Content-Based Recommender Systems + DBpedia Knowledge = Semantics-Aware Recommender Systems

Pierpaolo Basile(✉), Cataldo Musto, Marco de Gemmis, Pasquale Lops, Fedelucio Narducci, and Giovanni Semeraro

Department of Computer Science, University of Bari Aldo Moro,
Via E. Orabona 4, 70125 Bari, Italy
{pierpaolo.basile,cataldo.musto,marco.degemmis,pasquale.lops,
fedelucio.narducci,giovanni.semeraro}@uniba.it

Abstract. This paper provides an overview of the work done in the ESWC Linked Open Data-enabled Recommender Systems challenge, in which we proposed an ensemble of algorithms based on popularity, Vector Space Model, Random Forests, Logistic Regression, and PageRank, running on a diverse set of semantic features. We ranked 1st in the top-N recommendation task, and 3rd in the tasks of rating prediction and diversity.

1 Introduction and Description of the Challenge

Over the last years, more and more semantic data are published following the Linked Data principles. These datasets, interlinked with each other, form a global graph, called Linked Open Data (LOD) cloud. In the context of recommender systems, this data might be useful to interlink information about users, items, and their relations. The challenge is to investigate whether and how this large amount of linked knowledge may help to mitigate the cold-start and the data sparsity problems. This was the primary goal of the LOD-enabled Recommender Systems challenge, aiming to show how LOD can boost the creation of a new breed of knowledge-enabled and content-based recommender systems. The contest consisted of 3 tasks: *Task 1: Rating Prediction in Cold-start Situations*, i.e. when users have a few past ratings, and when items have been rated by a few users; *Task 2: Top-N Recommendation from Binary User Feedback*, i.e. generating ranked lists of items for which only binary ratings are available; *Task 3: Diversity*, i.e. evaluation of both accuracy of the recommendation list, and diversity of items in the list (in terms of Intra-List Diversity - ILD). Given the domain of books, diversity is measured with respect to the properties: http://dbpedia.org/ontology/author and http://purl.org/dc/terms/subject.

The dataset used is DBbook, which contains user data and preferences retrieved from the Web in the book domain. Each book is mapped to the corresponding DBpedia URI. The mapping contains 8,170 DBpedia URIs, which can be used to extract features from datasets in the LOD cloud. The training set

V. Presutti et al. (Eds.): SemWebEval 2014, CCIS 475, pp. 163–169, 2014.
DOI: 10.1007/978-3-319-12024-9_21

for Task 1 contains 75,559 ratings (scale 0–5) provided by 6,181 users on 6,166 items which have been rated by at least one user. The training set for Task 2 and Task 3 contains 72,372 binary ratings provided by 6,181 users on 6,733 items.

2 Description of the UNIBA Approach

2.1 Methods

The methodology adopted by UNIBA is based on a blend of the following methods/algorithms to face the three different tasks of the challenge:

(1) Popularity: item-based popularity recommender, where the popularity of an item is computed as the ratio between the number of positive ratings and the total number of ratings (positive and negative) it received by all users.

(2) enhanced Vector Space Model (eVSM) with negation: content-based recommender based on an incremental dimensionality reduction technique called Random Indexing. Details about the approach are in [4], in which a negation operator [6] is adopted to represent negative preferences, besides positive ones.

(3) PageRank with Priors: widely used method to obtain an authority score for a node based on the network connectivity, in which a non-uniform personalization vector may be used for assigning different weights to different nodes [3].

(4) Random Forests (RF) [1]**:** ensemble learning method used for classification or regression, which combines different tree predictors constructed using different samples of the training data and random subsets of the data features.

(5) Logistic Regression (LR): supervised learning method for classification which builds a linear model based on a transformed target variable.

2.2 Data Model

The above mentioned methods used a combination of the following features:

(1) Keywords: we processed the book descriptions extracted from Wikipedia. Stopwords were removed, and keywords were stemmed. For books not existing in Wikipedia, we processed the DBpedia abstracts.

(2) Tagme concepts: Tagme [2] implements an anchor disambiguation algorithm to produce a Wikipedia-based representation of text fragments, where the most relevant concepts occurring in the text are mapped to the Wikipedia articles (i.e. DBpedia nodes) they refer to. Tagme performs a sort of feature selection by filtering out the noise in text fragments, and its main advantage is the ability to annotate very short texts. As an example, the resulting representation obtained for the book *The Great and Secret Show* is: *Dead letter office, Nebraska, New Mexico, Quiddity, Ephemeris, Narcissistic personality disorder, Nuncio, California, Rape.* Interestingly, the technique is able to associate several concepts which are somehow related to the book, and which could be useful to provide accurate and diverse recommendations, as well.

