Comparative Evaluation for Recommender Systems for Book Recommendations

Araek Tashkandi[1], Lena Wiese[2], Marcus Baum[3]

Abstract: Recommender System (RS) technology is often used to overcome information overload. Recently, several open-source platforms have been available for the development of RSs. Thus, there is a need to estimate the predictive accuracy of such platforms to select a suitable framework. In this paper we perform an offline comparative evaluation of commonly used recommendation algorithms of collaborative filtering. They are implemented by three popular RS platforms (LensKit, Mahout, and MyMediaLite) using the BookCrossing data set containing 1,149,780 user ratings on books. Our main goal is to find out which of these RSs is the most applicable and has high performance and accuracy on these data. We consider performing a fair objective comparison by benchmarking the evaluation dimensions such as the data set and the evaluation metric. Our evaluation shows the disparity of evaluation results between the RS frameworks. This points to the need of standardizing evaluation methodologies for recommendation algorithms.

Keywords: Recommender System, LensKit, Mahout, MyMediaLite, Book recommendations.

1 Introduction

In this age of Big Data, we face the problem of acute information overload of the available online data. Thus, it is difficult for users to find items matching their preferences. Many researchers focus on building tools to help users obtain personalized resources [Ga07]. One example of this qualified intelligent system is the recommender system (RS). Recommender systems have become a significant component in e-commerce systems since directing users to relevant content is increasingly important today. RSs have been applied in different application domains, such as entertainment, content, e-commerce and services. Some examples of these applications are Amazon.com (that recommends books, CDs and other products) and MovieLens (that recommends movies). One of the application areas where the RS shows its value is book recommendation, which is the focus of this paper. Recently, a large scale of open-source platforms has been available for the development of RSs. The selection of the foundational platform is an important step in developing the RS. Thus, there is a need to evaluate the quality in terms of predictive accuracy of such frameworks. In this paper we get inspired by [SB14], and we perform an offline comparative evaluation of commonly used recommendation algorithms of collaborative

[1] King Abdulaziz University, Faculty of Computing and Information Technology, 21589 Jeddah, Kingdom of Saudi Arabia, asatashkandi@kau.edu.sa
[2] Georg-August-University Goettingen, Institute of Computer Science, Goldschmidtstr. 7, 37077 Goettingen, Germany, wiese@cs.uni-goettingen.de
[3] Georg-August-University Goettingen, Institute of Computer Science, Goldschmidtstr. 7, 37077 Goettingen, Germany, marcus.baum@cs.uni-goettingen.de

filtering. They are implemented by three popular RS platforms (LensKit, Mahout, and MyMediaLite). We use the book ratings data set from BookCrossing (BX) from [Zi05]. The data set contains 278,858 users providing 1,149,780 ratings on 271,379 books. The evaluation is performed by internal evaluation methods (i.e. evaluation methods that are implemented by the frameworks). Our main goal is to find out which of these RSs is the most applicable and has high performance and accuracy on these data and in general for the book recommendation. Keep in mind the challenges of RSs evaluation as described by [He04] and [SB14] and the difficulties of such a result comparison due to the differences in evaluation and recommendation algorithms implementations. Therefore, in order to reach our goal, we mainly try to make a fair objective comparison of the recommendation accuracy of these RSs by considering the benchmarking of the evaluation dimensions such as the data set, data splitting, recommendation algorithms (including their parameters) and the evaluation method and metric. Our evaluation shows the disparity of evaluation results between the common RS frameworks even with benchmarking the evaluation dimensions.

2 Background

A RS learns about a user's needs and then proactively recommends information that meets his needs. Recommender systems are defined as "software agents that elicit the interests and preferences of individual users explicitly or implicitly and make recommendations accordingly" [XB07]. Three dimensions or models can describe the RS environment: users, data and application. In order to design a RS, the designer has to make choices that are related to the algorithms of the recommendation method, the system architecture and user profile. These choices are constrained by the environment models of the recommender. There are various techniques of how recommendations are generated by a RS. Depending on the information that the system gathers and the technique that is used to generate the item recommendations, the RSs can be classified into different types. The most common recommendation approaches are: content-based, collaborative filtering (CF), hybrid and knowledge-based approach [Ja11]. This paper discusses the CF recommendation approach. In the CF approach to predict the items that the current user will most probably like, the past behavior or opinions of the system's existing user community need to be exploited. Basically, its idea is based on "if users shared the same interests in the past, they will also have similar taste in the future" [Ja11]. It takes the matrix of user-item ratings as an input. It gives an output of numerical rating prediction of the degree that the current user likes or dislikes a specific item.

