CF4CF: Recommending Collaborative Filtering algorithms using Collaborative Filtering

Tiago Cunha
Faculdade de Engenharia da Universidade do Porto
Porto, Portugal
tiagodscunha@fe.up.pt

Carlos Soares
Faculdade de Engenharia da Universidade do Porto
Porto, Portugal
csoares@fe.up.pt

André C.P.L.F. de Carvalho
Universidade de São Paulo, ICMC
São Carlos, São Paulo, Brasil
andre@icmc.usp.br

ABSTRACT

As Collaborative Filtering becomes increasingly important in both academia and industry recommendation solutions, it also becomes imperative to study the algorithm selection task in this domain. This problem aims at finding automatic solutions which enable the selection of the best algorithms for a new problem, without performing full-fledged training and validation procedures. Existing work in this area includes several approaches using Metalearning, which relate the characteristics of the problem domain with the performance of the algorithms. This study explores an alternative approach to deal with this problem. Since, in essence, the algorithm selection problem is a recommendation problem, we investigate the use of Collaborative Filtering algorithms to select Collaborative Filtering algorithms. The proposed approach integrates subsampling landmarkers, a data characterization approach commonly used in Metalearning, with a Collaborative Filtering methodology, named CF4CF. The predictive performance obtained by CF4CF using benchmark recommendation datasets was similar or superior to that obtained with Metalearning.

CCS CONCEPTS

• **Information systems → Recommender systems**; *Data mining*; • **Computing methodologies → Machine learning**;

KEYWORDS

Collaborative Filtering, Metalearning, Label Ranking

ACM Reference Format:
Tiago Cunha, Carlos Soares, and André C.P.L.F. de Carvalho. 2018. CF4CF: Recommending Collaborative Filtering algorithms using Collaborative Filtering. In *Twelfth ACM Conference on Recommender Systems (RecSys'18), October 2–7, 2018, Vancouver, BC, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3240323.3240378

1 INTRODUCTION

Most research in the algorithm selection problem for Collaborative Filtering (CF) [23] have used Metalearning (MtL) [1, 4–6, 10, 12, 16]. The problem is modeled using a set of features (i.e., metafeatures), describing the problem domain, and the performance obtained by a group of algorithms according to a specific measure. This creates a metadataset. Afterwards, learning algorithms are applied to the metadataset, inducing a metamodel, which can be used to predict the best algorithms for a new problem. However, the design of suitable metafeatures is a hard problem. This is especially difficult in the CF problem, where there is no clear separation between independent and dependent variables. So far, several metafeatures based on statistical and/or information-theoretical measures [1, 5, 7, 10, 12, 16] and landmarking [6] have been proposed, with relevant results. However, the merits of using metafeatures continue to be questioned, since it is difficult to understand what they mean and how can they contain useful information for the algorithm selection problem. Hence, this study investigates another approach, which does not explicitly use metafeatures to induce a metamodel.

The proposed approach, named CF4CF, refers to the application of CF algorithms to select the best CF algorithms for a new problem. It considers the traditional users and items to be datasets and algorithms, respectively. The recommendation performance of all algorithms on a particular dataset are leveraged and converted into ratings. Next, a proper rating matrix can be built using performance data only. Thus, a CF algorithm can be used to create a metamodel, which can be used to predict the best ranking of algorithms for a new recommendation problem. Specifically in the prediction step, when no data is available regarding the algorithm performance, CF4CF uses subsampling landmarkers (performance estimations on a data sample extracted from the original dataset) to obtain initial ratings. This strategy allows CF4CF to deal with the cold start problem in algorithm selection. CF4CF then predicts the remaining ratings and converts the outcome into a ranking of algorithms.

As far as the authors know, CF4CF is the first attempt to use CF algorithms to recommend CF algorithms. Also, this study allows to objectively compare the approach proposed by [7] with CF4CF. This is only possible since (1) the experimental procedure leverages the same CF datasets, algorithms and evaluation measures and (2) the metalevel evaluation procedure is the same, except for the algorithms and training and test metadata, i.e. either ratings or metafeatures. We compare the metalevel accuracy and its impact on the baselevel accuracy for both algorithm selection strategies, showing that CF4CF is a good alternative for algorithm selection.

This document is organized as follows: Section 2 presents the related work on Algorithm Selection for CF; Section 3 presents the core contribution in this work: CF4CF, while Section 4 explains the experimental procedure. In Section 5, the proposed approach is evaluated and the results discussed. Finally, Section 6 presents the main conclusions and future work tasks.

RecSys '18, October 2–7, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-5901-6/18/10…$15.00
https://doi.org/10.1145/3240323.3240378

2 RELATED WORK

Although the use of MtL for CF has been previously investigated [1, 10, 12, 16], the proposed approaches have limited scope: they use a small number of datasets, recommendation algorithms and metafeatures. An extensive overview of their positive and negative aspects can be seen in a recent survey [8]. The design of metafeatures, which is the most important challenge in Mtl [2, 3, 22, 25], has recently been systematized [19]. It proposes a systematic approach for metafeature extraction, based on three elements: object o, function f and post-function pf. The framework applies the function f to every possible object o. The outcome(s) of f is(are) processed by the post-function pf to derive the metafeatures. Thus, any metafeature can be represented as: $\{o.f.pf\}$ [19].

