Linked Open Data-enabled Strategies for Top-N Recommendations

Cataldo Musto
Dept. of Computer Science
Univ. of Bari Aldo Moro, Italy
cataldo.musto@uniba.it

Pierpaolo Basile
Dept. of Computer Science
Univ. of Bari Aldo Moro, Italy
pierpaolo.basile@uniba.it

Pasquale Lops
Dept. of Computer Science
Univ. of Bari Aldo Moro, Italy
pasquale.lops@uniba.it

Marco de Gemmis
Dept. of Computer Science
Univ. of Bari Aldo Moro, Italy
marco.degemmis@uniba.it

Giovanni Semeraro
Dept. of Computer Science
Univ. of Bari Aldo Moro, Italy
giovanni.semeraro@uniba.it

ABSTRACT

The huge amount of interlinked information referring to different domains, provided by the Linked Open Data (LOD) initiative, could be effectively exploited by recommender systems to deal with the *cold-start* and *sparsity* problems.

In this paper we investigate the contribution of several features extracted from the Linked Open Data cloud to the accuracy of different recommendation algorithms. We focus on the *top-N* recommendation task in presence of binary user feedback and cold-start situations, that is, predicting ratings for users who have a few past ratings, and predicting ratings of items that have been rated by a few users.

Results show the potential of Linked Open Data-enabled approaches to outperform existing state-of-the-art algorithms.

Categories and Subject Descriptors

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

Keywords

Content-based Recommender Systems; Top-N recommendations; Implicit Feedback; Linked Open Data; DBpedia

1. INTRODUCTION

Recently, novel and more accessible forms of information coming from different open knowledge sources represent a rapidly growing piece of the big data puzzle.

Over the last years, more and more semantic data are published following the Linked Data principles[1], by connecting information referring to geographical locations, people, companies, book, scientific publications, films, music, TV and radio programs, genes, proteins, drugs, online communities, statistical data, and reviews in a single global data space, the *Web of Data* [2].

This information, interlinked with each other, forms a global graph called *Linked Open Data cloud*, whose nucleus is represented by `DBpedia`[2].

A fragment of the Linked Open Data cloud is depicted in Figure 1.

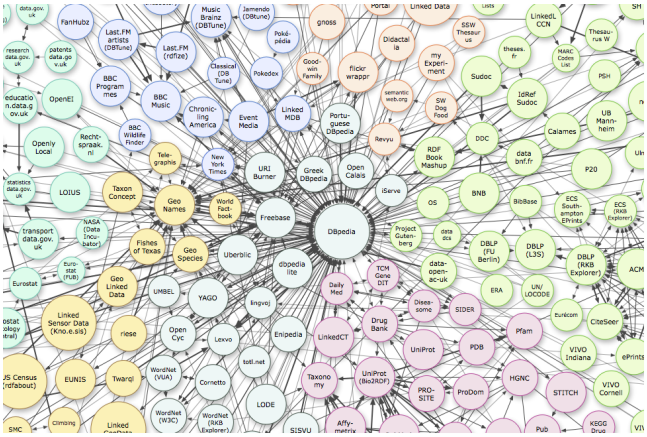

Figure 1: Fragment of the Linked Open Data cloud (as of September 2011).

Using open or pooled data from many sources, often combined and linked with proprietary big data, can help develop insights difficult to uncover with internal data alone [4], and can be effectively exploited by recommender systems to deal with classical problems of cold-start and sparsity.

On the other hand, the use of a huge amount of interlinked data poses new challenges to recommender systems researchers, who have to find effective ways to integrate such knowledge into recommendation paradigms.

This paper presents a preliminary investigation in which we propose and evaluate different ways of including several kinds of Linked Open Data features in different classes of recommendation algorithms. The evaluation is focused on the *top-N* recommendations task in presence of binary user feedback and cold-start situations.

[1] http://www.w3.org/DesignIssues/LinkedData.html

[2] http://dbpedia.org

CBRecSys 2014, October 6, 2014, Silicon Valley, CA, USA.

This paper extends our previous work carried out to participate to the Linked Open Data-enabled Recommender Systems challenge[3] [1], by presenting results for new tested algorithms, along with the various combinations of features. Results show the potential of Linked Open Data-enabled approaches to outperform existing state-of-the-art algorithms.

2. RELATED WORK

Previous attempts to build recommender systems that exploit Linked Open Data are presented in [17], where a music recommender system uses DBpedia to compute the *Linked Data Semantic Distance*, which allows to provide recommendations by computing the semantic distance for all artists referenced in DBpedia.

