Popular Books and Linked Data: Some Results for the ESWC'14 RecSys Challenge

Michael Schuhmacher, Christian Meilicke

Research Group Data and Web Science, University of Mannheim
{michael,christian}@informatik.uni-mannheim.de

Abstract. Within this paper we present our contribution to Task 2 of the ESWC'14 Recommender Systems Challenge. First we describe an unpersonalized baseline approach that uses no linked-data but applies a naive way to compute the overall popularity of the items observed in the training data. Then we describe an algorithm that makes use of several features acquired from DBpedia, like author and type, for item representation and comparison. Furthermore we leverage Wikipedia links and keywords found within the Wikipedia abstract. Item recommendations are generated by a mixture-model of Naive Bayes classifiers that have been learned on user neighborhood clusters with classifiers learned globally on training data. While our Linked-Data-based approach achieves an F_1 measure of 0.5607, the increase over the popularity baseline remains surprisingly low.

1 Introduction

Within this paper we describe the methods we developed for participating in Task 2 of the ESWC'14 Linked Open Data-enabled Recommender Systems Challenge[1]. For Task 2, the aim was a top-5 item recommendation per user, based on a training dataset with binary ratings. For each user, a relatively small set of books, on average 11.00 (min 0, max 20), to select from was already given. Assigning a score to each item-user pair defines an ordered list from which the top-5 ranked items for each user are interpreted as recommendations.

The training dataset consisting of 72,371 user-book-rating tuples was given, with 6,181 distinct users and 6,733 distinct books. The evaluation/test dataset contains the same 6,181 users, consequently for every user at least one training data recommendation was available. However, out of the 6,903 unique books in the test data, 939 have not been observed before in the training data, thus creating an item-cold-start situation for 1,964 item-user-pairs. While those item-user-pairs might be of special interest for a LOD-enabled system, this subset represents only a small fraction of the 67,989 user-item pairs in the evaluation dataset.

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

2 System Description

For better understanding the dataset and the potential of linked data for improving the recommendation process, we implemented both an unpersonalized baseline using no external knowledge and a machine learning approach exploiting item features derived from DBpedia. Our baseline does not make use of any external data, but uses only the information found within the given training dataset. It is based on the naive idea to recommend top-rated books. Our main system takes a very different approach as it relies heavily on DBpedia for item representation and uses supervised learning on the training data for making rating predictions. This system has participated as UniMannheim in Task 2 of the challenge.

2.1 Unpersonalized Baseline

To understand the benefit of exploiting Linked Open Data, we implemented a naive baseline that follows the simple idea to recommend books according to their overall rating. Our baseline is thus an unpersonalized method that produces the same recommendations for each user by computing a popularity score for each book. In particular, we computed for each book b the score

$$pop(b) = \#likes(b)/(\#likes(b) + \#dislikes(b) + 1)$$

where $\#likes(b)$ refers to the number of users that gave a positive rating for b and $\#dislikes(b)$ refers to the number of users that gave a negative rating to b. Note that we added $+1$ to the denominator to avoid that a book with no negative ratings and a low number of positive ratings achieves a high score. We refer to this method as the popularity baseline in the following.

For those 939 books that have not been observed in the training data, this method yields $pop(b) = 0$, i.e. such a book is always less preferred compared to any of the observed books. According to the results presented in [3] we would expect that such a method is clearly outperformed by any personalized approach or any other approach that uses external knowledge. However, we wanted to implement a naive baseline to better understand in the results of the methods discussed the next section.

2.2 Linked-Data-based Recommender

In contrast to the baseline, our Linked-Data-based recommender (LDR) makes prominent use of external information, namely item features obtained from DBpedia. The key components are (i) the item representation model employing features from DBpedia data, (ii) the naive Bayes classifier for rating prediction, and (iii) the user-neighborhood-based collaborative filtering for reducing data sparsity.

