. In Fig. 5, $Nr(TheMatrix) = \frac{1}{3}$.

Clustering Coefficient (Cc): it measures the degree to which nodes in a graph tend to cluster together. More specifically, clustering of a node i is given by the proportion of links between the nodes within its neighborhood divided by the number of links that could possibly exist between them. Let e_i be the number of edges between its neighbors (number of connections between i's neighbors), the clustering coefficient is then defined as:

$$Cc(i) = \frac{e_i}{\frac{|N(i)|(|N(i)|-1)}{2}} \quad (2)$$

This measure is meaningful only for $|N(i)| > 1$. If $|N(i)| = 1$ then we consider Cc(v) = 0. As shown in Fig. 5, in the bipartite graph there are no connections between users, hence the clustering coefficient is always equal to zero.

Please refer to West et al. [61] for a more comprehensive overview of these measures. As mentioned in the related work, most of these measures already proved their effectiveness for recommendation tasks.

3.3.2. Tripartite graph-based features (TP)

The *bipartite* graph-based representation modeling only the connections between users and items can be extended by taking into account the properties available in the LOD cloud. Indeed, given that each RDF triple can be modeled in a graph-based fashion as well (items can be connected to their properties and edges can be labeled with the name of the property), we can inject these novel nodes and edges in the bipartite graph in order to give rise to a *tripartite graph* that also models the properties gathered from the LOD cloud. An example of such a representation is provided in Fig. 2.

Formally, we define an extended graph $G_T = \langle U, I, P, E_T \rangle$, where U is the set of users, I is the set of items, P is the new set of nodes (e.g. *Wachowski brothers* or *Keanu Reeves*) connected to the items through the properties of the LOD cloud, and E_T is enriched with the set of the new connections resulting from the properties encoded in the LOD cloud. It is worth to note that we only include nodes and properties which are *directly* connected to the items to be recommended. This choice is due to the results already presented in [36], where it is shown that the introduction of non-direct relationships does not produce a significant improvement in the precision of the recommendation process.

In our approach the graph-based measures mentioned in the previous section are also calculated on the *tripartite graph*, and are included in the hybrid model. Given that these features are calculated on the ground of the topology of the graph, the huge amount of new nodes and edges resulting from the injection of the knowledge gathered from the LOD cloud tremendously changes the structure of the representation and the values of the measures as well.

In the tripartite graph the *degree centrality* of an item corresponds to the number of users and properties connected to that item, i.e. its popularity, increased by the number of properties used to represent that item. This score is high for popular items having a high number of properties. In Fig. 2, $Dc(MoulineRouge!) = 4$, while $Dc(TheMatrix) = 10$. The *average neighbor degree* of an item incorporates the popularity of its properties (i.e. the number of items having that property in the graph), besides the average activity of users who liked that item. In Fig. 2, $ANd(MoulineRouge!) = \frac{3+2+2+2}{4}$. The *PageRank* score in the tripartite graph is affected by the new nodes and edges introduced by the properties from the LOD cloud as well. The *node redundancy* in the tripartite graph also incorporates pairs of properties other items are associated with as well. Differently from the case of the bipartite graph, the *clustering coefficient* can be computed for item nodes in the *tripartite* graph by taking into account the links occurring between the properties coming from the LOD cloud.

3.4. Recommendation framework

We formulate the problem of computing *top-N* recommendations as a *classification task*: given a target user u, the recommendation process is formulated as a binary classification task, in which each item has to be classified as interesting or not with respect to the preferences of the target user u.

User-item pairs $(u, i) \in U \times I$ are labeled with relevance judgments that indicate the *degree of interest* of the user u towards the item i. Each item, represented by a set of features and coupled with its relevance judgment, is treated as a single datapoint, and a set of datapoints can be used for training purposes. This allows the recommendation framework to learn a function to *predict the relevance judgment* of new unknown items.

More formally, let $x_i = \phi(i)$, where ϕ is a feature extractor and x_i is a m-dimensional vector. Let $TR(u) = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a set of item representations and their associated relevance ratings $y_i \in Y$ given by user u. In our recommendation scenario, relevance is 1 for the items interesting to the user u, i.e. *positive examples*, and 0 for the other items, i.e. *negative examples*. $TR(u)$ is used to train a classification model for the user u.

In the experimental session the overall effectiveness of the recommendation framework was evaluated on different sets of features, i.e. different implementations of ϕ (Section 3) and using three different classification algorithms, namely *Naïve Bayes, Logistic Regression*, and *Random Forests*.

Naïve Bayes (NB) algorithm [34] is adopted as the simplest classification algorithm which often does a great job in practice. The main disadvantage is that it cannot learn interactions between features, e.g., it cannot learn that, although you love movies with Brad Pitt and Tom Cruise, you hate movies where they are together. *Logistic Regression* (LR) [12] is a pretty robust to noise and well-behaved classification algorithm. It is efficient and a good advantage is that the output can be interpreted as a probability value, which makes it suitable for ranking in addition to classification. The problem is that it can hardly handle categorical (binary) features. Finally, *Random Forests* (RF) algorithm [10], belonging to the family of tree ensembles, has different advantages over LR. RF combines several tree predictors built using different samples of the training data (extracted with replacement from the whole training set) and random subsets of the data features. The class of an item is determined by the *majority voting* of the classes returned by the

Table 6
Statistics of the datasets.

	MovieLens	DBbook	Last.fm
Users	6040	6181	1892
Items	3883	6733	17,632
Ratings	1,000,209	72,372	92,834
Sparsity	96.42%	99.85%	99.80%
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

individual trees. The use of different samples of the data from the same distribution and of different sets of features for learning the individual decision trees prevents the overfitting problem, and allows the algorithm to handle efficiently high dimensional spaces as well as a large number of training examples. Differently from LR, Random Forests are able to handle categorical features.