In that work, the semantics of the DBpedia relations is not taken into account, differently from the approach described in [5], where properties extracted from DBpedia and LinkedMDB [12] are exploited to perform a semantic expansion of the item descriptions, suitable for learning user profiles.

In [15], DBpedia is used to enrich the playlists extracted from a Facebook profile with new related artists. Each artist in the original playlist is mapped to a DBpedia node, and other similar artists are selected by taking into account shared properties, such as the genre and the musical category of the artist.

DBpedia is also used in [16] to capture the complex relationships between users, items and entities by extracting the paths that connect users to items, in order to compute recommendations through a *learning to rank* algorithm called *SPRank*. *SPRank* is a hybrid recommendation algorithm able to compute *top-N* item recommendations from implicit feedback, that effectively incorporates ontological knowledge coming from DBpedia (content-based part) with collaborative user preferences (collaborative part) in a graph-based setting. Starting from the common graph-based representation of the content and collaborative data models, all the paths connecting the user to an item are considered in order to have a relevance score for that item. The more paths between a user and an item, the more that item is relevant to that user.

The increasing interest in using Linked Open Data to create a new breed of content-based recommender systems is witnessed by the success of the recent Linked Open Data-enabled Recommender Systems challenge held at the European Semantic Web Conference (ESWC 2014). The contest consisted of 3 tasks, namely rating prediction in cold-start situations, top-N recommendation from binary user feedback, and diversity. Interestingly, top-N recommendation from binary user feedback was the task with the highest number of participants. The best performing approach was based on an ensemble of algorithms based on popularity, Vector Space Model, Random Forests, Logistic Regression, and PageRank, running on a diverse set of semantic features [1]. The performance of the single methods were aggregated using the Borda count aggregation strategy. Most of the techniques used in the contest are presented in this paper.

Similarly to the best performing approach, the second best performing one was based on the same ingredients [18]. Indeed, it combined different base recommenders, such as collaborative and content-based ones, with a non-personalized recommender based on popularity. Content-based strategies leveraged various features sets created from DBpedia. Additional Linked Open Data sources were explored, such as British Library Bibliography[4] and DBTropes[5], even though they did not provide meaningful features with respect to those derived from DBpedia. The results of the individual recommenders were combined using stacking regression and rank aggregation using Borda.

[3] challenges.2014.eswc-conferences.org/index.php/RecSys

3. METHODOLOGY

Section 3.1 describes the set of different features extracted from the Linked Open Data cloud, while Section 3.2 presents different kinds of recommendation algorithms, i.e. those based on vector space and probabilistic models, those based on the use of classifiers, and graph-based algorithms, which are fed in different ways by the features extracted from the Linked Open Data cloud.

3.1 Features extracted from the Linked Open Data cloud

The use of Linked Open Data allows to bridge the gap between the need of background data and the challenge to devise novel advanced recommendation strategies.

There are two main approaches to extract Linked Open Data features to represent items:

1. use of the *Uniform Resource Identifier* (URI)
2. use of *entity linking* algorithms.

The first approach directly extracts DBpedia properties for each item by using its *Uniform Resource Identifier* (URI). URIs are the standard way to identify real-world entities, and allow to define an entry point to DBpedia.

However, DBpedia provides a huge set of properties for each item, hence a proper strategy to select the most valuable ones is necessary. We could manually identify and select a subset of domain-dependent properties, or we could take into account a subset of the most frequent ones.

Referring to the book domain, in which we performed the evaluation, we selected the 10 properties in Table 1, which are both very frequent and representative of the specific domain.

Starting from these properties, further resources could be recursively added. For example, starting from a book, we could retrieve its author through the property
`http://dbpedia.org/ontology/author`
and then retrieve and link other resources by the same author, or other genres of works by the same author.

As an example, the resulting representation obtained for the book *The Great and Secret Show* is provided in Figure 2. The book is linked to its author (*Clive Barker*), to the genre (*Fantasy literature*), and to the Wikipedia categories (*British fantasy novels* and *1980s fantasy novels*). Furthermore, other books by Clive Barker are reported, such as *Books of Blood* and *Mister B. Gone*.

The second approach to extract LOD features uses *entity linking* algorithms to identify a set of Wikipedia concepts occurring in the item description. Next, those Wikipedia concepts can be easily mapped to the corresponding DBpedia nodes.