Item Features. In order to overcome the item sparsity and to be able to make predictions for the unobserved items, we opt to represent each item by a set

of multi-valued multinomial features. Given the gold standard mappings from book item ids to their corresponding DBpedia entities,[2] we opt to focus on the information available from DBpedia (Version 3.9). We manually chose the following predicates to be queried and added as features:[3]

- Genre: `dbo:literaryGenre`
- Wikipedia subjects: `dcterms:subject`
- DBpedia and Yago types: `rdf:type`
- Author(s): `dbo:author`, `dbo:writer`
- Book Series: `dbo:series`
- Publisher: `dbo:publisher`

For reducing sparsity, we furthermore expanded all retrieved Wikipedia categories by their immediate super-category via the `skos:broader` predicate and also add Wikipedia-links using `dbo:wikiPageWikiLink` (inspired by [1]).

Manual categories. The problem in using subjects, genres, and similar properties in the appropriate way is related to the fact that (i) there exist a waste amount of different values, (ii) these values are often scattered over different properties, and (iii) the values can be very specific (e.g., `High_fantasy_novels` instead of `fantasy`). To overcome these problems, we have created a list of 30 categories like `science fiction`, `fantasy`, `horror`, `philosophy`, and so on. To each of these categories we have assigned a simple regular expression (in most cases just the name of the category). Then we have parsed the abstract (`dbo:abstract`), the genre (`dbo:literaryGenre`, `dbp:genre`) and the subject (`dcterms:subject`) of each book and checked whether the pattern defined by the regular expression was identified. This resulted in a new aggregated feature with more coverage and a restricted value set.

Feature expansion. To overcome the problem of sparsely populated features, we computed similarity scores between values of the properties `dcterms:subject`, `dbo:literaryGenre` and `dbp:genre`. For each value pair s_1 and s_2 we computed the Dice similarity between all books labeled with s_1 and all books labeled with s_2. This way we detected a high similarity between, for example, the values `Literary_history` and `History_of_literature`. Given a feature value v, we added all those values v' for which the similarity between v and v' was higher than 0.2. In a similar way, we expanded the author feature by adding to a book all those authors with whom the original author ever wrote a book in co-authorship.

Rating Prediction Classifier. All item features were combined into one model with multinomial variables. We opt to predict ratings with a supervised machine learning approach in order to create the required, implicit item ranking. After initial tests with different established machine learning methods[4] (Naive Bayes,

[2] The challenge data contain some inconsistencies, as e.g. different items have the same DBpedia URI (384 duplicates), or the same title (319 duplicates). We opt explicitly to not fix those errors and work with the dataset as given.

[3] We abbreviate namespaces according to common rules (`http://www.prefix.cc`).

[4] For experiments we used the Weka 3.7.10 Java API with LibSVM 1.0.5, alternatingDecisionTrees 1.0.5, and bestFirstTree 1.0.3.

Support Vector Machine, Linear Regression, ADTrees) we decided to use a simple Naive Bayes classifier, primarily due to its robustness and good performance in terms of learning/training time. Even though we made several efforts in feature creation and expansion, as described above, it turned out that learning one classifier per user was not a successful approach. The most likely reason for that is the relatively low training instances count of 11.71 items per user as well as the high ratio of unrated to rated items per user of 1.2 (min 0, max 14, median 1). As a consequence, we created a mixture-model which we describe next.

Collaborative Filtering. As a per-user-based classifier was not successful, we followed the idea of user-neighborhood-based collaborative filtering (see e.g. [2]). For that purpose, we first compute user neighborhoods, i.e. clusters of varying size that aggregate a given user and all other users from the training data that have at least one common book in their ratings list. To account for different and multiple ratings, we compute a simple score

$$UserSim(u_1, u_2) = |\{b \mid r(u_1, b) = r(u_2, b)\}| - |\{b \mid r(u_1, b) \neq r(u_2, b)\}|$$

where $r(u, b)$ refers to the rating of user u for book b. Taking all user pairs with $UserSim(u_1, u_2) > 0$ into account, we obtain a neighborhood per user. However, to mitigate the effect of ratings sparsity, we also learned a global classifier for smoothing, and combined the scores of both classifiers, unweighted and linear, for those user neighborhoods being in the upper quartile Q_3 (on this dataset ≥ 64) of the neighborhood's user count. For users having a smaller number of users in their neighborhood, we use only the global classifier to predict ratings.