4. Experimental evaluation

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

1. Which set of *single* features or which *combination* can provide the best predictive accuracy (Experiment 1)?
2. Are topological graph-based features able to improve the overall performance of the recommendation framework (Experiment 2)?
3. How does our best-performing configuration compare with respect to *state-of-the-art techniques* (Experiment 3)?

4.1. 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 Systems challenge and focuses on book recommendation, while the latter is a music recommendation dataset relying on `Last.fm`'s users listening habits. Some statistics about the datasets are provided in Table 6.

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 are provided with a significant number of ratings (165.59 per user and 269.88 per item, on average), and this makes easier the neighborhood computation and similarity calculations. On the other side, `DBbook` and `Last.fm` are more sparse. `DBbook` 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 per item (5.26, with a mode equal to 1). Due to the sparsity issue, it is likely that both these datasets will benefit from 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.

Experimental protocol. Experiments were performed by adopting different protocols: we used a 80%-20% training-test split for `MovieLens 1M` and for `Last.fm`. The split was built on a *per-user* basis, in order to maintain enough ratings for each user in the training set. For `DBbook` we used the training-test split provided in literature that was exploited in the previously mentioned ESWC 2014 Recommender Systems challenge.

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 [46]: 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 classification algorithms we used the implementations of *Logistic Regression, Random Forests* and *Naïve Bayes* available in the Weka Toolkit.[11] All the algorithms were launched with the default parameters.

Popularity features were extracted by simply processing the original data and by counting the ratings received by each item. As regards *collaborative features*, we replaced missing values with a special character and we used a binary representation to encode positive and negative ratings. Next, to generate *content-based features* we used the methods implemented in the Apache Lucene[12] library for tokenization, language detection and stopwords removal. Textual descriptions were all gathered from the Wikipedia pages of the items. In order to find the Wikipedia page each item refers to, we used some *mappings* already available in literature.[13] Finally, tokens were stemmed by exploiting the Snowball library.[14] Each feature was scored using TF-IDF.

In order to enrich the representation with the features extracted from the LOD cloud each item in the dataset was mapped to a `DBpedia` entry. Specifically, 3301 `MovieLens 1M` entries were successfully mapped (85% of the items), while all the 6733 items 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 ESWC 2014 RecSys challenge. Finally, 9490 `Last.fm`'s artists (53.8%) were mapped to `DBpedia`. In this case we used the mapping made available in [46]. The items for which a `DBpedia` entry was not found were represented by using only the basic groups of features.

Finally, *graph-based features* were calculated by exploiting the Jung framework,[15] a Java library to manage graph-based data. For each item node we calculated *Degree Centrality, Average Neighbor Degree, PageRank* score, *Node Redundancy* and *Clustering Coefficient* for both bipartite and tripartite graph-based representation.

To sum up, in Table 7 we recap the number of features encoded by each group. *Popularity* and *graph-based features* do not

Table 7
Comparison among the number of features encoded by each group, split by dataset.

#features	ML1M	DBbooks	Last.fm
Popularity	3	3	3
Collaborative	6040	6181	1892
Content-based	53,332	100,935	65,130
LOD-based	19,991	17,589	38,656
Bipartite Graph-based	4	4	4
Tripartite Graph-based	5	5	5

[9] http://grouplens.org/datasets/movielens/1m/.
[10] http://grouplens.org/datasets/hetrec-2011/.
[11] http://www.cs.waikato.ac.nz/ml/weka/.
[12] https://lucene.apache.org/.
[13] https://tinyurl.com/datasets-lod-recsys.
[14] http://snowball.tartarus.org/.
[15] http://jung.sourceforge.net/.

Table 8
Results of Experiment 1 on MovieLens data. Best-performing configuration for each algorithm is highlighted in **bold**. The overall best configuration is emphasized with a **grey backround**.

MovieLens	F1@5			nDCG@5		
	Random Forests	Naïve Bayes	Logist. Regres.	Random Forests	Naïve Bayes	Logist. Regres.
Popularity (P)	0.5338	0.5458	0.5526	0.7950	0.8160	0.8323
Collaborative (C)	0.5618	**0.5486**	0.5560	0.8541	**0.8170**	0.8399
Textual (T)	0.4913	0.4913	0.5135	0.7177	0.7140	0.7688
Lod-based (L)	0.5065	0.5094	0.5106	0.7579	0.7590	0.7644
P+C	0.5635	0.5483	0.5631	0.8551	0.8088	0.8528
P+T	0.5051	0.4965	0.5465	0.7445	0.7244	0.8247
P+L	0.5312	0.5320	0.5520	0.7908	0.7971	0.8349
C+T	0.5187	0.5180	0.5528	0.7761	0.7578	0.8390
C+L	0.5609	0.5450	0.5573	0.8537	0.8043	0.8442
T+L	0.4943	0.4932	0.5147	0.7254	0.7169	0.7701
P+C+T	0.5246	0.5189	0.5616	0.7862	0.7592	0.8508
P+T+L	0.5079	0.4974	0.5484	0.7484	0.7251	0.8286
P+C+L	**0.5642**	0.5451	**0.5639**	**0.8578**	0.8040	**0.8538**
C+T+L	0.5188	0.5169	0.5514	0.7775	0.7558	0.8375
P+C+T+L	0.5246	0.5174	0.5587	0.7870	0.7567	0.8471

differ for the different datasets since they are computed according to the available data (for the bipartite graph there are 4 features since the clustering coefficient is always equal to 0), while the number of *collaborative features* corresponds to the number of users in each dataset. Finally, the *vocabulary* used to encode textual and LOD-based data makes the number of content-based features much higher if compared to the other groups. Clearly, when the recommendation framework is fed with different groups of features (e.g. collaborative and textual, at the same time), the features encoded in each training example are the results of the merge of the features encoded by each single group.

