Semantics-aware Recommender Systems Exploiting Linked Open Data and Graph-based Features

Cataldo Musto
Dept. of Computer Science - University of Bari Aldo Moro
cataldo.musto@uniba.it

Pasquale Lops
Dept. of Computer Science - University of Bari Aldo Moro
pasquale.lops@uniba.it

Marco de Gemmis
Dept. of Computer Science - University of Bari Aldo Moro
marco.degemmis@uniba.it

Giovanni Semeraro
Dept. of Computer Science - University of Bari Aldo Moro
giovanni.semeraro@uniba.it

ABSTRACT

In this contribution we propose a *hybrid recommendation framework* based on classification algorithms such as Random Forests and Naive Bayes, which are fed with several heterogeneous groups of features. We split our features into two classes: *classic features*, as popularity-based, collaborative and content-based ones, and *extended features* gathered from the Linked Open Data (LOD) cloud, as basic ones (i.e. *genre* of a movie or the *writer* of a book) and graph-based features calculated on the ground of the different topological characteristics of the *tripartite* representation connecting users, items and properties in the LOD cloud. In the experimental session we evaluate the effectiveness of our framework on varying of different groups of features, and results show that both LOD-based and graph-based features positively affect the overall performance of the algorithm, especially in *highly sparse* recommendation scenarios. Our approach also outperforms several state-of-the-art recommendation techniques, thus confirming the insights behind this research.

This extended abstract summarizes the content of the journal paper [7] published on *Knowledge-based Systems*.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; *Web data description languages*; • **Computing methodologies** → *Supervised learning by classification*;

KEYWORDS

Recommender Systems, Machine Learning, Linked Open Data

ACM Reference Format:
Cataldo Musto, Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. 2018. Semantics-aware Recommender Systems Exploiting Linked Open Data and Graph-based Features. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3184558.3186233

WWW '18 Companion, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5640-4/18/04.
https://doi.org/10.1145/3184558.3186233

1 INTRODUCTION

According to recent statistics[1], 150 billions of RDF triples and almost 10,000 linked datasets are now available in the so-called LOD cloud: such RDF triples interconnect in a semantics-aware fashion the information covering many topical domains, such as geographical locations, people, books, films, music, and so on. The *nucleus* of such data is commonly represented by DBpedia [1], the RDF mapping of Wikipedia. This huge availability of semantics-aware machine-readable data attracted researchers and practitioners in the area of Content-based Recommender Systems (RS) [4], willing to investigate how such information can be exploited to improve the effectiveness of existing algorithms or to tackle several problems RSs typically suffer from.

In this article we investigate the impact of such *exogenous knowledge* on the performance of a *hybrid* recommendation framework based on classification techniques as Random Forests and Naive Bayes. In this work we followed the hybridization strategy which is typically referred to as *feature combination* [2], that is to say, we represented the items by means of different heterogeneous groups of features and we used this unique representation to feed the classifiers with training examples. 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 are roughly classified in two groups: *classic features* and *extended* ones. The features that are typically used in hybrid item representations, as unstructured *content-based* features, *collaborative* features and simple *popularity-based* ones, fall in the first group. Next, we extended the representation by introducing data points gathered from the LOD cloud, as *basic* structured features (as the *genre* of a movie or the *writer* of a book) and *graph-based features*, calculated by mining the different topological characteristics of the *tripartite* graph-based representation that connects users, items and properties in the LOD cloud.

In the experimental session we evaluated the effectiveness of our framework on varying of these sets of features, and results provided several interesting insights, since it emerged that the overall accuracy significantly benefits from the introduction of LOD-based and graph-based features. Moreover, the results we obtained also overcame several state-of-the-art recommendation techniques.

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

2 METHODOLOGY

In this section we provide the details of our methodology, by introducing the groups of features we used to feed the classification algorithms and by describing our recommendation framework.

2.1 Description of the Features and Recommendation Framework

Popularity features. This set of features includes basic popularity-based information about the items, such as the *number of ratings* received by the item, the number of *positive* ratings received by the item and the *ratio* between positive ratings and the overall number of ratings.

This (tiny) group of features may seem trivial and not useful, but this kind of data is typically very informative for a recommendation task, since it gives information about how popular is a certain item among the users and how positive is their general opinion about it. As Cremonesi et al. already shown [3], non-personalized algorithms based on simple popularity measures can obtain performance comparable to that of more sophisticated techniques.

Collaborative features. This class of features models the information encoded in the *user-item matrix* which is typically exploited in collaborative filtering (CF) algorithms. Differently from classical CF algorithms, that use the *whole* matrix to calculate the *neighbors* of the target user and to predict the items the user may be interested in, 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 number of the *rows* of the matrix, that is to say, to the number of the *users* in the dataset.