(3) DBpedia properties: for each book, we selected the following 10 most frequent properties in DBpedia (http://dbpedia.org/ prefix removed for brevity): (1) `ontology/wikiPageWikiLink`, providing the link from a Wikipedia page to another Wikipedia page. This property allows to take into account other Wikipedia pages which are somehow related; (2) http://purl.org/dc/terms/subject, providing the topic of a book; (3) `property/genre`, providing the genre of a book; (4) `property/publisher`, providing the publisher of a book; (5) ontology/author, providing the author of a book; (6) `property/followed` By, providing the book followed by a specific book; (7) `property/precededBy`, providing the book preceded by a specific book; (8) property/series, providing the series of a book; (9) `property/dewey`, providing the Dewey Decimal library Classification code; and (10) `ontology/nonFictionSubject` providing the subject of a non-fiction book (e.g.: history, biography, cookbook, ...). PageRank with Priors is performed (for each single user) using graphs with different sets of nodes. Initially, only users, items and links represented by the positive feedback are included; next, we enriched the graph with the 10 properties extracted from DBpedia. Then, we ran a second level expansion stage of the graph to retrieve the following additional resources: (1) internal wiki links of the new added nodes; (2) more generic categories according to the hierarchy in DBpedia; (3) resources of the same category; (4) resources of the same genre; (5) genres pertaining to the author of the book; (6) resources written by the author; and (7) genres of the series the book belongs to.

The graph is pruned by removing nodes which are neither users nor books having a total number of inlinks and outlinks less than 5, and eventually consisted of 340,000 nodes and about 6 millions links.

3 Experimental Evaluation

3.1 Task 1: Rating Prediction in Cold-Start Situations

We ranked 3rd in Task 1 using a *linear combination* of the following algorithms, by obtaining a RMSE equal to 0.8742:

Random Forests, using 2,500 trees, and *Tagme concepts* as features, along with *DBpedia properties* described in Sect. 2.2. We adopted the implementation provided by the Weka library (www.cs.waikato.ac.nz/ml/weka/).

Logistic Regression, using the following features: number of positive, negative and total feedbacks provided by the users (items), ratio between positive (negative) and total number of feedbacks provided by the users (items), stems extracted by the item descriptions, *DBpedia properties* (Sect. 2.2), and *Tagme concepts*. As regards the last three sets of features, their value is the number of occurrences of that feature. Each example, represented using more than 220,000 features, is labelled with the rating provided by that specific user for that specific item. All the features were normalized in the [0,1] interval. We adopted the implementation provided by Liblinear[1]. RF and LR ranked items according to the class probability.

[1] http://www.csie.ntu.edu.tw/~cjlin/liblinear/

Combination of baseline predictors, i.e. user/item average rating.

Since 520 out of 6,181 users did not have positive ratings in the training set, we assigned as positive feedback the 5 most popular items (5 is the average number of users' positive ratings in the dataset). Results for Task 1 are reported in Table 1. The weights used in the linear combination (0.2 to RF, 0.2 to the baseline predictors and 0.6 to LR) are selected to maximize performance on testing data, without the use of a validation set.

Table 1. Results for Task 1.

	RF	LR	Baseline predictors	Linear combination
RMSE	0.9285	0.8915	0.8945	**0.8742**

3.2 Task 2: Top-N Recommendation from Binary User Feedback

We ranked 1st in Task 2 by blending together the following five different algorithms, using the Borda count aggregation method:

eVSM: we implemented a content-based recommender as described in [4]. The best result was obtained using *Tagme concepts* as features, 500 as the context vectors dimension, and the negation operator for negative users' preferences.

Popularity: simple baseline as described in Sect. 2.1, which recommends items by ranking them according to their popularity (in decreasing order).

Random Forests: we used 5,000 trees and the same features as in Task 1.

PageRank with Priors: a different configuration of weights is assigned to the nodes. Generally, the prior probability assigned to each node is evenly distributed ($\frac{1}{N}$, where N is the number of nodes). We assigned a higher weight to some nodes according to the user profile. More specifically, 80 % of the weight is evenly distributed among books liked by the users (0 assigned to books disliked by the users), and 20 % of the weight is evenly distributed among the remaining nodes. The damping factor of PageRank was set to 0.85. Both weights and damping factor are chosen after a tuning step on a subset of the training data. The PageRank computed for each node is used to rank the items in the test set. We adopted the implementation of PageRank provided by the Jung library[2].

Logistic Regression: the configuration is as in Task 1. The only difference is that each example is labelled with the binary feedback provided by that specific user for that specific item.

Similarly to Task 1, RF and LR ranked items according to the probability of the class, and the 5 most popular items are used for users with no positive ratings in the training set. Table 2 reports the performance of the single methods, eventually aggregated using the linear combination and Borda count. As regards the linear

[2] jung.sourceforge.net

Table 2. Results for Task 2.

	eVSM	Popularity	RF	PageRank	LR	Linear Comb.	Borda
Pr@5	0.6195	0.6431	0.6260	0.6433	0.6445	0.6568	**0.6586**
Re@5	0.4688	0.4875	0.4751	0.4871	0.4888	0.5009	**0.5048**
F1@5	0.5337	0.5546	0.5402	0.5544	0.5560	0.5684	**0.5715**

combination, we assigned 0.1 to eVSM, 0.2 to the popularity baseline and to LR, and finally 0.25 to RF and Page Rank. As for Task 1, the weights were set after a rough tuning.