The performance for each configuration of our recommendation framework was evaluated in terms of *F1-measure* and *normalized Discounted Cumulative Gain (nDCG)*, computed by averaging results obtained by each user in the dataset. Metrics were calculated through the Rival toolkit[16] [54] and were obtained by following the `unrated-test-items` evaluation methodology [57], that is to say, we ranked only the items available in test set, by ignoring all the others.

Statistical significance was assessed by exploiting Wilcoxon and Friedman tests, chosen after running the Shapiro-Wilk test,[17] which revealed the non-normal distribution of the data.

4.2. Discussion of the results

Experiment 1. To gain insights about the predictive power of different feature families, we first separately assess the performance of each set of features, i.e. popularity, collaborative, content- and LOD-based (see Table 1); at the next step, we assess the performance of all their possible combinations. Results are reported in Tables 8–10, for MovieLens, DBbook and Last.fm, respectively.

Single sets of features used in isolation work better for MovieLens and DBbook, than for Last.fm. On MovieLens, the collaborative features outperform the others, regardless the classification algorithm adopted and the evaluation measure taken into account. This is in line with what we expected, since MovieLens is the least sparse dataset (96.42%), with each item having more than 269 ratings, on average. Collaborative features show the best performance for DBbook when the NB classifier is adopted, while RF and LR have better performance when fed with popularity features. This supports the finding that non-personalized features have an important role when the datasets are very sparse and few user preferences are available. This is the case of DBbook, which also results unbalanced in terms of positive and negative ratings. Differently from the other two datasets, collaborative features show the worst performance on Last.fm, probably due to the high sparsity and to the lowest number of available ratings per item (5.26). Finally, for textual and LOD-based features a clear pattern does not emerge: on MovieLens their performance is worse than popularity and collaborative features, on Last.fm they perform better than collaborative features, while on DBbook their performance are satisfactory with RF and LR, but not with NB.

When combining features in pairs, interesting patterns emerge. First of all, even though we combine features which perform best individually, their combination could not lead to a significant improvement of the overall performance. On MovieLens the best combination is obtained by merging popularity and collaborative features, and this might be due to the characteristics of the dataset, where the low sparsity and the high number of ratings per items and users are sufficient to obtain accurate recommendations, without the need of leveraging any kind of content information. Same result for DBbook, when NB and LR are adopted, while RF works better with the combination of popularity and LOD-based features. On Last.fm, a clear pattern does not emerge, even though the best configurations include popularity features most of the times. It is worth to note that combining textual and LOD-based properties improves the performance only slightly, as the signal that they bring is very overlapping. Given that obtaining textual features requires a specific NLP pipeline, which is also language-dependent, it appears that the use of LOD-based features is a viable alternative. Indeed, among the different combinations of features in triples, the one based on popularity, collaborative and LOD-based features leads to good performance for MovieLens and DBbook. Moreover, the overall best configuration for each dataset always involves LOD-based features, besides those based on popularity.

To sum up, the outcomes of this experiment confirmed the goodness of *collaborative* features, and showed that *non-personalized* ones are very important to improve the effectiveness of single features by just introducing simple popularity-based data points. A connection between the effectiveness of the features and the sparsity of the datasets partially emerged. When the sparsity is low, collaborative features have to be used. On the other side, as

[16] http://rival.recommenders.net/.
[17] http://en.wikipedia.org/wiki/Shapiro-Wilk_test.

Table 9
Results of Experiment 1 on DBbook data. Best-performing configuration for each algorithm is highlighted in **bold**. The overall best configuration is emphasized with a **grey backround**.

	F1@5			nDCG@5		
DBbook	Random Forests	Naïve Bayes	Logist. Regres.	Random Forests	Naïve Bayes	Logist. Regres.
Popularity (P)	0.5610	0.5576	0.5615	0.7267	0.7227	0.7291
Collaborative (C)	0.5421	0.5610	0.5482	0.7043	0.7290	0.7121
Textual (T)	0.5532	0.5465	0.5581	0.7192	0.7040	0.7260
Lod-based (L)	0.5544	0.5553	0.5525	0.7236	0.7191	0.7162
P+C	0.5627	**0.5615**	**0.5654**	0.7299	**0.7298**	**0.7366**
P+T	0.5567	0.5467	0.5651	0.7235	0.7048	0.7359
P+L	**0.5659**	0.5577	0.5619	**0.7379**	0.7246	0.7302
C+T	0.5549	0.5464	0.5634	0.7191	0.7042	0.7346
C+L	0.5560	0.5564	0.5518	0.7248	0.7201	0.7162
T+L	0.5551	0.5494	0.5565	0.7224	0.7093	0.7209
P+C+T	0.5583	0.5468	0.5623	0.7230	0.7049	0.7345
P+T+L	0.5569	0.5497	0.5627	0.7243	0.7101	0.7309
P+C+L	0.5630	0.5580	0.5632	0.7342	0.7249	0.7312
C+T+L	0.5553	0.5491	0.5566	0.7202	0.7092	0.7209
P+C+T+L	0.5560	0.5497	0.5621	0.7213	0.7102	0.7310

Table 10
Results of Experiment 1 on Last.fm data. Best-performing configuration for each algorithm is highlighted in **bold**. The overall best configuration is emphasized with a **grey backround**.