In order to evaluate a RS some aspects must be specified including the data set, the evaluation method and the performance metric:

- **Data Set:**
  Ekstrand et al. [ERK11] state that in order to evaluate a RS that is developed for a particular context, a data set which is relevant to that context is used. Herlocker et al. [He04] describe the properties that make a data set a best model for the tasks the RS is evaluated for. There are several publicly available data sets that are used for evaluating the RSs. These data sets are helpful in many cases: they provide a basis for

comparing the performance of a new algorithm against a known performance of an existing system, and in case of building a new system they can be used in preliminary testing. Some examples of these publicly available data sets are MovieLens, Jester, BookCrossing, and Netflix data sets.

- **Recommender System Quality Features:**
  There are some features that are considered when evaluating the RS and contribute to the system success and affect the user experience. The three commonly used RS properties from [Be13], [He04] and [SG11] are: (1) Prediction Accuracy where the RS with high accuracy recommends the most relevant items. (2) Coverage is the percentage of items in the data set that the RS is able to provide predictions for. (3) Diversity refers to the diverse features that the recommended items have. As stated by [SG11] the empirical evaluation of RSs has focused on evaluating the accuracy.

- **Evaluation Methods:**
  There are two designs or experimental settings to evaluate the RS and to compare several RSs [ERK11] and [GS09]. Online Evaluation is evaluating the RS while involving real users to use the system to perform real tasks. Offline Evaluation is based on the train-test setup in machine learning. It is performed by using a pre-collected data set of user ratings on items. By this data set we simulate the behavior of user interaction with a RS. Offline evaluation is commonly used in evaluating RSs since it is reproducible and easy to conduct and is not risky since it does not require the involvement of real users.

- **Evaluation Metrics for Performance Measurement:**
  In order to conduct a comparative evaluation for a set of candidate recommendation algorithms, some quality and performance features are measured by using evaluation metrics. We measure the prediction accuracy in evaluating the frameworks. The most popular metric that is used for evaluating accuracy of predicted ratings is Root Mean Squared Error (RMSE). RMSE measures the distance between the predicted preferences and the true preferences over items (i.e. the ones given by users in the test set). For the predicted rating $p_{u,i}$ of user u and item i, the true rating $r_{u,i}$ and data set size n it can be computed as follows [SG11]:

$$RMSE = \sqrt{\frac{1}{n}\sum_{u,i}(p_{u,i} - r_{u,i})^2} \tag{1}$$

3 Related Work

Researching evaluation of RS platforms, [Ek11] implement an evaluation of LensKit, and [SW12] perform an evaluation of Mahout. Said and Bellogin [SB14] perform a comparative evaluation of the three RS frameworks Mahout, LensKit and MyMediaLite. For each framework they perform an external controlled evaluation using a toolkit and an internal evaluation using the frameworks internal evaluation methods. They show disparity of evaluation results between the three recommendation frameworks both in the frameworks internal evaluations as well as in the external controlled evaluation.

The experimental work of evaluating the RS frameworks in this paper is based on [SB14]. They performed the evaluation to find if such a comparison is possible, while we perform a comparative evaluation of the frameworks to find out which of these RSs is the most applicable and has a high performance and accuracy on our used data and in general for book recommendations. Similar to [SB14], we use publicly available data for evaluating the RSs. However, our work differs from theirs in some aspects. First, their evaluation by framework internal evaluation methods was based on two different evaluation metrics since they state there is no common evaluation metric between the frameworks. They used Normalized Discounted Cumulative Gain (nDCG) to compare LensKit with Mahout and Root Mean Squared Error (RMSE) to compare LensKit with MyMediaLite. This results in an ambiguous cross-system evaluation of the three framework RSs. However, we identified RMSE as a common metric between the three frameworks which we use to compare all of them. Second, [SB14] show that the disparity of evaluation results occurs in both the external controlled evaluation as well as in internal frameworks evaluation; however, their internal framework evaluation was not by the same evaluation metric. Thus, we perform and focus only on an evaluation that uses the framework internal evaluation methods. Moreover, in their evaluation they use Movielens 100k data set, while we use the BX data set which is much bigger with higher sparsity. This shows the frameworks' performance, accuracy and capability in such a case using a unified evaluation metric.