In Borda count, each item in a ranked list produced by each single method is awarded with a score given according to its position in that list. The lower the item position in the list, the smaller the score. The final score of each item is obtained by summing all the single scores, and this allows to produce the aggregated ranking (in decreasing score value). The single scores in the sum were weighed in order to boost some single methods (weights are reported in parenthesis). As for Task 1, weights are chosen to maximize performance on testing data.

3.3 Task 3: Diversity

We ranked 3rd in the Task 3 by using the PageRank with Priors algorithm, running on the graph described in Sect. 2.2. We assigned a higher weight to some nodes according to the user profile, and to a heuristic of diversity. More specifically, 80 % of the weight is evenly distributed among books liked by the users (0 for books disliked by the users), 10 % of the weight is evenly distributed between all the nodes which are not books, and 10 % of the weight is proportionally distributed among the remaining books (not rated as positive or negative) according to a *diversity score* computed for each item. The diversity score of each item it_j with respect to the profile u_i of the user i is computed in order to take into account both the *similarity* of, and the *novelty* between the user profile and the item. Let U_i the set of *DBpedia properties* of items liked by the user i, and I_j the set of *DBpedia properties* of it_j. The similarity is computed as the Jaccard index between U_i and I_j, while the novelty is the ratio between the cardinality of $I_j \setminus U_i$ (i.e. the set of features of I_j different from those of items liked by the user), and the cardinality of I_j. If the item has features not overlapping with those occurring in the user profile, the similarity is equal to 0, and the novelty is equal to 1. The diversity score is an average between similarity and novelty. Weighing more those items with a higher diversity score allows to impose a bias to the PageRank towards items different from the user profile. The final score computed by the PageRank for each node is used to rank the nodes. Then, the top-20 (book) nodes are selected, as requested by the task. The results obtained by our algorithm are: F@20 = 0.0481 (Pr@20 = 0.0319, Re@20 = 0.0977), and ILD@20 = 0.4717.

4 Discussion

An important outcome of our participation to the challenge is that it was not possible to face all the different tasks using just a single method. We ran hundreds of experiments using different algorithms and features. Results are not reported in the paper due to space limitation, but allow to draw important conclusions. Very simple algorithms based on Vector Space Model and probabilistic models (BM25 and Divergence from Randomness) have performance comparable to more complex algorithms, when fed with semantic features coming from the LOD cloud. The usefulness of the semantic features is also evident when using recommendation algorithms based on classifiers, such as RF or LR, in which the best results were obtained using features based on *DBpedia properties* and *Tagme concepts*. The use of LOD also helps to diversify the results, due to the wealth of relations taken into account in the recommendation process. To sum up, there is an empirical evidence of the potential of the LOD to define advanced semantic recommender systems, even though it is necessary to investigate innovative ways to leverage this huge amount of knowledge. When compared to (few) previous attempts to use LOD to build recommender systems, the novelty of our methods relies on 1) the use of entity linking approaches, such as *Tagme*, which represents an innovative way to access DBpedia knowledge, and on 2) the use of domain-specific DBpedia properties/paths to build the graph model. As to the former aspect, the typical way to define an entry point to DBpedia is to identify the URIs corresponding to items (books for example) and extract the corresponding properties. This complex process of mapping may hinder the use of DBpedia; indeed, the organizers of the challenge explicitly provided a mapping of books to DBpedia URIs. The use of entity linking algorithms represents a novel way to access the DBpedia knowledge through the analysis of the item descriptions, without exploiting any explicit mapping of items to URIs. As regards the exploitation of domain-specific properties/paths in DBpedia, this could allow to fully exploit the semantics of DBpedia relations, differently from previous approaches based just on link-based measures built on DBpedia [5].

References

1. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
2. Ferragina, P., Scaiella, U.: Fast and accurate annotation of short texts with wikipedia pages. IEEE Softw. **29**(1), 70–75 (2012)
3. Haveliwala, T.H.: Topic-sensitive pagerank: a context-sensitive ranking algorithm for web search. IEEE Trans. Knowl. Data Eng. **15**(4), 784–796 (2003)
4. Musto, C., Semeraro, G., Lops, P., de Gemmis, M.: Random indexing and negative user preferences for enhancing content-based recommender systems. In: Huemer, C., Setzer, T. (eds.) EC-Web 2011. LNBIP, vol. 85, pp. 270–281. Springer, Heidelberg (2011)

5. Passant, A.: dbrec — music recommendations using DBpedia. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part II. LNCS, vol. 6497, pp. 209–224. Springer, Heidelberg (2010)
6. Widdows, D.: Orthogonal negation in vector spaces for modelling word-meanings and document retrieval. In: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, pp. 136–143 (2003)