	F1@5			nDCG@5		
Last.fm	Random Forests	Naïve Bayes	Logist. Regres.	Random Forests	Naïve Bayes	Logist. Regres.
Popular (P)	0.4278	0.4347	0.4276	0.5667	0.5727	0.5698
Collaborative (C)	0.3837	0.4097	0.4145	0.5130	0.5464	0.5513
Textual (T)	0.4286	0.4285	0.4315	0.5569	0.5588	0.5702
Lod-based (L)	0.4264	0.4295	0.4227	0.5600	0.5587	0.5556
P+C	0.4362	0.4342	0.4398	0.5756	0.5738	0.5858
P+T	0.4408	0.4348	0.4372	0.5795	0.5692	0.5780
P+L	0.4479	0.4417	0.4353	0.5849	0.5721	0.5710
C+T	0.4339	0.4340	0.4402	0.5659	0.5662	0.5796
C+L	0.4316	0.4392	0.4357	0.5701	0.5678	0.5694
T+L	0.4416	0.4337	0.4390	0.5780	0.5560	0.5731
P+C+T	0.4366	0.4336	**0.4438**	0.5712	0.5656	**0.5866**
P+T+L	**0.4527**	**0.4469**	0.4396	**0.5873**	0.5708	0.5778
P+C+L	0.4463	0.4436	0.4377	0.5836	**0.5741**	0.5745
C+T+L	0.4453	0.4336	0.4428	0.5825	0.5554	0.5799
P+C+T+L	0.4491	0.4376	0.4426	0.5857	0.5612	0.5818

the sparsity increases, it is necessary to look for different sources of information. Moreover, when the *textual content* is not noisy (as for Last.fm), it represents a good alternative, otherwise the simple *popularity* features do their job, as shown on DBbook.

The adoption of *LOD-based* features has to be carefully evaluated, since these data showed to be particularly helpful to filter out some noise from *textual* representations, but often they do not improve the overall performance of the framework. This is evident for datasets with low sparsity as MovieLens, which makes superfluous most of the features with the exception of collaborative ones.

Moreover, we also noted a connection between the classification algorithm and the effectiveness of LOD-aware representations, since the algorithm able to take (most) advantage of the exogenous information coming from the LOD cloud is RF. Indeed, the best overall configuration for the three datasets is obtained using RF, and all those configurations include popularity and LOD-based features. It is likely that the ability of RF to automatically select the most relevant properties is helpful to identify the most informative features. Last but not the least, all the observations discussed so far are valid for both F1 and nDCG measures. Indeed, the trend observed in the results is the same for both the evaluation measures, i.e. the recommendation framework developed using the three classifiers on different sets of features is able to identify the most relevant items, and to correctly rank them in the final recommendation list.

Experiment 2. In Experiment 2 we evaluated the impact of topological features on our recommendation framework. For each dataset we assessed the performance of the *bipartite* and *tripartite* graph-based features in isolation, and properly combined with three baselines. The first baseline (B_{bas}) corresponds to the best result obtained in the previous experiment using the *basic* feature family (see Table 1), i.e. the best result obtained using *popularity* and *collaborative* features, in isolation or combined. The second baseline (B_{cont}) is similar to the previous one, but related to the *content-based* feature family, i.e. the best result obtained using *textual content* and *LOD-based* features, in isolation or combined. The third baseline (B_{All}) corresponds to the best combination of *basic* and *content-based* feature families, i.e. the best result obtained using *popularity, collaborative, textual content* and *LOD-based* features, in isolation or combined. Results are shown in Tables 11, 12, and 13, respectively.

Comparing bipartite and tripartite features used in isolation, it is worth to notice that they have the very same performance on

Table 11
Results of Experiment 2 on MovieLens data. Best-performing configuration for each algorithm is highlighted in **bold**. The overall best configuration is emphasized with a **grey backround**.

MovieLens	F1@5			nDCG@5		
	Random Forests	Naïve Bayes	Logist. Regres.	Random Forests	Naïve Bayes	Logist. Regres.
BP	0.5057	0.5230	0.5120	0.7433	0.7799	0.7594
TP	0.5054	0.5225	0.5112	0.7453	0.7793	0.7583
BEST BASIC(B_{BAS})	0.5635	**0.5486**	0.5631	0.8551	**0.8170**	0.8528
B_{bas}+BP	0.5620	0.5482	0.5128	0.8538	0.8086	0.7609
B_{bas}+TP	0.5621	0.5483	0.5118	0.8539	0.8088	0.7583
B_{bas}+BP+TP	0.5607	0.5480	0.5141	0.8529	0.8093	0.7667
BEST CONTENT(B_{CONT})	0.5065	0.5094	0.5147	0.7579	0.7590	0.7701
B_{cont}+BP	0.5135	0.5276	0.5181	0.7602	0.7894	0.7706
B_{cont}+TP	0.5150	0.5266	0.5170	0.7650	0.7877	0.7687
B_{cont}+BP+TP	0.5198	0.5299	0.5148	0.7738	0.7903	0.7670
BEST OVERALL(B_{ALL})	0.5642	**0.5486**	**0.5639**	0.8578	**0.8170**	**0.8538**
B_{All}+BP	0.5635	0.5482	0.5188	0.8582	0.8086	0.7720
B_{All}+TP	**0.5678**	0.5483	0.5177	**0.8602**	0.8088	0.7701
B_{All}+BP+TP	0.5622	0.5480	0.5146	0.8547	0.8093	0.7675

Table 12
Results of Experiment 2 on DBbook data. Best-performing configuration for each algorithm is highlighted in **bold**. The overall best configuration is emphasized with a **grey backround**.

