A Hybrid Multi-strategy Recommender System Using Linked Open Data

Petar Ristoski[1(✉)], Eneldo Loza Mencía[2], and Heiko Paulheim[1]

[1] Research Group Data and Web Science, University of Mannheim, Mannheim, Germany
{petar.ristoski,heiko}@informatik.uni-mannheim.de
[2] Knowledge Engineering Group, Technische Universität Darmstadt, Darmstadt, Germany
eneldo@ke.tu-darmstadt.de

Abstract. In this paper, we discuss the development of a hybrid multi-strategy book recommendation system using Linked Open Data. Our approach builds on training individual base recommenders and using global popularity scores as generic recommenders. The results of the individual recommenders are combined using stacking regression and rank aggregation. We show that this approach delivers very good results in different recommendation settings and also allows for incorporating diversity of recommendations.

Keywords: Linked Open Data · Hybrid recommender systems · Stacking

1 Overall Approach

We propose a hybrid, multi-strategy approach that combines the results of different base recommenders and generic recommenders into a final recommendation. A *base recommender* is an individual collaborative or content based recommender system, whereas a *generic recommender* makes a recommendation solely on some global popularity score, which is the same for all users. The approach has been evaluated on the three tasks of the *LOD-enabled Recommender Systems Challenge 2014* from the domain of book recommendations.[1] For base recommenders, we use two collaborative filtering strategies (item and user based), as well as different content-based strategies exploiting various feature sets created from DBpedia[2].

Generic Recommenders. We use different generic recommenders in our approach. First, the RDF Book Mashup dataset[3] provides the average score assigned

[1] 75,559 numeric ratings on 6,166 books (from 0–5, Task 1) and 72,372 binary ratings on 6733 books (Tasks 2 and 3), resp., from 6,181 users for training, and evaluation on 65,560 and 67,990 unknown ratings, resp. See http://challenges.2014.eswc-conferences.org/index.php/RecSys for details.
[2] http://dbpedia.org
[3] http://wifo5-03.informatik.uni-mannheim.de/bizer/bookmashup/

© Springer International Publishing Switzerland 2014
V. Presutti et al. (Eds.): SemWebEval 2014, CCIS 475, pp. 150–156, 2014.
DOI: 10.1007/978-3-319-12024-9_19

to a book on Amazon. Furthermore, DBpedia provides the number of ingoing links to the Wikipedia article corresponding to a DBpedia instance, and the number of links to other datasets (e.g., other language editions of DBpedia), which we also use as global popularity measures. Finally, SubjectiveEye3D delivers a subjective importance score computed from Wikipedia usage information.[4]

Features for Content-Based Recommendation. The features for content-based recommendation were extracted from DBpedia using the RapidMiner Linked Open Data extension [8]. We use the following feature sets for describing a book:

- All *direct types*, i.e., `rdf:type`, of a book[5]
- All *categories of a book*
- All *categories of a book including broader categories*[6]
- All *categories of a book's author(s)*
- All *categories of a book's author(s) and of all other books* by the book's authors
- All *genres of a book* and of all other books by the book's authors
- All *authors that influenced or were influenced* by the book's authors
- A bag of words created from the *abstract* of the book in DBpedia. That bag of words is preprocessed by tokenization, stemming, removing tokens with less than three characters, and removing all tokens less frequent than 3 % or more frequent than 80 %.

Furthermore, we created a *combined book's feature set*, comprising direct types, qualified relations, genres and categories of the book itself, its previous and subsequent work and the author's notable work, the language and publisher, and the bag of words from the abstract. Table 1 depicts the number of features in each set.

Besides DBpedia, we made an effort to retrieve additional features from two additional LOD sources: British Library Bibliography and DBTropes[7]. Using the RapidMiner LOD extension, we were able to link more than 90 % of the books to BLB entities, but only 15 % to DBTropes entities. However, the generated features from BLB were redundant with the features retrieved from DBpedia, and the coverage of DBTropes was too low to derive meaningful features. Hence, we did not pursue those sources further.

Recommender Strategies. For implementing the collaborative and content-based recommendation systems, we used the RapidMiner Recommendation Extension [5], which uses k-NN classification. We use $k = 80$ and cosine similarity for the base recommenders. The rationale of using cosine similarity is that,

[4] https://github.com/paulhoule/telepath/wiki/SubjectiveEye3D

[5] This includes types in the YAGO ontology, which can be quite specific (e.g., *American Thriller Novels*).

[6] The reason for not including broader categories by default is that the category graph is not a cycle-free tree, with some subsumptions being rather questionable.

[7] http://bnb.data.bl.uk/ and http://skipforward.opendfki.de/wiki/DBTropes

unlike, e.g., Euclidean distance, only common features influence the similarity, but not common absence of features (e.g., two books *not* being American Thriller Novels).

Furthermore, we train an additional recommender on the joint feature set, using Random Decision Trees (RDTs) [11].[8] RDTs generate k_1 decision trees with maximal depth k_2 and random attribute tests at the inner nodes. Each tree collects a distribution over the target variables at each of its leaf nodes by seeing the training data. E.g. for multilabel data, RDT's leaves collect the label distribution so that each RDT predicts for each test instance a distribution over the labels. These predictions are subsequently averaged over all trees in order to produce one single prediction. The predictions of several of such trees are then combined into a final prediction. RDTs provide a good tradeoff between scalability for large example sets and prediction accuracy (often outperforming SVMs).