[4] http://bnb.data.bl.uk/

[5] http://skipforward.opendfki.de/wiki/DBTropes

Property	Description	Frequency
http://dbpedia.org/ontology/wikiPageWikiLink	Link from a Wikipedia page to another Wikipedia page. This property allows to take into account other Wikipedia pages which are somehow related.	523,321
http://purl.org/dc/terms/subject	The topic of a book.	38,627
http://dbpedia.org/property/genre	The genre of a book.	12,488
http://dbpedia.org/property/publisher	The publisher of a book.	9,798
http://dbpedia.org/ontology/author	The author of a book.	8,669
http://dbpedia.org/property/followedBy	The book followed by a specific book.	6,351
http://dbpedia.org/property/precededBy	The book preceded by a specific book.	5,978
http://dbpedia.org/property/series	The series of a book.	3,196
http://dbpedia.org/property/dewey	The Dewey Decimal library Classification.	1,930
http://dbpedia.org/ontology/nonFictionSubject	The subject of a non-fiction book (e.g.: history, biography, cookbook, ...).	966

Table 1: DBpedia properties selected for the book domain.

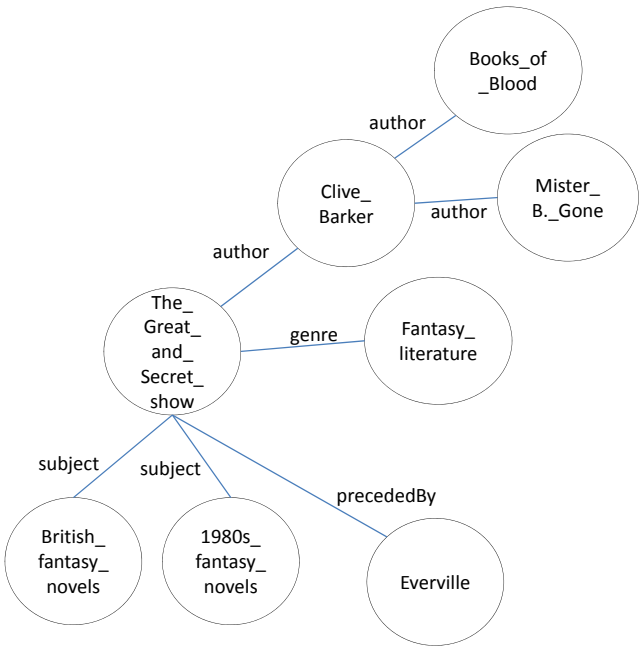

Figure 2: Properties of the book *The Great and Secret Show* by *Clive Barker*.

Several techniques can be adopted, such as `Explicit Semantic Analysis` [8] or `Tagme` [7].

In this work we adopt `Tagme`, that 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 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 provided in Figure 3. 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.

Figure 3: Tagme representation of the book *The Great and Secret Show* by *Clive Barker*.

All these features are used in different ways by the different recommendation algorithms presented in the following section. Details are reported in Section 4.2.

3.2 Recommendation Algorithms

We tested three different classes of algorithms for generating *top-N* recommendations, by using several combinations of features extracted from the Linked Open Data cloud.

3.2.1 *Algorithms based on the Vector Space and Probabilistic Models*

Most content-based recommender systems rely on simple retrieval models to produce recommendations, such as keyword matching or Vector Space Model (VSM).

VSM emerged as one of the most effective approaches in the area of Information Retrieval, thanks to its good compromise between effectiveness and simplicity. Documents and queries are represented by vectors in an n-dimensional vector space, where n is the number of index terms (words, stems, concepts, etc.).

Formally, each document is represented by a vector of weights, where weights indicate the degree of association between the document and index terms.

Given this representation, documents are ranked by computing the distance between their vector representations and the query vector. Let $D = \{d_1, d_2, ..., d_N\}$ denote a set of documents or corpus, and $T = \{t_1, t_2, ..., t_n\}$ be the dictionary, that is to say the set of words in the corpus. T is obtained by applying some standard natural language processing operations, such as tokenization, stopwords removal, and stemming. Each document d_j is represented as a vector in a n-dimensional vector space, so $d_j = \{w_{1j}, w_{2j}, ..., d_{nj}\}$, where w_{kj} is the weight for term t_k in document d_j. The

most common weighting scheme is the TF-IDF (Term Frequency-Inverse Document Frequency).