DBbook	F1@5			nDCG@5		
	Random Forests	Naïve Bayes	Logist. Regres.	Random Forests	Naïve Bayes	Logist. Regres.
BP	0.5503	0.5493	0.5486	0.7140	0.7130	0.7116
TP	0.5476	0.5461	0.5411	0.7078	0.7064	0.7003
BEST BASIC(B_{BAS})	0.5627	**0.5615**	**0.5654**	0.7299	**0.7298**	**0.7366**
B_{bas}+BP	0.5585	0.5589	0.5594	0.7246	0.7252	0.7243
B_{bas}+TP	0.5607	0.5542	0.5411	0.7259	0.7170	0.7003
B_{bas}+BP+TP	0.5601	0.5566	0.5416	0.7265	0.7193	0.7022
BEST CONTENT(B_{CONT})	0.5551	0.5553	0.5581	0.7236	0.7191	0.7260
B_{cont}+BP	0.5548	0.5577	0.5551	0.7331	0.7249	0.7239
B_{cont}+TP	0.5545	0.5574	0.5411	0.7272	0.7200	0.7003
B_{cont}+BP+TP	0.5565	0.5588	0.5416	0.7328	0.7242	0.7020
BEST OVERALL(B_{ALL})	0.5659	**0.5615**	**0.5654**	0.7379	**0.7298**	**0.7366**
B_{All}+BP	0.5642	0.5589	0.5594	0.7361	0.7252	0.7243
B_{All}+TP	0.5667	0.5542	0.5411	0.7381	0.7170	0.7003
B_{All}+BP+TP	**0.5697**	0.5566	0.5416	**0.7389**	0.7193	0.7022

MovieLens and DBbook, while tripartite features outperform bipartite ones for Last.fm. This is probably due to the very low number of ratings per item, and to the need of further extending the connections between users and items with the properties in the LOD cloud. An interesting result is that bipartite and tripartite features have performance comparable to that of textual or LOD-based features. Given that the process that computes textual and LOD-based features requires a quite complex NLP pipeline or a mapping of items to DBpedia, topological features represent a more lightweight (they are very few) and therefore a more viable alternative. Moreover, tripartite features also represent a very useful alternative to collaborative features for very sparse datasets, such as DBbook and Last.fm.

On MovieLens, the combination of bipartite and tripartite features with the baselines must be carefully evaluated, since they do not always improve the overall performance. Combining topological features with basic ones leads to quite the same performance when using RF and NB, and to a considerable decrease of performance with LR. A different outcome is obtained with the combination of content feature family. In that case, topological features give a contribution to the performance of RF and NB, even though it is fairly small. Overall, the best configuration for MovieLens is obtained by feeding a RF classifier with a combination of all the feature families and the tripartite ones.

Similar outcomes emerged for DBbook. LR never benefits from the introduction of topological features since the best overall configuration corresponds to the use of basic feature family. Conversely, when using RF and NB, topological features lead to an improvement of the performance, with the best overall configuration obtained by feeding again the RF classifier with all the feature families, i.e. basic, content and topological. The best performance of the RF classifier is expected, due to its ability to deal with very high dimensional spaces for representing training examples.

Finally, very similar outcomes are also confirmed for Last.fm, where LR again does not always benefit from the introduction of topological features. For RF and NB, even though the improvements are pretty small, topological features could contribute to improve the accuracy and ranking of the recommendation lists. Similarly to DBbook, the best overall configuration is obtained by RF and the combination of all the feature families.

Experiment 3. In the last experiment we compared the effectiveness of our hybrid recommendation methodology with several state of the art recommendation algorithms, i.e. User-to-User (U2U-KNN) and Item-to-Item Collaborative Filtering (I2I-KNN), the Bayesian Personalized Ranking (BPRMF) which uses Matrix Factorization as the learning model with Bayesian Personalized Ranking (BPR) optimization criterion [53], and an implementation of PageRank with Priors [22].

Please cite this article as: C. Musto et al., Semantics-aware Recommender Systems exploiting Linked Open Data and graph-based features, Knowledge-Based Systems (2017), http://dx.doi.org/10.1016/j.knosys.2017.08.015

Table 13
Results of Experiment 2 on `Last.fm` data. Best-performing configuration for each algorithm is highlighted in **bold**. The overall best configuration is emphasized with a **grey background**.

	F1@5			nDCG@5		
`Last.fm`	Random Forests	Naïve Bayes	Logist. Regres.	Random Forests	Naïve Bayes	Logist. Regres.
BP	0.4092	0.4019	0.3918	0.5428	0.5412	0.5111
TP	0.4261	0.4325	0.4240	0.5606	0.5582	0.5495
Best Basic(B_{BAS})	0.4362	0.4347	0.4398	0.5756	0.5738	0.5858
B_{bas}+BP	0.4354	0.4224	0.3995	0.5750	0.5641	0.5240
B_{bas}+TP	0.4408	0.4482	0.4242	0.5767	**0.5859**	0.5496
B_{bas}+BP+TP	0.4442	0.4357	0.4222	0.5821	0.5759	0.5477
Best Content(B_{CONT})	0.4416	0.4337	0.4390	0.5780	0.5588	0.5731
B_{cont}+BP	0.4495	0.4333	**0.4477**	0.5861	0.5656	0.5834
B_{cont}+TP	0.4473	0.4358	0.4354	0.5788	0.5622	0.5649
B_{cont}+BP+TP	0.4477	0.4450	0.4348	0.5805	0.5667	0.5630
Best Overall(B_{ALL})	0.4527	0.4469	0.4438	0.5873	0.5741	**0.5866**
B_{All}+BP	0.4549	0.4471	0.4396	0.5895	0.5742	0.5594
B_{All}+TP	0.4561	**0.4484**	0.4389	0.5899	0.5758	0.5634
B_{All}+BP+TP	**0.4577**	0.4477	0.4364	**0.5910**	0.5751	0.5789

Table 14
Results of Experiment 3. The best-performing algorithm is highlighted with a **grey backround**. The Lod-RecSys configuration refers to the overall best-performing configuration emerged for each dataset from the previous experiments.