For applying RDTs to the collaborative filtering data, we transformed the problem into a multilabel task: For each user we generated n different labels indicating each of the possible user ratings, i.e. $n = 5$ for task 1 and $n = 2$ for task 2. During training RDTs learn – for each known book/user combination – the mapping between the feature set of each book and the generated labels. Given an unknown book/user combination x, y, we are now able to estimate a distribution $P(i|\, x, y)$ over the different ratings i. The final predicted rating r is obtained by weighting the ratings $r = \sum_{i=0}^{5} i \cdot P(i|\, x, y)$ (task 1) or by computing the probability difference $P(1|\, x, y) - P(0|\, x, y)$ (task 2).

RDTs do not suffer from high dimensionality and sparseness as much as k-NN does, thus we have built $k_1 = 10$ trees with depth $k_2 = 10$ on the combined book's properties feature set, instead of individual RDTs on each feature set.[9]

2 Predicting Ratings and Top K Lists

For predicting ratings (task 1 in the challenge), we use all the recommendation algorithms discussed above for training a regression model in the range of $[0; 5]$. The results for the base and generic recommenders are shown in Fig. 1.

In order to create a more sophisticated combination of those recommenders, we trained a *stacking* model as described in [10]: We trained the base recommenders in 10 rounds in a cross validation like setting, collected their predictions, and learned a stacking model on the predictions. The results in Table 1 show that the stacked prediction outperforms the base and generic recommenders, with the RDT based stacking (with $k_1 = 500$ and $k_2 = 20$) slightly ahead of linear

[8] We used the implementation available at http://www.dice4dm.com/.

[9] In general, it holds that the higher k_1 and k_2 the better, since this increases the number of covered feature dimensions and the diversity of the ensemble. However, comparably small values of k_1 and k_2, around 10 or 20 and maximally 100, are sufficient according to experiments by Zhang et al. [11] and Kong and Yu [4]. In our experiments, we tried to find a good balance between computational costs and predictive quality, and we report the combination which we used for our final recommendations.

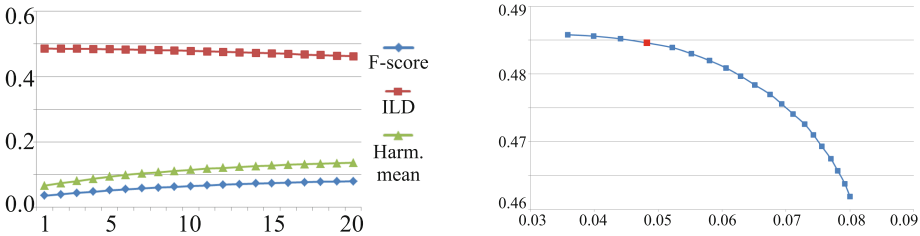
Table 1. Performances of the base and generic recommenders, the number of features used for each base recommender, and the performance of the combined recommenders

Recommender	#Features	Task 1		Task 2
		RMSE	LR β	F-Score
Item-based collaborative filtering	–	0.8843	+0.269	0.5621
User-based collaborative filtering	–	0.9475	+0.145	0.5483
Book's direct types	534	0.8895	-0.230	0.5583
Author's categories	2,270	0.9183	+0.098	0.5576
Book's (and author's other books') genres	582	0.9198	+0.082	0.5567
Combined book's properties	4,372	0.9421	+0.0196	0.5557
Author and influenced/influencedBy authors	1,878	0.9294	+0.122	0.5534
Books' categories and broader categories	1,987	0.939	+0.012	0.5509
Abstract bag of words	227	0.8893	+0.124	0.5609
RDT recommender on combined book's properties	4,372	0.9223	+0.128	0.5119
Amazon rating	–	1.037	+0.155	0.5442
Ingoing Wikipedia links	–	3.9629	+0.001	0.5377
SubjectiveEye3D score	–	3.7088	+0.001	0.5369
Links to other datasets	–	3.3211	+0.001	0.5321
Average of all individual recommenders	14	0.8824	–	–
Stacking with linear regression	14	0.8636	–	0.4645
Stacking with RDT	14	**0.8632**	–	0.4966
Borda rank aggregation	14	–	–	**0.5715**

regression, and both stacking approaches outperforming the baseline approach of averaging all recommenders' ratings.

To further analyze the contribution of each feature, we also report the β parameters found by linear regression. It can be observed that apart from the direct types, all base and generic recommenders contribute to the linear regression. A possible reason for that anomaly is that direct types and categories are rather redundant. Furthermore, we can see the benefit of using stacking approaches as the three generic recommenders with high RMSE are filtered out by the LR model.