4 Experiment and Evaluation

For evaluating the RS capabilities of the three frameworks, we use the internal evaluation methods. In order to perform a fair comparison, we benchmark the evaluation dimensions as will be described in this section. The experiment was run on a computer with Windows 10 Home (v.1511), 8 Gb RAM, Intel® Core(TM) i7-5500U 2.40GHz CPU, a 64-bit operating system and a x62-based processor. For Mahout and LensKit the Java JDK 1.7.0_79 is used. The LensKit evaluator groovy file was executed directly from Windows shell. Mahout was executed by NetBeans IDE 8.1 by creating a Java application in a Maven project and adding Mahout in the dependencies occurring in pom.xml file. MyMediaLite was run on a Mono version 4.2.3.

A natural starting point to evaluate recommender output is to measure the accuracy of prediction [ERK11]. To evaluate each of the three frameworks, the RMSE is computed and compared.

4.1 Data Set

As specified in [SB14], to ensure reproducibility and replicability, documenting the research, using open data, following standards and using best practice algorithms is mandatory. Therefore, we looked for publicly available data sets. As in this paper we focus on book recommendations; hence we select the BookCrossing (BX) data set. The BX data set is collected by a 4-week crawl during August-September 2004 [Zi05]. It is a collection of book ratings containing 278,858 users providing 1,149,780 ratings (explicit and implicit) on 271,379 books. In addition, the BX data set is also used in some RS evaluations such as [Zi05] and [GS09].

The BookCrossing data set includes three files or tables. The one we use in our evaluation experiment is the Book-Ratings table including user ID, ISBN and the associated ratings in a scale from 1 to 10 for explicit ones and 0 for implicit ones. In order to use this data set in our evaluation, it should be validated with the frameworks required format. After analyzing the data set we discovered many invalid ISBNs: such as there are around 1455 ISBNs with letters and around 885 ISBNs with symbols. In total, there are 2,340 invalid ISBNs. After cleaning the invalid ISBNs out, of 1,149,780 ratings we retain 1,147,440 ratings with valid ISBNs.

Evaluating the frameworks with using this cleaned and validated data set gives a high value of RMSE around 4 which means low prediction accuracy. [Zi05] describe the BookCrossing data set as extremely sparse (i.e. there are a lot of items and few ratings per user). Therefore, we had to condense the data to reduce the sparsity. Thus, we condense the data by removing the implicit ratings with a zero value proceeding in the same way as [GS09]. The implicit ratings are basically no ratings, so we ignore them. From the data set serving only valid ISBNs, 715,019 out of 1,147,440 are implicit ratings. After removing all the implicit ratings, the final data set contains 77472 users, 183744 items and 432,421 ratings. The frameworks split the data set during the evaluation process. LensKit and MyMediaLite evaluators provide cross validation data splitting. The Mahout evaluator lacks cross validation. Instead it creates hold-out partitions according to a set of run-time parameters.

4.2 Recommendation Frameworks

- **LensKit:** LensKit (LK) is an open source RS software that was developed by researchers at Texas State University and GroupLens Research at the University of Minnesota including contributions from developers around the world [Ek11]. It can be used for two purposes: for building a recommender application or for research. It is implemented in Java. The current version at the time of writing was 2.2.1. It provides ready to use implementations for these recommender algorithms: item-based CF, user-based CF, matrix factorization and slope-one.
- **Mahout:** Apache Mahout (AM) from [Fo16], is an Apache Software Foundation project of a well-known machine learning tool. It is an open source library containing several machine learning algorithms. It mainly focuses on the following three areas of machine learning: recommender engines collaborative filtering, classification and clustering. Similar to LensKit, Mahout is developed in Java. At the time of writing it reached version 0.10.1. The algorithms provided by Mahout are available as non-distributed (i.e. executed by a single machine) as well as distributed (i.e. MapReduce, executed in parallel in large clusters of thousands of nodes). In our scenario, we use the non-distributed case for the comparison purpose with the other RS frameworks. For the collaborative filtering algorithms, Mahout implements these algorithms: user-based CF, item-based CF and matrix factorization.