	F1@5			nDCG@5		
Algorithm	`MovieLens`	`DBbook`	`Last.fm`	`MovieLens`	`DBbook`	`Last.fm`
Lod-RecSys	**0.5678**	**0.5697**	0.4577	**0.8602**	**0.7389**	0.5910
U2u-Knn	0.4270	0.5193	0.4510	0.7163	0.6486	0.7801
I2i-Knn	0.4320	0.5111	0.4419	0.7253	0.6323	0.7689
Bprmf	0.5218	0.5290	0.4506	0.7815	0.6575	0.7730
Bprmf+Lod	0.5215	0.5304	0.4516	0.7797	0.6629	0.7762
Ppr	0.5397	0.5502	0.4635	0.8100	0.6874	0.8078
Ppr+Lod	0.5400	0.5540	**0.4768**	0.8109	0.6935	**0.8189**

Moreover, we also compared our methodology to other LOD-aware recommendation techniques. Specifically, we used the features gathered from the LOD cloud as side information for BPRMF, as proposed by Gantner et al. [20]; next, we also we also evaluated an implementation of PageRank with Priors (PPR) extended with LOD-based features, as we already investigated in our previous research [37,38].

For U2U-KNN and I2I-KNN, experiments were carried out by setting the neighborhood size to 50, 80 and 100 and by using cosine similarity as measure for computing the neighborhood, while BPRMF was run by setting the number of latent factors 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-KNN and I2I-KNN, 100 factors for BPRMF, 50 factors for BPRMF with side information). For U2U-KNN, I2I-KNN and BPRMF we exploited the implementations available in MyMediaLite,[18] while the methods implemented in the Jung framework[19] were used to run PPR. As in [37], PPR 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), and the damping factor was set to 0.85.

As shown in Table 14, our hybrid recommendation framework always significantly overcomes all the baselines on `MovieLens` and `DBbook`, for both F1@5 and nDCG@5. It is worth to note that our approach obtains better results when compared to both classic baselines as well as to other LOD-aware techniques as BPRMF+LOD and PPR+LOD.

A different behavior can be noted on `Last.fm`, since our approach overcomes all the baselines with the exception of PPR and PPR+LOD. This outcome is worth to be further analyzed, even though it can be directly related to the particular topology of the user-item patterns available in `Last.fm`. Indeed, as shown in the previous experiment, `Last.fm` was the dataset which benefited the most from the *topological* features included in the model, since most of the tested hybrid representations got an improvement both in terms of F1@5 and nDCG@5 after the injection of those features. Thus, it is likely that a *pure* graph-based representation represents the best way to model such data, and this can justify the very high performance of PPR algorithm on `Last.fm`.

However, regardless this aspect, also this last experiment confirmed the insight behind our research. Indeed, in all the other experimental settings our hybrid recommendation framework well performed against several state-of-the-art baselines, by showing the usefulness of introducing both LOD-based and topological graph-based features in a hybrid item representation.

5. Conclusions and future work

In this article we presented a hybrid recommendation framework based on the combination of different groups of features, as *popularity-based, collaborative, content-based* and *graph-based* ones. Such groups of features were combined and used to feed classification algorithms as RF, NB and LR. Next, we extended our item representation by introducing two extra groups of features gathered from the LOD cloud, as *LOD-based features* (e.g. *genre* of a movie) and *graph-based ones* built on the ground of the topolog-

[18] http://www.mymedialite.net/.
[19] http://jung.sourceforge.net/.

ical characteristics of both the bipartite and tripartite graph-based representations connecting users, items and properties.

Given this item representation, we evaluated our framework through an extensive experimental evaluation, and several interesting outcomes emerged: first, RF was the classification algorithm able to take the best out of our hybrid data representation. Indeed, for all the datasets we considered it always obtained the best results when compared to both NB classifiers and LR. Another interesting outcome was the connection between the*sparsity* of the dataset and the choice of the features to be included in the model. Generally speaking, *collaborative* features emerged as the most informative ones: when the dataset is not sparse, they always tend to obtain the best results. Also non-personalized *popularity-based* features performed well, especially when coupled with collaborative ones. Conversely, content-based features did not provide a clear benefit to the overall representation of the items.

On the other side, when data are sparse, collaborative features need to be replaced with different information sources. In this case the benefit of injecting the exogenous knowledge coming from the *Linked Open Data* cloud particularly emerged. Indeed, on more sparse datasets the best configurations always included features gathered from the LOD cloud along with collaborative and popularity-based ones. Moreover, when the number of ratings for each item is low (as for `Last.fm` dataset), LOD features can even replace collaborative ones, thus representing an interesting alternative source for data points in *cold-start* settings. Our experiments also showed the usefulness of *topological graph-based* features. Even if in most of the experiments they did not provide any benefit to our model, the combined use of LOD-based and graph-based features on sparse datasets as `DBbook` or `Last.fm` led to the best overall results. Finally, we also compared our methodology to several state-of-the-art baselines and the results showed that our approach can overcome most of the algorithms taken into consideration, thus confirming the insight behind this research.