In content-based recommender systems relying on VSM, the *query* is the *user profile*, obtained as a combination of the index terms occurring in the items liked by that user, and recommendations are computed by applying a vector similarity measure, such as the *cosine* coefficient, between the user profile and the items to be recommended in the same vector space.

However, VSM is not able to manage either the latent semantics of each document or the position of the terms occurring in it. Hence, we proposed an approach able to produce a *lightweight* and *implicit* semantic representation of documents (items and user profiles). The technique is based on the *distributional hypothesis*, according to which *"words that occur in the same contexts tend to have similar meanings"* [11]. This means that the meaning of a word is inferred by analyzing its usage in large corpora of textual documents, hence words are semantically similar to the extent that they share contexts.

The gist of the technique is presented in [14], in which a novel content-based recommendation framework, called *enhanced Vector Space Model* (eVSM), is described. eVSM adopts a latent semantic representation of items in terms of *contexts*, i.e. a *term-context* matrix is adopted, instead of the classical term-document matrix adopted in the VSM. The advantage is that the *context* can be adapted to the specific granularity level of the representation required by the application: for example, given a word, its context could be either a single word it co-occurs with, a sentence, or the whole document.

The use of fine-grained representations of contexts calls for specific techniques for reducing the dimensionality of vectors. Besides the classical Latent Semantic Indexing, which suffers of scalability issues, more scalable techniques were investigated, such as Random Indexing [22], adopted in the eVSM model.

Random Indexing in an incremental method which allows to reduce a vector space by projecting the points into a randomly selected subspace of enough high dimensionality. The goal of using eVSM is to compare a vector space representation which adopts very few dimensions for representing items, with respect to a classical VSM.

As an alternative to VSM, we used the BM25 probabilistic model [19], one of the most dominant retrieval paradigm today. The ranking function for matching a query q (user profile) and an item I is:

$$R = \sum_{t \in q} \frac{n_t \cdot (\alpha + 1)}{n_t + \alpha \cdot (1 - \beta + \beta \frac{|I|}{avgdl})} \cdot idf(t) \qquad (1)$$

n_t is frequency of t in the item I, α and β are free parameters, $avgdl$ is the average item length, and $idf(t)$ is the IDF of feature t:

$$idf(t) = log \frac{N - df(t) + 0.5}{df(t) + 0.5} \qquad (2)$$

$df(t)$ is the number of items in which the feature t occurs, N is the cardinality of the collection.

For all the previous models we explicitly managed negative preferences of users by adopting the *vector negation* operator proposed in [23], based on the concept of *orthogonality* between vectors.

Several works generally rely on the Rocchio algorithm [21] to incrementally refine the user profiles by exploiting positive and negative feedback provided by users, even though the method needs an extensive tuning of parameters for being effective.

Negative relevance feedback is also discussed in [6], in which the idea of representing negation by subtracting an unwanted vector from a query emerged, even if nothing about *how much to subtract* is stated. Hence, vector negation is built on the idea of subtracting exactly the *right* amount to make the unwanted vector irrelevant to the results we obtain.

This removal operation is called *vector negation*, which is related to the concept of *orthogonality*, and it is proposed in [23].

3.2.2 Algorithms based on Classifiers

The recommendation process can be seen as a binary classification task, in which each item has to be classified as interesting or not with respect to the user preferences.

We learned classifiers using two algorithms, namely *Random Forests* (RF) [3] and *Logistic Regression* (LR).

RF is an ensemble learning method, combining different tree predictors built using different samples of the training data and random subsets of the data features. The class of an item is determined by the majority voting of the classes returned by the individual trees. The use of different samples of the data from the same distribution and of different sets of features for learning the individual trees prevent the overfitting.

LR is a supervised learning method for classification which builds a linear model based on a transformed target variable.

3.2.3 Graph-based Algorithms

We adopted *PageRank with Priors*, widely used to obtain an authority score for a node based on the network connectivity. Differently from PageRank, it is biased towards the preferences of a specific user, by adopting a non-uniform personalization vector to assign different weights to different nodes [13].

In order to run the PageRank, we need to represent data using a graph model. To this purpose, users and items in the dataset are represented as nodes of a graph, while links are represented by the positive users' feedback. The graph may be enriched in different ways, for example exploiting entities and relations coming from `DBpedia`: in this case the whole graph would contain nodes representing *users*, *items*, and *entities*, and edges representing items relevant to users, and relations between entities. This unified representation allows to take into account both *collaborative* and *content-based* features to produce recommendations.