For creating top k lists from binary ratings (task 2 in the challenge), we again trained regression models like for rating prediction, using a range of $[0; 1]$. The top k lists were then obtained by ranking by the predicted rating. As shown in Table 1, the base recommenders worked quite well, but the combination with linear regression delivered non-satisfying results. The reason is that the outcome of the base recommenders is not scaled equally for each user, but strongly depends on the user's total number of positive and negative ratings. This made it impossible to learn a suitable regression function.



(a) F-measure, ILD and their harmonic mean (y-axis) for different values of m

(b) ILD (y-axis) plotted against F-measure (x-axis)

Fig. 1. Trade-off between F-measure and diversity

However, we observed that despite being incompatible in scale, the base and generic recommenders delivered good *rankings* for each user. Thus, we performed an aggregation of the rankings produced by the different recommenders, using Borda's rank aggregation algorithm, which outperforms all the individual recommenders, as well as the stacking regression.

3 Creating Diverse Predictions

The final task in the challenge was to address *diversity* of predictions, i.e., trade off the accuracy of predictions, measured in F1 score, and their diversity, measured in intra-list diversity (ILD), both on a top k list. To address that trade-off, we followed a greedy top down approach which creates a ranking as for top k lists. First, we select the top m items from that list. Then, we process the list from position $m + 1$ on, adding each book that does not share author and categories with any of the books already on the list, until the list has k items.

The results are depicted in Fig. 1 for k = 20, selecting items from a list of the top 100 predictions. It can be observed that the F1 score gradually rises when using higher values of m, while the ILD drops. Although the harmonic mean is optimal for using simply the top 20 predictions (given the different orders of magnitude of F1 and ILD), we decided to submit the solution with $m = 4$ to the challenge.[10]

4 Related Work

The area of recommender systems has been extensively studied in the literature, resulting in a variety of techniques for performing recommendation, including content-based, collaborative, and hybrid techniques. However, only a handful of approaches exploit Linked Open Data to provide recommendations. Among the

[10] The reason is that the challenge uses the average rank w.r.t. F1 and ILD as a scoring function, which makes the selection of an optimal parameter strongly depend on the other participants' solutions. It turned out that $m = 4$ optimized our scoring.

earliest such efforts is *dbrec* [7], which is using DBpedia as a knowledge base to build a music content-based recommender system. Heitmann et al. [3] propose an open recommender system which utilize Linked Data to mitigate the new-user, new-item and sparsity problems of collaborative recommender systems.

More recent approaches [1,2,6,9] have shown that using data from the LOD cloud can improve the performances for both content-based and collaborative recommender systems, in various domains.

5 Conclusion and Outlook

In this paper, we have layed out a hybrid multi-strategy approach for linked data enabled recommender systems. We have shown that combining the predictions of different base recommenders is a feasible strategy, and that generic (i.e., non user specific) recommenders can be a useful ingredient.

In particular, our approach allows for the addition of new feature groups without interaction effects, and for the combination of different recommender strategies. By exploiting stacking regression, an optimal combination of different recommenders can be found automatically, however, for ranking-based problems, rank aggregation turned out to be the more promising strategy.

Acknowledgements. The work presented in this paper has been partly funded by the German Research Foundation (DFG) under grant number PA 2373/1-1 (Mine@LOD).

References

1. Di Noia, T., Mirizzi, R., Ostuni, V.C., Romito, D.: Exploiting the web of data in model-based recommender systems. In: Proceedings of the Sixth ACM Conference on Recommender Systems, RecSys '12, pp. 253–256, ACM. New York (2012)
2. Di Noia, T., Mirizzi, R., Ostuni, V.C., Romito, D., Zanker, M.: Linked open data to support content-based recommender systems. In: Proceedings of the 8th International Conference on Semantic Systems, I-SEMANTICS '12, pp. 1–8. ACM, New York (2012)
3. Heitmann, B., Hayes, C.: Using linked data to build open, collaborative recommender systems. In: AAAI Spring Symposium: Linked Data Meets Artificial Intelligence (2010)
4. Kong, X., Yu, P.S.: An ensemble-based approach to fast classification of multi-label data streams. In: CollaborateCom, pp. 95–104 (2011)
5. Mihelčić, M., Antulov-Fantulin, N., Bošnjak, M., Šmuc, T.: Extending rapidminer with recommender systems algorithms. In: RapidMiner Community Meeting and Conference (RCOMM 2012) (2012)
6. Ostuni, V.C., Di Noia, T., Mirizzi, R., Di Sciascio, E.: Top-n recommendations from implicit feedback leveraging linked open data. In: IIR, pp. 20–27 (2014)
7. 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)

8. Paulheim, H., Ristoski, P., Mitichkin, E., Christian, B.: Data mining with background knowledge from the web. In: RapidMiner World (2014)
9. Schmachtenberg, M., Strufe, T., Paulheim, H.: Enhancing a location-based recommendation system by enrichment with structured data from the web. In: Web Intelligence, Mining and Semantics (2014)
10. Ting, K.M., Witten, I.H.: Issues in stacked generalization. J. Artif. Intell. Res. **10**(1), 271–289 (1999)
11. Zhang, X., Yuan, Q., Zhao, S., Fan, W., Zheng, W., Wang, Z.: Multi-label classification without the multi-label cost. In: Proceedings of the 2010 SDM (2010)