As future work we will investigate several research lines. We will introduce or replace some groups of features: for example, we could replace the content-based feature family with more advanced representations of items based on word embedding techniques [39]. This would allow to replace the vector-space representation of items with a lower-dimensional vector space representation learned by analyzing the usage of the terms in (very) large corpora of textual documents. Among the most popular techniques, we can cite Latent Semantic Indexing, Random Indexing and Word2Vec [39]. Similarly, we could extend the set of topological graph-based features, or we could even learn compositional vector space representations of the whole knowledge graphs connecting users, items and LOD properties, using recent techniques as the *holographic embeddings* (HolE) [43]. Finally, we also intend to evaluate the impact of such groups of features on other evaluation metrics, as the *novelty* or the *diversity* of the recommendations.

References

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, DBpedia: A Nucleus for a Web of Open Data, Springer, 2007.
[2] R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, 463, ACM press New York, 1999.
[3] P. Basile, A. Caputo, G. Semeraro, An enhanced lesk word sense disambiguation algorithm through a distributional semantic model, in: J. Hajic, J. Tsujii (Eds.), COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23–29, 2014, Dublin, Ireland, ACL, 2014, pp. 1591–1600.
[4] P. Basile, C. Musto, M. de Gemmis, P. Lops, F. Narducci, G. Semeraro, Aggregation Strategies for Linked Open Data-enabled Recommender Systems, in: M. Stankovic (Ed.), Semantic Web Evaluation Challenges (SemWebEval 2014), Crete, Greece, May 25–29, 2014.
[5] 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, in: Communications in Computer and Information Science, 475, Springer, 2014, pp. 163–169.
[6] 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.
[7] T. Berners-Lee, J. Hendler, O. Lassila, et al., The semantic web, Sci. Am. 284 (5) (2001) 28–37.
[8] C. Bizer, The emerging web of linked data, IEEE Intell. Syst. 24 (5) (2009) 87–92.
[9] 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.
[10] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32, doi:10.1023/A:1010933404324.
[11] R. Burke, Hybrid recommender systems: survey and experiments, User Model User-adapt Interact 12 (4) (2002) 331–370.
[12] D. Cox, The regression analysis of binary sequences, J. R. Stat. Soc. B 20 (1958) 215–242. (with discussion)
[13] P. Cremonesi, Y. Koren, R. Turrin, Performance of recommender algorithms on top-n recommendation tasks, in: Proceedings of the Fourth ACM Conference on Recommender Systems, ACM, 2010, pp. 39–46.
[14] M. De Gemmis, L. Iaquinta, P. Lops, C. Musto, F. Narducci, G. Semeraro, Learning preference models in recommender systems, in: Preference Learning, Springer Berlin Heidelberg, 2010, pp. 387–407.
[15] 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.
[16] T. Di Noia, I. Cantador, V.C. Ostuni, Linked open data-enabled recommender systems: ESWC 2014 challenge on book recommendation, in: Semantic Web Evaluation Challenge, Springer, 2014, pp. 129–143.
[17] T. Di Noia, R. Mirizzi, V.C. Ostuni, D. Romito, Exploiting the web of data in model-based Recommender Systems, in: Proceedings of the sixth ACM conference on Recommender Systems, ACM, 2012, pp. 253–256.
[18] T. Di Noia, V.C. Ostuni, P. Tomeo, E.D. Sciascio, Sprank: Semantic path-based ranking for top-n recommendations using linked open data, ACM Trans. Intell. Syst.Technol. (TIST) 8 (1) (2016) 9.
[19] 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.
[20] 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.
[21] F.M. Harper, J.A. Konstan, The movielens datasets: history and context, ACM Trans. Interact. Intell. Syst. 5 (4) (2016) 19:1–19:19, doi:10.1145/2827872.
[22] T.H. Haveliwala, Topic-sensitive PageRank: acontext-sensitive ranking algorithm for web search, IEEE Trans. Knowl. Data Eng. 15 (4) (2003) 784–796.
[23] J.L. Herlocker, J.A. Konstan, A. Borchers, J. Riedl, An algorithmic framework for performing collaborative filtering, in: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 1999, pp. 230–237.
[24] P.B. Kantor, L. Rokach, F. Ricci, B. Shapira, Recommender Systems Handbook, Springer, 2011.
[25] 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.
[26] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.
[27] P. Lops, M. De Gemmis, G. Semeraro, Content-based Recommender Systems: State of the art and trends, in: Recommender Systems handbook, Springer, 2011, pp. 73–105.
[28] P. Lops, M. de Gemmis, G. Semeraro, C. Musto, F. Narducci, M. Bux, A semantic content-based recommender system integrating folksonomies for personalized access, in: Web personalization in intelligent environments, 2009, pp. 27–47.
[29] H. Ma, I. King, M.R. Lyu, Effective missing data prediction for collaborative filtering, in: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2007, pp. 39–46.
[30] H. Mak, I. Koprinska, J. Poon, Intimate: a web-based movie recommender using text categorization, in: Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on, IEEE, 2003, pp. 602–605.
[31] C.D. Manning, H. Schütze, Foundations of Statistical Natural Language Processing, 999, MIT Press, 1999.
[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), 27, 2015, pp. 11–17.
[33] S.E. Middleton, D. De Roure, N.R. Shadbolt, Ontology-based Recommender Systems, in: Handbook on ontologies, Springer, 2004, pp. 477–498.