In the classic PageRank, the prior probability assigned to each node is evenly distributed ($\frac{1}{N}$, where N is the number of nodes), while *PageRank with Priors* is biased towards some nodes, i.e. the preferences of a specific user (see Section 4.2).

4. EXPERIMENTAL EVALUATION

The goal of the experiments is to evaluate the contribution of diverse combinations of features, including those extracted from the Linked Open Data cloud, to the accuracy of different classes of recommendation algorithms.

The experiments that have been carried out try to answer to the following questions:

1. Which is the contribution of the Linked Open Data features to the accuracy of *top-N* recommendations algorithms, in presence of binary user feedback and cold-start situations?

2. Do the Linked Open Data-enabled approaches outperform existing state-of-the-art recommendation algorithms?

4.1 Dataset

The dataset used in the experiment is `DBbook`, coming from the recent Linked-Open Data-enabled Recommender Systems challenge. It contains user preferences retrieved from the Web in the book domain. Each book is mapped to the corresponding `DBpedia` URI, which can be used to extract features from different datasets in the Linked Open Data cloud.

The training set released for the *top-N* recommendation task contains 72,372 binary ratings provided by 6,181 users on 6,733 items. The dataset sparsity is 99.83%, and the distribution of ratings is reported in Table 2.

The test set contains user-item pairs to rank in order to produce a *top-5* item recommendation list for each user, to be evaluated using F1@5 accuracy measure.

4.2 Experimental setup

Each recommendation algorithm is fed by a diverse set of features.

Besides `TAGME` and `LOD` features, algorithms may also use `BASIC` features, i.e. number of positive, number of negative, and total number of feedbacks provided by users and provided on items, ratio between positive, negative and total number of feedbacks provided by users and provided on items and `CONTENT` features, obtained by processing book descriptions gathered from Wikipedia. A simple NLP pipeline removes stopwords, and applies stemming. For books not existing in Wikipedia, `DBpedia` abstracts were processed.

For all the methods, the 5 most popular items are assigned as liked to users with no positive ratings in the training set. Indeed, 5.37 is the average number of positive ratings for each user in the dataset (see Table 2).

Algorithms based on the Vector Space and Probabilistic Models.

Recommender systems relying on VSM and probabilistic framework index items using `CONTENT`, `TAGME` and `LOD` features, and use as *query* the *user profile* obtained by combining all the index terms occurring in the items liked by that user.

Items in the test set are ranked by computing the similarity with the user profile. For VSM and eVSM the cosine measure is adopted, while Equation 1 is used for the probabilistic model. According to the literature [20], parameters α and β are set to 1.6 and 0.75, respectively.

Algorithms based on Classifiers.

Classifiers based on Random Forests and Logistic Regression are trained with examples represented using `CONTENT`, `TAGME` and `LOD` features, and labeled with the binary ratings provided by users. The value of each feature is the number of times it occurs in each item, normalized in the [0,1] interval.

The LR classifier always includes `BASIC` features in the training examples, while these did not provide valuable results for RF.

The RF classifier used 1,500 trees to provide a good trade-off between accuracy and efficiency.

For Logistic Regression we adopted the implementation provided by Liblinear[6], while for Random Forests we adopted the implementation provided by the Weka library[7].

Top-N recommendations are produced by ranking items according to the probability of the class.

Graph-based Algorithms.

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` (see Section 3.1). 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
7. genres of the series the book belongs to.

This process adds thousands of nodes to the original graph. For this reason, we pruned the graph by removing nodes which are neither users nor books and having a total number of inlinks and outlinks less than 5. This graph eventually consisted of 340,000 nodes and 6 millions links.

The prior probabilities assigned to nodes depend on the users' preferences, and are assigned according to the following heuristics: 80% of the total weight is evenly distributed among items liked by users (0 assigned to disliked items), 20% is evenly distributed among the remaining nodes. We ran the algorithm with a damping factor set to 0.85.

We adopted the implementation of PageRank provided by the Jung library[8].

The PageRank computed for each node is used to rank items in the test set.

4.3 Results

Figure 4 shows the results of VSM and probabilistic models. The *paired t-test* is used for testing the significance.

The first interesting outcome is that the worst configurations are always obtained using `LOD` data alone, and the best ones always contain `TAGME` features. In detail, the best VSM configuration is obtained combining `TAGME` and `LOD` features, which is significantly better than using `LOD` features alone ($p < 0.0001$) and `CONTENT` alone ($p < 0.05$). The combination of `CONTENT` and `LOD` features outperforms the models using just `CONTENT` ($p < 0.01$) or just `LOD` features ($p < 0.001$).