The choice of including this set of features in our hybrid representation is quite straightforward, since CF algorithms and matrix factorization techniques tend to obtain very good performance especially when the *sparsity* of the original matrix is not high.

Content-based features. Textual content is another interesting source that can be exploited to provide items with useful and descriptive features. As an example, the *plot* of a movie contains several distinctive characteristics of the item, which can be extracted from such data.

However, *textual descriptions* are typically *noisy*, thus it is necessary to properly process such data by adopting Natural Language Processing (NLP) techniques before including them in our items representation. In our pipeline the content was first tokenized, then stop-words were removed and the entities occurring in the text were identified. Next, the remaining tokens were stemmed. In this case, the amount of features added to the model corresponds to the size of the *vocabulary*, that is to say, to the number of different tokens occurring in the description of *all the items* in the dataset.

LOD-based features. The first group of *extended features* includes structured basic properties gathered from the LOD cloud, as the *genre* of a *movie* or the author of a book. To gather LOD-based features we preliminarily carried out a mapping procedure to obtain the corresponding URI for each item in the dataset. 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[2]. It is worth to emphasize that the mapping is a necessary and mandatory step to get an *entry point* to the LOD cloud.

Table 1: Partial representation of the vector modeling the LOD-based features extracted from DBpedia for the movie *The Matrix*

property - value	The Matrix
dbo:director - dbr:The_Wachowski	1
dbo:director - dbr:Mel_Gibson	0
dbo:composer - dbr:Ennio_Morricone	0
dct:subject - dbc:Dystopian_films	1
dct:subject - dbc:American_Horror_movies	0
...	...
dbo:producer - dbr:Joel_Silver	1

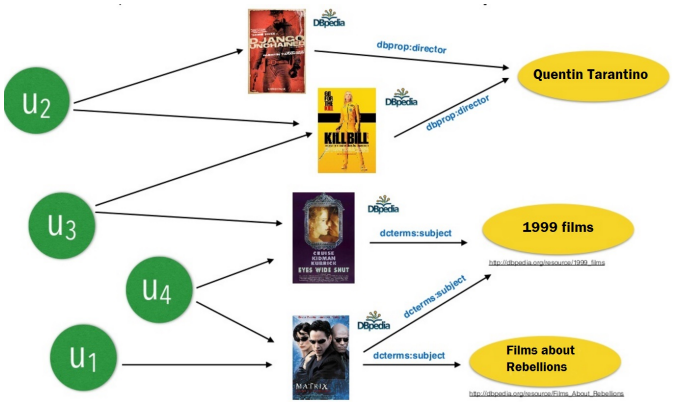

Figure 1: A toy example of a tripartite graph, modeling users, items and properties gathered from the LOD.

Next, for each domain, we defined a subset of relevant properties by exploiting the outcomes of our previous research [5, 6] and finally we used SPARQL to extract such data.

As we did for content-based features, we built a *vocabulary* of LOD-based properties and we provided each item with these new features. The score of each feature was set to 1 if the item is described through that RDF property, 0 otherwise. Table 1 reports some of the properties describing *The Matrix* gathered from the LOD cloud. In this case, each feature is represented through the couple <*property,value*>, since each entity can have different roles in the same movie (and in different ones, as well).

Graph-based Features. The second group of extended features is built on the ground of the graph-based representation obtained by connecting the users to the items they liked and, in turn, the items to the properties gathered from DBpedia (see Figure 1). We refer to these features as *tripartite* ones.

Given such representations, we decided to mine this graph and to calculate some measures describing its *topological characteristics*. Specifically, in our item representation we encoded five graph-based features calculated on the tripartite user-item graph, that is to say:

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

Degree Centrality, *Average Neighbor Degree*, *PageRank* score, *Node Redundancy* and *Cluster Coefficient*.

Recommendation Framework. In this work we *cast* the recommendation task to a classification one, that is to say, we used the vectors representing the items the user liked as *positive examples* and those he did not like as *negative examples*. Next, we trained the classifiers and we exploited them to classify all the items the user did not consumed yet as *interesting* or not *interesting* for her.

To sum up, given a target user u, a training set $TR(u)$ (the items the user previously rated), and a group of features F, our classifier is fed with the examples $i_F \in TR(u)$ and we use the classification model to predict the most interesting items for the target user. Specifically, items in the test set are ranked according to the *confidence* of the prediction returned by the classification algorithm and the *top-K* items are returned to the target user. In the experimental session the overall effectiveness of our recommendation framework has been evaluated by varying different sets of features and by using two different classification algorithms, namely *Random Forests* and *Naïve Bayes*.

3 EXPERIMENTAL EVALUATION

Our experiments were designed on the ground of three different research questions: How do *LOD-based features* impact on the overall performance of the recommendations? (*Experiment 1*). How do *graph-based features* impact on the overall performance of the recommendations? (*Experiment 2*). How does our best-performing configuration perform with respect to *state-of-the-art techniques*? (*Experiment 3*).