- **MyMediaLite:** MyMediaLite (MML) from [Ga11], was developed at Hildesheim University. At the time of writing, it had reached version 3.11. It is an open source software that provides a library of RS algorithms for the Common Language Runtime (CLR, often called .NET). It is developed in C and it can be run on different platforms, such as Linux, Windows and Mac OS X supported by Mono. It supports ready to use implementations for the two common scenarios in collaborative filtering: rating prediction and item prediction from positive-only feedback, which are combined with various recommender methods for each.

4.3 Algorithmic Details

The main goal of this paper is a fair comparative evaluation of three selected frameworks. Thus, to achieve this goal we benchmarked our comparison by controlling the evaluation dimensions, by using the same data set as explained in the previous section, and by selecting the same recommendation methods, algorithms and metrics. The common recommendation approach implemented by the RS frameworks is the collaborative filtering. They implement memory-based CF and model-based CF methods. A full list of the common methods within the frameworks is given in [SB14].

- **Memory-based CF Algorithms:** The rating matrix is held in memory and is directly used for generating the recommendation. The recommendations are issued based on the relationship between the queried item and the rest of the rating matrix. The common implemented methods by the frameworks are neighborhood-based methods [Ja11]:
  A **User-based CF algorithm** predicts the rating to which degree a specific user *u* will like a certain item *p* by getting the rating matrix as input and identifying the nearest neighbors k or the users that had similar preferences to the active user *u*. Then it gives out the numerical rating prediction of an item *p* that has not been seen yet by the active user, based on the neighbor users' ratings of *p*. Similar to the user-based CF, the **Item-based CF algorithm** takes the ratings matrix as an input and it identifies the nearest neighbors k or the items that have similar rating compared to the current item *p* that we want to compute the prediction for. Then it gets the active user's rating of these similar items to compute his rating for item *p*, which is the average of them. For computing the similarity, Pearson correlation and Cosine similarity are the similarity methods that can be used [ERK11].

- **Model-based CF Algorithms:**
  The ratings matrix is first processed offline and at the run time the precomputed model makes predictions. It uses data mining and machine learning algorithms to find patterns in the training data. Then it builds the model that can be fit on the test data. The rating prediction and the recommendations are issued based on the built model. The common implemented method by the frameworks is the matrix factorization Singular Value Decomposition (SVD). It is inferred from item rating patterns which is used to characterize both items and users by vectors of factors. An

item is recommended when a high correspondence between the item and user factors exists [SB14].

5 Result and Discussion

5.1 User-based CF algorithm

While evaluating the user-based CF algorithm, we use both of the similarity methods Pearson and Cosine with various neighborhood sizes k. Table 1a shows the evaluation results in terms of RMSE, generated by the frameworks of the user-based CF algorithm with the Pearson correlation method. In general, we observe that the LensKit (LK) prediction accuracy outperforms the Apache Mahout (AM) with different neighborhood sizes by 1.4% of the difference between the averages of the RMSE of both frameworks. However, with the neighborhood size k of 5, the AM gives a lower RMSE than the LK by 0.2% for the same neighborhood size, and lower by 0.2% for the lowest RMSE of the LK. Table 1b shows the evaluation results in terms of RMSE, generated by the frameworks of the user-based CF algorithm with the Cosine similarity method. The difference between the accuracy predictions of the frameworks is not as high as in the previous case. Furthermore, the AM outperforms the LK by 0.02%; the average of the AM's RMSE is 1.9234 and the average of the LK's RMSE is 1.9457. However, there is no big difference between the lowest RMSE of the AM, which is 1.854 and of the LK, which is 1.943. Consequently, we conclude that LK performs better than AM in the user-based CF algorithm. MyMediaLite (MML) was not able to complete the user-based CF evaluation, since it requires too much memory to finish the process. In [SB14] the authors experienced the same problems.

Various neighborhood sizes are tested to identify the best neighbors to consider and to discard the remaining users. However, the best neighborhood size for the accuracy differs between the frameworks. For instance, the best neighborhood size in the user-based CF algorithm with the Pearson method of the AM is 5, whereas in the LK is 10.