[6] www.csie.ntu.edu.tw/~cjlin/liblinear/
[7] www.cs.waikato.ac.nz/ml/weka/
[8] jung.sourceforge.net

Property	Value
Statistics about users	
Avg. ratings provided by users	11.71 (5.37 positive, 6.34 negative)
# of users who provided only negative ratings	520 (8.41%)
# of users having a number of positive ratings below the avg.	3,804 (61.54%)
# of users having more negative than positive ratings	3,343 (54.09%)
Statistics about items	
Avg. ratings received by items	10.75 (4.93 positive, 5.82 negative)
# of items with no positive ratings	1,839 (27.31%)
# of items having a number of positive ratings below the avg.	6,447 (95.75%)
# of items having more negative than positive ratings	4,046 (60.09%)

Table 2: Distribution of ratings for the DBbook dataset.

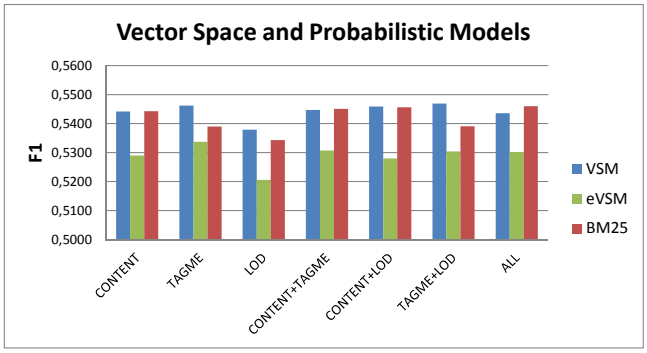

Figure 4: Results of the VSM and probabilistic models using different combinations of features.

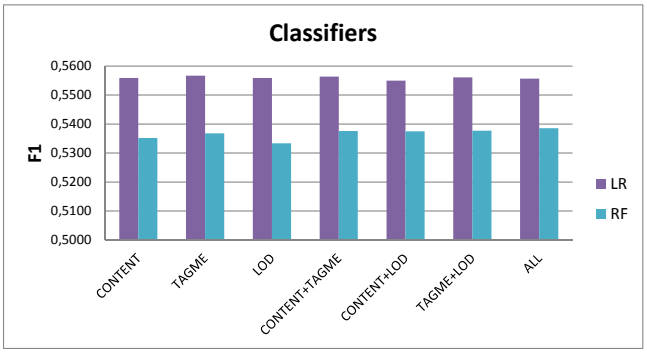

Figure 5: Results of the classifiers using different combinations of features.

The best configuration for eVSM adopts `TAGME` features alone, and is significantly better than all the configurations but the one combining `CONTENT` and `TAGME` features ($p = 0.13$). This could mean that the entity linking algorithm is able to select the most important features in the book descriptions, while `CONTENT` features introduce noise.

For BM25, the best configuration with `ALL` the features significantly outperforms all the others but the one combining `CONTENT` and `LOD` features ($p = 0.53$).

Surprisingly, there is no statistical difference between the best performing configuration for VSM and the best one for BM25.

A final remark is that eVSM performance is not comparable to the other methods, even though it is worth noting that it represents items using very low-dimensional vectors (dimension=500), compared to VSM, which uses vectors whose dimensionality is equal to the number of items (6,733).

Figure 5 presents the results obtained by the classifiers.

We note that Logistic Regression always outperforms Random Forests, and provides better results than the vector space and probabilistic models, regardless the set of adopted features.

The best result using Logistic Regression is obtained with `TAGME` features alone. This configuration significantly outperforms the one including `CONTENT` and `LOD` features ($p < 0.05$), while it is not different with respect to the other configurations. This is probably due to the high sparsity of the feature vector used to represent each training example (220,000 features).

Random Forests classifiers outperform eVSM, but they are worse than vector space and probabilistic models. The best result is obtained using `ALL` features. Since Random Forests classifiers are able to automatically perform feature selection, this was an unexpected result which deserves further investigations.

Finally, Figure 6 presents the results obtained by the PageRank with Priors algorithm.