[34] T.M. Mitchell, Machine learning, McGraw Hill series in computer science, McGraw-Hill, 1997.
[35] R.J. Mooney, L. Roy, Content-based book recommending using learning for text categorization, in: Proceedings of the Fifth ACM Conference on Digital Libraries, ACM, 2000, pp. 195–204.
[36] 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., CEUR Workshop Proceedings, 1245, CEUR-WS.org, 2014, pp. 49–56.
[37] C. Musto, P. Basile, P. Lops, M. de Gemmis, G. Semeraro, Introducing linked open data in graph-based recommender systems, Inf. Process. Manage. 53 (2) (2017) 405–435.
[38] 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, in: UMAP '16, ACM, New York, NY, USA, 2016, pp. 229–237.
[39] C. Musto, G. Semeraro, M. de Gemmis, P. Lops, Learning word embeddings from wikipedia for content-based recommender systems, in: European Conference on Information Retrieval, Springer International Publishing, 2016, pp. 729–734.
[40] C. Musto, G. Semeraro, P. Lops, M. de Gemmis, Random indexing and negative user preferences for enhancing content-based Recommender Systems, in: EC-Web 2011, in: Lecture Notes in Business Inf. Processing, 85, Springer, 2011, pp. 270–281.
[41] 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, in: Lecture Notes in Business Information Processing, 123, Springer, 2012, pp. 112–123.
[42] R. Navigli, Word sense disambiguation: a survey, ACM Comput. Surv. (CSUR) 41 (2) (2009) 10.
[43] M. Nickel, L. Rosasco, T.A. Poggio, Holographic embeddings of knowledge graphs, in: D. Schuurmans, M.P. Wellman (Eds.), Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12–17, 2016, Phoenix, Arizona, USA., AAAI Press, 2016, pp. 1955–1961.
[44] M.P. O'Mahony, P. Cunningham, B. Smyth, An assessment of machine learning techniques for review recommendation, in: Irish Conference on Artificial Intelligence and Cognitive Science, Springer, 2009, pp. 241–250.
[45] V.C. Ostuni, T. Di Noia, E. Di Sciascio, S. Oramas, X. Serra, A semantic hybrid approach for sound recommendation, in: Proceedings of the 24th International Conference on World Wide Web, ACM, 2015, pp. 85–86.
[46] 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 Seventh ACM Conference on Recommender Systems, ACM, 2013, pp. 85–92.
[47] L. Page, S. Brin, R. Motwani, T. Winograd, The PageRank citation ranking: bringing order to the web., Stanford InfoLab, 1999.
[48] A. Passant, dbrec - Music Recommendations Using DBpedia, in: International Semantic Web Conference, Revised Papers, in: LNCS, 6497, Springer, 2010, pp. 209–224.
[49] M.J. Pazzani, J. Muramatsu, D. Billsus, Syskill & Webert: identifying interesting web sites, in: AAAI/IAAI, Vol. 1, 1996, pp. 54–61.
[50] L. Peska, P. Vojtás, Enhancing recommender system with Linked Open Data, in: Flexible Query Answering Systems, Springer, 2013, pp. 483–494.
[51] G. Piao, J.G. Breslin, Measuring semantic distance for linked open data-enabled recommender systems, in: Proceedings of the 31st Annual ACM Symposium on Applied Computing, ACM, 2016, pp. 315–320.
[52] M.F. Porter, An algorithm for suffix stripping, Program 14 (3) (1980) 130–137.
[53] 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.
[54] A. Said, A. Bellogín, Rival: a toolkit to foster reproducibility in recommender system evaluation, in: Proceedings of the 8th ACM Conference on Recommender systems, ACM, 2014, pp. 371–372.
[55] M. Schmachtenberg, T. Strufe, H. Paulheim, Enhancing a location-based recommendation system by enrichment with structured data from the web, in: R. Akerkar, N. Bassiliades, J. Davies, V. Ermolayev (Eds.), 4th International Conference on Web Intelligence, Mining and Semantics (WIMS 14), WIMS '14, Thessaloniki, Greece, June 2–4, 2014, ACM, 2014, pp. 17:1–17:12.
[56] G. Semeraro, P. Lops, M. De Gemmis, C. Musto, F. Narducci, A folksonomy-based recommender system for personalized access to digital artworks, J. Comput. Cult. Heritage (JOCCH) 5 (3) (2012) 11.
[57] H. Steck, Evaluation of recommendations: rating-prediction and ranking, in: Proceedings of the 7th ACM Conference on Recommender Systems, ACM, 2013, pp. 213–220.
[58] T. Strohman, W.B. Croft, D. Jensen, Recommending citations for academic papers, in: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2007, pp. 705–706.
[59] A. Tiroshi, S. Berkovsky, M.A. Kaafar, T. Chen, T. Kuflik, Cross social networks interests predictions based on graph features, in: Proceedings of the 7th ACM Conference on Recommender Systems, ACM, 2013, pp. 319–322.
[60] A. Tiroshi, S. Berkovsky, M.A. Kâafar, D. Vallet, T. Chen, T. Kuflik, Improving business rating predictions using graph based features, in: T. Kuflik, O. Stock, J.Y. Chai, A. Krüger (Eds.), IUI'14 19th International Conference on Intelligent User Interfaces, IUI'14, Haifa, Israel, February 24–27, 2014, ACM, 2014, pp. 17–26, doi:10.1145/2557500.2557526.
[61] D.B. West, et al., Introduction to Graph Theory, 2, Prentice hall Upper Saddle River, 2001.
[62] P. Xia, B. Liu, Y. Sun, C. Chen, Reciprocal recommendation system for online dating, in: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015, ACM, 2015, pp. 234–241.
[63] H.-R. Zhang, F. Min, Three-way recommender systems based on random forests, Knowl. Based Syst. 91 (2016) 275–286.
[64] T. Zhang, V.S. Iyengar, Recommender systems using linear classifiers, J. Mach. Learn. Res. 2 (Feb) (2002) 313–334.

Please cite this article as: C. Musto et al., Semantics-aware Recommender Systems exploiting Linked Open Data and graph-based features, Knowledge-Based Systems (2017), http://dx.doi.org/10.1016/j.knosys.2017.08.015