3 Results and Analysis

Evaluation of our systems was performed directly via the official web-service of the ESWC'14 Challenge. The submitted user-item-score list was treated as a ranked results set, and Recall@5 (R@5), Precision@5 (P@5), and F-measure@5 (F_1@5) was computed. Our results for different variations are shown in Table 1.

Results The first interesting observation is related to the good performance of the popularity baseline. With an an F_1 of 0.5583 the result for the Popularity Baseline are nearly as good as the results of our Linked-Data-based Recommender. Moreover, to our knowledge the best participating system achieved an F_1 score of 0.5715, which shows that we provide a rather strong baseline.

With the above described Linked-Data-based Recommender (LDR) configuration (challenge submission name UniMannheim), we achieve an F_1 of 0.5607. Each of the features described shows by itself only a marginal contribution to the overall performance, e.g. excluding the expansion of the DBpedia features causes the F_1 measure to decrease only to 0.5603. A similar observation holds for the manual category schema that contributes only 0.002 points to the F_1 score.

However, the different user aggregation methods (global, neighborhood, user) show a significant influence on the overall performance. Our initial approach to learn individually one classifier for each user yields only 0.5302 in F_1, when using

	Recall	Precision	F_1-measure
Popularity Baseline	0.4905	0.6484	0.5585
LD-based Recommender (LDR)	0.4938	0.6486	0.5607
LDR w/o Feature Expansion	0.4933	0.6483	0.5603
LDR w/o Manual Categories	0.4936	0.6485	0.5605
Global Classifier	0.4927	0.6476	0.5596
Neighborhood+Global Classifier	0.4977	0.6502	0.5638
Neighborhood Classifier	0.4806	0.6326	0.5462
User Classifier	0.4644	0.6177	0.5302

Table 1. Evaluation results on test data, computed by ESWC'14 RecSys webservice.

the same feature set as for the LDR system. In contrast, using the same global classifier for all users yields in 0.5596, thus showing better performance than the Popularity Baseline. The middle ground between both approaches, learning one classifier for each user neighborhood, has an F_1 of 0.5462. Using an equally weighted linear combination of the neighborhood classifiers and global classifier, we were able achieve an F_1 measure of 0.5638, which is the best result compared to all other settings. However, this setting was not used for our official submission, even though it is the more intuitive approach compared to our submission setting described above.

Conclusion. In summary, it was somehow surprising to us, that our unpersonalized baseline system, performed comparably well on Task 2 of the Recommender System Challenge. Furthermore, given that our LD-based solution differs significantly from our baseline approach, the marginal difference in F_1-measure of 0.0022 seems at first surprising. However, the key component of our LD-based recommender is in the end the global classifier learned on all user ratings – which is essentially the same idea that we follow with the unpersonalized popularity score of the baseline. In conclusion, it seems that, at least for our approach, the usage of LD was not significantly superior compared to a naive approach.

Acknowledgments

We would like to thank our colleagues Arnab Dutta and Johannes Knopp for their valuable contribution to our systems, as well as Orphee De Clercq and Robert Meusel for their support in understanding the data and technology used.

References

1. Di Noia, T., Mirizzi, R., Ostuni, V.C., Romito, D.: Exploiting the web of data in model-based recommender systems. In: Proc. of RecSys'12. pp. 253–256 (2012)
2. Herlocker, J., Konstan, J., Riedl, J.: An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. Information Retrieval 5(4), 287–310 (2002)
3. Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender systems: an introduction. Cambridge University Press (2011)