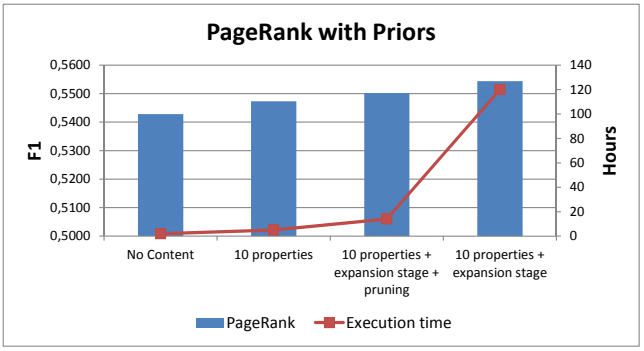

Figure 6: Results of the PageRank with Priors using different combinations of features.

When using PageRank with Priors, we observe the impact of the graph size on both the accuracy and execution time. Starting with a graph not including content information, we observe the worst performance and the lowest execution time (2 hours on an Intel i7 3Ghz 32Gb RAM - the algorithm

is performed for each user with different weights initially assigned to the nodes).

Enriching the graph with the 10 selected DBpedia properties leads to an improvement of accuracy ($p < 0.001$), and to a 5 hours execution time. Running the expansion stage and pruning of nodes as described in Section 4.2, the time needed to run the algorithm increases to 14 hours and produces a slight accuracy improvement ($p < 0.001$). Results using the graph with no pruning procedure are not different from the previous method ($p = 0.09$), but its time complexity is not acceptable. This call for a more efficient implementation of the algorithm.

To complete the empirical evaluation, we compare the best performing configuration of each algorithm in each class, with some state-of-the-art algorithms.

More specifically, we report the performance of *user-to-user* and *item-to-item collaborative filtering*, besides two non-personalized baselines based on *popularity* and *random* recommendations.

Furthermore, we report the results for two algorithms for *top-N* recommendations from implicit feedback: an extension of matrix factorization optimized for Bayesian Personalized Ranking (*BPRMF*) [9] and *SPRank* [16], able to exploit LInked Open Data knowledge bases to compute accurate recommendations.

Except for SPRank, we used the implementations available in MyMediaLite 3.10 [10], using the default parameters.

The analysis of results in Figure 7 unveils the difficulty of collaborative filtering algorithms to deal with the high sparsity of the dataset (99.83%), and with the high number of users who provided only negative preferences, or more negative than positive ratings. It is unexpected the better performance of *BPRMF* compared to *SPRank*, differently from previous results obtained on the MovieLens and Last.fm datasets [16]. It is also surprising the better performance of simple algorithms based on the vector space and probabilistic models with respect to matrix factorization.

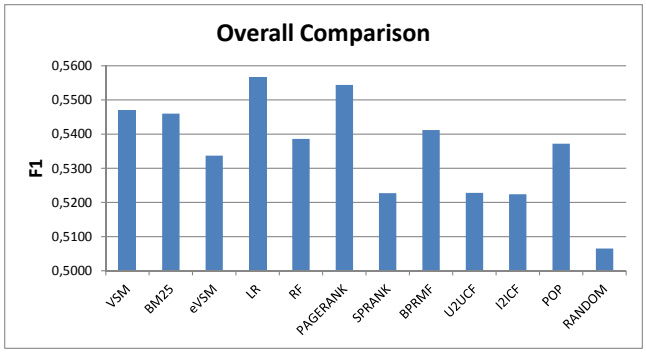

Figure 7: Comparison with other state-of-the-art approaches.

4.4 Discussion

The analysis of the previous results allows to conclude that TAGME and LOD features have the potential to improve the performance of several recommendation algorithms for computing *top-N* recommendations from binary user feedback.

However, in order to generalize our preliminary results, it is necessary to further investigate:

- the effect of different levels of sparsity on the recommendation accuracy: to this purpose, it is needed to assess the extent to which LOD features are able to improve the performance of recommendation algorithms for different levels of sparsity
- the accuracy on other datasets to generalize our conclusions: further experiments on different target domains are needed. Indeed, different item types, such as books, movies, news, songs have different characteristics which could lead to different results. Moreover, experiments on a much larger scale are needed
- the effect of the selection of domain-specific DBpedia properties to feed the recommendation algorithms: it is needed to assess the effect of the selection of specific sets of properties on the performance of the recommendation algorithms. Indeed, DBpedia contains a huge number of properties, and their selection could have a strong influence on the accuracy of the recommendation methods. Our preliminary experiments leverage 10 DBpedia properties which are both frequent and representative of the specific domain, but a subset of these properties, or a different set of features could lead to different results.

As future work, we will study the effect of enriching the graph-based representation with DBpedia nodes extracted from the Tagme entity linking algorithm.