(a) User-based CF with Pearson correlation

Parameters	RMSE of LK	RMSE of AM
k=5	1.942345021	1.734362382
k=10	1.938357458	2.14859848
k=30	1.94569808	3.436938497
k=50	1.943045692	3.642906765
k=100	1.941659419	3.686270042
k=200	1.944997373	3.697794514
k=400	1.944071748	3.697794514
k=600	1.94546677	3.697794514
k=1000	1.943604582	3.697794514
k=1500	1.944357718	3.697794514

(b) User-based CF with Cosine similarity

Parameters	RMSE of LK	RMSE of AM
k=5	1.949359429	1.854495873
k=10	1.94548228	2.123300997
k=30	1.94519585	1.9428385
k=50	1.945271313	1.974455281
k=100	1.944991072	1.900317615
k=200	1.947228137	1.903636759
k=400	1.945185463	1.888266977
k=600	1.944815753	1.886360028
k=1000	1.943080086	1.880583559
k=1500	1.946471742	1.880583559

Tab. 1: Results of evaluating user-based CF

5.2 Item-based CF algorithm

For evaluating the item-based CF algorithm we use both similarity methods Pearson and Cosine. As shown in Table 2, AM has the lowest RMSE and is outperforming the LK in the Cosine similarity method by 0.4%.

Algorithm	Parameters	RMSE of LK	RMSE of AM
IBPea	no parm	1.953229033	2.240947847
IBCos	no parm	1.962735886	1.579909495

Tab. 2: Results of evaluating item-based CF with Pearson correlation and with Cosine similarity

Similar to the user-based evaluation, we were not able to evaluate the item-based CF in MyMediaLite. We can relate the MML problem to the sparsity problem of the data we used. Thus, we conclude the MML is not advisable if the used data is too sparse. In general, the problem of sparsity data does not only affect the RS performance (e.g. it requires too much memory or time), it also has an impact on the accuracy of the recommendation algorithm, as observed by [He04]. Since we use a low density and high sparsity data set, we conjecture that the computation performance and the prediction accuracy of the CF recommendation algorithms are affected. On a denser data set the three frameworks might perform better.

5.3 Model-based CF Algorithm

In addition, we compare the three frameworks in terms of run time of building and evaluating the algorithm SVD. We experienced that the LK is the fastest and the MML is the slowest, as shown in Table 3. Even though the MML is the weakest with the sparse data in the user-based and the item-based CF algorithms, it gives the highest accuracy in the SVD algorithm evaluation.

SVD method takes two parameters: 1) the number of latent factors or features (f) the users and items are characterized by, and 2) the number of iterations (i) for training and computation:

Parameters	RMSE of LK	Time	RMSE of AM	Time	RMSE of MML	Time
f=50, i=10	1.926852438	8min	2.294968976	15min	1.720527	45min
f=50, i=20	1.925003523	10min	2.416660616	27min	1.69042	2h

Tab. 3: Results of evaluating Singular Value Decomposition (SVD)

5.4 Discussion of Results

To summarize, after evaluating the three RS frameworks using the sparse BookCrossing data set, we observe that the evaluation results of LK's RMSE values are more consistent in the three recommendation algorithms, and in general in most of the cases its RMSE values are lower than the ones of AM. AM's RMSE values fluctuate, but the lowest RMSEs are better than the ones of LK. LK's performance in terms of the run time is the fastest. In addition, we observe that AM performs in the Cosine similarity better than in the Pearson correlation method. By using this sparse data we conclude that not all the RS frameworks are able to deal equally with too sparse data. Comparing to the other frameworks, the MML is the weakest in dealing with sparse data sets in the user-based and the item-based CF algorithms. It requires too much memory to finish the process and it takes the longest time to finish the SVD algorithm. Nevertheless, it gives the lowest RMSE in the SVD algorithm comparing to the other RSs. However, our presented result is limited to our used data set, which has sparsity problems that affect the prediction accuracy and the performance of the frameworks.

Even though we benchmarked the comparison dimensions such as the used data set and the algorithms with the parameters, there were difficulties to fairly compare the evaluation of the three RSs. As we mentioned, most of the RS literature points to the challenges of RS evaluation. The reason lies in the performance discrepancies and the disparity of the evaluation results of the same settings and algorithms with its parameters between the recommendation frameworks. For instance, our comparative evaluation of the SVD algorithm with the same parameters gives a 72% difference in the RMSE of the AM and the MML, and a 24% difference in the RMSE of the LK and the MML. Moreover, there is a higher than 100% difference in the RMSE of evaluating LK's and AM's user-based CF with the Pearson correlation method with the same neighborhood size parameter k of 200. Similarly, Said and Bellogin [SB14] encounter this problem of disparity of the evaluation results between the recommendation frameworks. They linked it to the differences in the implementations of the evaluation and recommendation algorithms in the frameworks.