Experimental protocol. Experiments were carried out on two state-of-the-art datasets, i.e. `MovieLens-1M`[3], and DBbook. The first one is a widespread dataset for movie recommendation, the second was used in the ESWC 2014 Recommender Systems challenge[4] and focuses on book recommendation.

Experiments were performed by adopting different protocols. We used a 80%-20% training-test split for `MovieLens-1M`. For `DBbook` we used the training-test split that provided with the data. Different protocols were also adopted to build user profiles. In `MovieLens-1M`, user preferences are expressed on a 5-point discrete scale, thus we decided to consider as *positive* only those ratings equal to 4 and 5. 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 *Random Forest* and *Naive Bayes* made available in the Weka Toolkit[5].

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[6] library for tokenization, language detection and stop-words removal. Textual descriptions were all gathered from the Wikipedia pages of the items. Finally, tokens were stemmed by exploiting the Snowball library [7].

Table 2: Impact of LOD-based Features on `MovieLens` data.

	RF		NB	
F1@5	*No-LOD*	LOD	*No-LOD*	LOD
Popular (P)	*0.5338*	0.5312	*0.5458*	0.5320
Collaborative (C)	*0.5618*	0.5609	*0.5486*	0.5450
Content-based (T)	*0.4913*	**0.4943**	*0.4913*	**0.4932**
P+C	*0.5635*	**0.5642 (*)**	*0.5483*	0.5451
P+T	*0.5051*	**0.5079**	*0.4965*	**0.4974**
C+T	*0.5187*	**0.5188**	*0.5180*	0.5169
P+C+T	*0.5246*	0.5246	*0.5189*	0.5174

As previously explained, each item was mapped to a `DBpedia` entry in order to gather the features from the LOD cloud. To this end, we exploited some mappings already available in literature. In our setting, 3,300 `MovieLens-1M` entries and 6,600 items (98.02%) from `DBbook` (85% of the items) were successfully mapped. The items for which a `DBpedia` entry was not found were represented by using the basic groups of features alone. Finally, *graph-based features* were calculated by exploiting the Jung framework[8], a Java library to manage graph-based data. As previously explained, for each item node we calculated *Degree Centrality*, *Average Neighbor Degree*, *PageRank* score, *Node Redundancy* and *Cluster Coefficient* for tripartite graph.

The performance of each configuration of our recommendation framework was evaluated in terms of *F1@5*, calculated through the Rival toolkit[9].

Discussion of the Results. By analyzing the behavior of LOD features on `MovieLens` data (Table 2), it emerges that the only configuration that benefits of such injection is the one exploiting *content-based features*. This can be probably due to the low sparsity of the dataset, which makes superfluous most of the features except *collaborative* ones. However, even if these experimental settings showed that the adoption of LOD features has to be carefully evaluated, the overall best configuration (highlighted with (*)) actually *includes LOD features*, since the configuration merging popular, collaborative and LOD features obtained the higher F1@5. A similar pattern was noted on `DBbook`, since RF is the algorithm which takes the best from the LOD-based features. An interesting outcome emerging from this experiment is that when data are sparse, as for `DBbook`, *LOD-based* data points represent a good alternative also to *collaborative* features. Indeed, in this experiment *Popular+LOD* obtained the best overall F1@5. This means that, when the rating patterns are noisy, LOD features can be used to enrich the representation with new and relevant information.

Next, we evaluated the impact of graph-based features on our recommendation framework. For each dataset we considered as *baseline* the best-performing configuration emerged from the previous tables and we extended the representation by introducing *tripartite* features. By considering `MovieLens` dataset, a positive

[3] http://grouplens.org/datasets/movielens/1m/
[4] http://challenges.2014.eswc-conferences.org/index.php/RecSys
[5] http://www.cs.waikato.ac.nz/ml/weka/
[6] https://lucene.apache.org/
[7] http://snowball.tartarus.org/
[8] http://jung.sourceforge.net/
[9] http://rival.recommenders.net/

Table 3: Impact of LOD-based Features on DBbook data.

	RF		NB	
F1@5	*No-LOD*	LOD	*No-LOD*	LOD
Popular (P)	*0.5610*	**0.5659 (*)**	*0.5576*	**0.5577**
Collaborative (C)	*0.5421*	**0.5560**	*0.5610*	0.5564
Content-based (T)	*0.5532*	**0.5551**	*0.5465*	**0.5494**
P+C	*0.5627*	**0.5630**	*0.5615*	0.5580
P+T	*0.5567*	**0.5569**	*0.5467*	**0.5497**
C+T	*0.5549*	**0.5553**	*0.5464*	**0.5491**
P+C+T	*0.5583*	0.5560	*0.5468*	**0.5497**

Table 4: Impact of Graph-based Features.