Indeed, using entity linking to access DBpedia knowledge is innovative and avoids the need of explicitly finding URIs for items, a complex process which may hinder the use of the Linked Open Data. Hence, 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.

Furthermore, starting from the preliminary evaluation carried out in [1], we will thoroughly investigate the potential of using the wealth of relations of LOD features to produce not only accurate, but also *diversified* recommendation lists.

Acknowledgments

This work fulfils the research objectives of the project "VINCENTE - A Virtual collective INtelligenCe ENvironment to develop sustainable Technology Entrepreneurship ecosystems" (PON 02_00563_3470993) funded by the Italian Ministry of University and Research (MIUR).

5. REFERENCES

[1] P. Basile, C. Musto, M. de Gemmis, P. Lops, F. Narducci, and G. Semeraro. Aggregation strategies for linked open data-enabled recommender systems. In *European Semantic Web Conference (satellite Events)*, 2014.

[2] C. Bizer. The emerging web of linked data. *IEEE Intelligent Systems*, 24(5):87–92, 2009.

[3] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[4] M. Chui, J. Manyika, and S. V. Kuiken. What executives should know about open data. *McKinsey Quarterly, January 2014*, 2014.

[5] T. Di Noia, R. Mirizzi, V. C. Ostuni, and D. Romito. Exploiting the web of data in model-based recommender systems. In P. Cunningham, N. J. Hurley, I. Guy, and S. S. Anand, editors, *Proceedings of the ACM Conference on Recommender Systems '12*, pages 253–256. ACM, 2012.

[6] M. D. Dunlop. The effect of accessing nonmatching documents on relevance feedback. *ACM Trans. Inf. Syst.*, 15:137–153, 1997.

[7] P. Ferragina and U. Scaiella. Fast and Accurate Annotation of Short Texts with Wikipedia Pages. *IEEE Software*, 29(1):70–75, 2012.

[8] E. Gabrilovich and S. Markovitch. Wikipedia-based semantic interpretation for natural language processing. *Journal of Artificial Intelligence Research (JAIR)*, 34:443–498, 2009.

[9] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In G. I. Webb, B. Liu, C. Zhang, D. Gunopulos, and X. Wu, editors, *10th IEEE International Conference on Data Mining*, pages 176–185. IEEE Computer Society, 2010.

[10] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Mymedialite: a free recommender system library. In B. Mobasher, R. D. Burke, D. Jannach, and G. Adomavicius, editors, *Proceedings of the ACM Conference on Recommender Systems '11*, pages 305–308. ACM, 2011.

[11] Z. S. Harris. *Mathematical Structures of Language*. Interscience, New York„ 1968.

[12] O. Hassanzadeh and M. P. Consens. Linked movie data base. In C.Bizer, T. Heath, T. Berners-Lee, and K. Idehen, editors, *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009*, volume 538 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

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

[14] C. Musto. Enhanced vector space models for content-based recommender systems. In *Proceedings of the ACM Conference on Recommneder Systems '10*, pages 361–364. ACM, 2010.

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

[16] V. C. Ostuni, T. Di Noia, E. Di Sciascio, and R. Mirizzi. Top-n recommendations from implicit feedback leveraging linked open data. In Q. Yang, I. King, Q. Li, P. Pu, and G. Karypis, editors, *Proceedings of the ACM Conference on Recommender Systems '13*, pages 85–92. ACM, 2013.

[17] A. Passant. dbrec - Music Recommendations Using DBpedia. In *International Semantic Web Conference, Revised Papers*, volume 6497 of *LNCS*, pages 209–224. Springer, 2010.

[18] P. Ristoski, E. L. Mencia, and H. Paulheim. A hybrid multi-strategy recommender system using linked open data. In *European Semantic Web Conference (Satellite Events)*, 2014.

[19] S. E. Robertson, S. Walker, M. H. Beaulieu, A. Gull, and M. Lau. Okapi at TREC. In *Text REtrieval Conference*, pages 21–30, 1992.

[20] S. E. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

[21] J. Rocchio. Relevance Feedback Information Retrieval. In Gerald Salton, editor, *The SMART retrieval system - experiments in automated document processing*, pages 313–323. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[22] M. Sahlgren. An introduction to random indexing. In *Proc. of the Methods and Applications of Semantic Indexing Workshop at the 7th Int. Conf. on Terminology and Knowledge Engineering*, 2005.

[23] D. Widdows. 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*, pages 136–143, 2003.