6 Conclusion

For implementing a Recommender System (RS) by using open-source platforms, the most suitable platform has to be chosen after evaluating the considered frameworks. In this paper we perform an offline comparative evaluation of commonly used recommendation algorithms of collaborative filtering that are implemented by three popular RS platforms (Apache Mahout (AM), LensKit (LK) and MyMediaLite (MML)) using the BookCrossing (BX) book ratings data set. The performance and the accuracy of the implemented RSs differ based on the circumstances of the use case and may differ when evaluating them by using other data sets with different characteristics. Even though we used identical settings, we encountered difficulties of making a fair comparative evaluation of the three RSs. The reason for this is the performance discrepancy and the disparsity of the evaluation results. Many related research articles have discussed the RS evaluation. However, the effective evaluation of RSs is still challenging since 2004 [He04]. There is no practical rule for evaluating the RSs and the recommendation algorithms. Thus, we need a development of more standardized evaluation methodologies for RSs and for recommendation algorithms.

References

[Be13] Beel, J.; Langer, S.; Genzmehr, M.; Gipp, B.; Breitinger, C.; Nürnberger, A.: Research Paper Recommender System Evaluation: A Quantitative Literature Survey. In: Proceedings of the Workshop on Reproducibility and Replication in Recommender Systems Evaluation (RepSys) at the ACM Recommender System Conference (RecSys). ACM, pp. 15–22, 2013.

[Ek11] Ekstrand, M.; Ludwig, M.; Konstan, J.; Riedl, J.: Rethinking The Recommender Research Ecosystem: Reproducibility, Openness, and LensKit. In: Proceedings of 5th ACM Conference on Recommender Systems RecSys. ACM, pp. 133–140, 2011.

[ERK11] Ekstrand, M.; Riedl, J.; Konstan, J.: Collaborative Filtering Recommender Systems. Foundations and Trends in Human–Computer Interaction, 4(2):81–173, 2011.

[Fo16] Apache Mahout: Scalable Machine Learning and Data Mining, The Apache Software Foundation. https://mahout.apache.org/.

[Ga07] Gao, F.; Chunxiao, X.; Xiaoyong, D.; Shan, W.: Personalized Service System Based on Hybrid Filtering for Digital Library. Tsinghua Science and Technology. Tsinghua University Press (TUP), 12(1):1–8, 2007.

[Ga11] Gantner, Z.; Rendle, S.; Freudenthaler, C.; Schmidt-Thieme, L.: Mymedialite: A Free Recommender System Library. In: Proceedings of 5th ACM Conference on Recommender Systems, RecSys '11. ACM, pp. 305–308, 2011.

[GS09] Gunawardana, A.; Shani, G.: A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. Journal of Machine Learning Research, 10:2935–962, 2009.

[He04] Herlocker, J.; Konstan, J.; Terveen, L.; Riedl, J.: Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems TOIS, 22(1):5–53, 2004.

[Ja11] Jannach, D.; Zanker, M.; Felfernig, A.; Friedrich, G.: Recommender Systems: An Introduction. Cambridge University Press, New York, 2011.

[SB14] Said, A.; Bellogin, A.: Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks. In: Proceedings of The 8th ACM Conference on Recommender Systems RecSys'14. ACM, pp. 129–136, 2014.

[SG11] Shani, G.; Gunawardana, A.: Evaluating Recommendation Systems. Recommender Systems Handbook. Springer, pp. 257–97, 2011.

[SW12] Seminario, C.; Wilson, D.: Case Study Evaluation of Mahout as a Recommender Platform. In: Workshop on Recommendation Utility Evaluation: Beyond RMSE (RUE 2012), Proceedings of the 6th ACM Conference on Recommender Engines (RecSys). ACM, pp. 45–50, 2012.

[XB07] Xiao, B.; Benbasat, I.: Ecommerce Product Recommendation Agents: Use, Characteristics, and Impact. MIS Quarterly, 31(1):137–209, 2007.

[Zi05] Ziegler, C.; Sean, M.; Konstan, J.; Lausen, G.: Improving Recommendation Lists Through Topic Diversification. In: Proceedings of The 14th International World Wide Web Conference WWW2005. ACM, pp. 22–32, 2005.