	MovieLens		DBbook	
	RF	NB	RF	NB
Baseline	*0.5635*	*0.5486*	*0.5627*	*0.5615*
Baseline+Trip.	0.5621	0.5483	0.5607	0.5542
Baseline+LOD	*0.5642*	*0.5451*	*0.5659*	*0.5580*
Baseline+LOD+Trip.	**0.5678(*)**	**0.5481**	**0.5667(*)**	**0.5589**

impact only emerged when *graph-based* features are merged with *LOD-based* ones. Indeed, both RF and NB are able to improve F1@5 with a statistically significant improvement when *tripartite graph-based features* are exploited. This means that the topological information coming from the injection of the features gathered from the LOD cloud can improve the performance of our framework. Overall, the best configuration for ML data is that based on both *LOD-based* and *tripartite graph-based* features which uses RF. Similar outcomes emerge if we take into account the results on DBbook data. Also in this case, when *LOD-based* features are included in the representation, graph-based features produce a significant increase of F1@5.

In the last experiment we compared the effectiveness of our hybrid recommendation methodology with several state of the art recommendation algorithms, as User-to-User (U2U-KNN), Item-to-Item Collaborative Filtering (I2I-KNN), the Bayesian Personalized Ranking (BPRMF) and an implementation of PageRank with Priors. Moreover, we also compared our methodology to other *LOD-aware recommendation techniques*. As future work, we plan to compare our approach also to other semantics-aware RS [8]. Specifically, we used the features gathered from the LOD as side information for BPRMF and we also extended PageRank with Priors (PPR) with LOD-based features as we investigated in our previous research [5]. PPR was run by using default settings (80% of the weight distributed to the items the user liked). 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 already available in MyMediaLite[10], while the methods implemented in the Jung framework[11] were used to run PPR.

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

[11]http://jung.sourceforge.net/

Table 5: Comparison to state of the art algorithms

	F1@5	
Algorithm	MovieLens-1M	DBbook
LOD-RecSys	**0.5678**	**0.5667**
U2U-KNN	0.4270	0.5193
I2I-KNN	0.4320	0.5111
BPRMF	0.5218	0.5290
BPRMF+LOD	0.5215	0.5304
PPR	0.5397	0.5502
PPR+LOD	0.5400	0.5540

As shown in Table 5, our hybrid recommendation framework always overcomes all the baselines on MovieLens-1M and DBbook data. All the increases are statistically significant. 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.

To sum up, several interesting outcomes emerge from these experiments: first, RF was the classification algorithm able to take the best out of our hybrid data representation. Another interesting outcome is the connection between the *sparsity* of the dataset and the choice of the features to be included in the model. When the dataset is not sparse, *collaborative* features along with non-personalized *popularity-based* emerge as the most informative ones. On the other side, when data are sparse, collaborative features need to be replaced or coupled with different information sources. These results further confirmed the outcomes behind this research, since they clearly showed that the injection of exogenous data points gathered from the LOD cloud (in the form of both *semantics-aware content-based features* and *topological tripartite* ones) can significantly improve the predictive accuracy of our recommendation framework, leading to an interesting improvement over all the state-of-the-art baselines.

REFERENCES

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R.Cyganiak, and Z. G. Ives. 2007. DBpedia: A Nucleus for a Web of Open Data. In *ISWC 2007 (Lecture Notes in Computer Science)*, Vol. 4825. Springer, 722–735. DOI: http://dx.doi.org/10.1007/978-3-540-76298-0_52

[2] R. Burke. 2002. Hybrid recommender systems: Survey and experiments. *UMUAI* 12, 4 (2002), 331–370.

[3] P. Cremonesi, Y. Koren, and R. Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *ACM RECSYS*. ACM, 39–46.

[4] M. de Gemmis, P. Lops, C. Musto, F. Narducci, and G. Semeraro. 2015. Semantics-Aware Content-Based Recommender Systems. In *Recommender Systems Handbook*. Springer, 119–159.

[5] C. Musto, P. Basile, P. Lops, M. de Gemmis, and G. Semeraro. 2017. Introducing linked open data in graph-based recommender systems. *Information Processing & Management* 53, 2 (2017), 405–435.

[6] C. Musto, P. Lops, P. Basile, M. de Gemmis, and 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 2016)*. ACM, 229–237.

[7] C. Musto, P. Lops, M. de Gemmis, and G. Semeraro. 2017. Semantics-aware Recommender Systems exploiting Linked Open Data and graph-based features. *Knowledge-Based Systems* 136, Supplement C (2017), 1 – 14.

[8] C. Musto, G. Semeraro, P. Lops, and M. de Gemmis. 2011. Random Indexing and Negative User Preferences for Enhancing Content-Based Recommender Systems. In *EC-Web 2011 (Lecture Notes in Business Inf. Processing)*, Vol. 85. Springer, 270–281.