Recent work has applied the systematic framework in the CF scope [5]. The objects used in the framework are CF's rating matrix R, and its sets of rows U and columns I. The functions f used to characterize these objects are: original ratings (*ratings*), count the number of elements (*count*), mean value (*mean*) and sum of values (*sum*). The post-functions pf are maximum, minimum, mean, standard deviation, median, mode, entropy, gini, skewness and kurtosis. Additionally, it includes 4 metafeatures: the number of users, items, ratings and the matrix sparsity. The results have shown that the most important metafeatures are: *nusers*, *R.ratings.kurtosis*, *R.ratings.sd*, *I.count.kurtosis*, *I.count.min*, *I.mean.entropy*, *I.sum.skewness*, *sparsity*, *U.mean.min*, *U.sum.entropy*, *U.sum.kurtosis*, *U.mean.skewness*. As an example, *R.ratings.kurtosis* represents the kurtosis of the distribution of all ratings in matrix R.

Furthermore, another type of MtL approach for CF has been introduced [6], which uses the performance of CF algorithms on data samples as the metafeatures: subsampling landmarkers [18]. This means that for each CF dataset, random samples are extracted. Then, CF algorithms are trained on these samples and their performance assessed using different metrics. The outcome is a subsampling landmarker for each pair algorithm/evaluation measure.

3 CF4CF

This work introduces a novel approach to the CF algorithm selection problem, CF4CF. The procedure is presented in Figure 1.

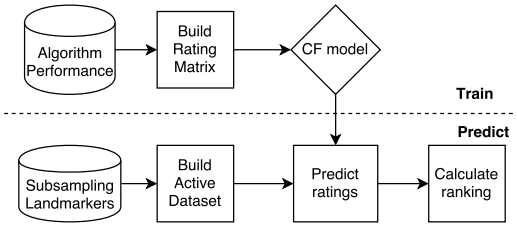

Figure 1: Overview of the CF4CF procedure.

This procedure has two main steps: train and predict. The training step obtains data from CF algorithm performance, builds a rating matrix and trains a CF model. The prediction step leverages subsampling landmarkers to create the initial ratings of the active dataset. It is then submitted to the CF model to obtain predictions. Afterwards, the final ranking of CF algorithms is calculated.

3.1 Building the Rating Matrix

Recall that CF requires three elements: users, items and ratings. As this work aims at recommending CF algorithms for CF datasets, the natural adaptation is to consider users and items as datasets and algorithms, respectively. Hence, to build the rating matrix $R^{D \times A}$, we consider the set of datasets D where each dataset $d_i \in D$ and the set of algorithms A where each algorithm $a_j \in A$. To complete the matrix, it is necessary to provide the available ratings. However, in the algorithm selection problem there is not an explicit assignment of ratings by each dataset to the algorithms. To solve this limitation, we model the preferences using the performance of algorithms on the datasets. The idea is to leverage how good the algorithm is for a particular dataset as its preference.

The proposed approach works by converting the rankings into ratings. Such conversion is required since CF models cannot work with ranking data. However, since ratings have the ability to provide different degrees of preference to each element in a list of items in a similar way to the way ranking operates, then these are the obvious choice. Furthermore, this conversion to a native CF data type allows to use any CF algorithm as the metamodel. Formally, consider a ranking of algorithms $R_{d_i} = (a_j)_{j=1}^{M}$ for a specific dataset d_i. This ranking is created by sorting the algorithms in decreasing order of performance. To convert the ranking R_{d_i} into a specific ratings scale $S \in [s_{min}, s_{max}]$, the following transformation f is applied to each position j:

$$f(R_{d_i}, j) = \frac{(S_{max} - S_{min})(M - j)}{M - 1} + S_{min} \quad (1)$$

Notice that this transformation is an combination of an inverse function (to state that high ratings must be assigned to the algorithm of lower ranking value) and a linear rescaling function (to adapt the ordered values to a specific ratings scale). The rating values are then $R_{d_i, a_j} = f(R_{d_i}, j)$. The matrix is completed by converting all rankings of algorithms for all datasets.

3.2 Training the CF model

The R matrix represents the rankings of a set of algorithms $a_j, j = 1 \ldots A$ on a set of CF problems $d_i, i = 1 \ldots D$. This rating matrix can, in principle, be trained using any CF algorithm to generate recommendations of algorithms for new datasets. The difference to standard MtL approaches is that now no metafeatures are needed.

However, the difference to traditional CF rating matrices concerns sparsity. Many preferences cannot be measured in CF problems, meaning that matrices are usually very sparse. However, standard MtL approaches to algorithm recommendation usually run all algorithms $a_j, j = 1 \ldots A$ on all CF problems $d_i, i = 1 \ldots D$l, thus generating a complete rating matrix. This means that some algorithms, e.g. Matrix Factorization (MF), which depend missing values to make predictions, cannot be used.

If a complete matrix is available, a simple solution is to apply such an algorithm to a sample from it, thus artificially creating missing values. This enables the use of algorithms such as MF, at the cost of losing some information contained in the ignored values. A different - and quite positive - perspective is to assume that the matrix is not complete to start with. This not only makes it possible to apply any CF algorithm but it also reduces the computational cost of generating the matrix, as it is no longer necessary to run all

algorithms on all CF problems. In other words, it is theoretically possible that CF4CF enables good performance with less information than what is required by standard MtL, translating into significant savings in computational resources. We investigate this hypothesis, by varying the parameter $N_{ratings}$, which refers to the number of ratings sampled by dataset to build the matrix.

3.3 Building the Active Dataset

After the CF model is induced, it can be used in the prediction step. For that purpose, the ratings based on the performance of some algorithms can be used to estimate the ratings of the remaining ones, and, thus, provide recommendations. However, in the context of algorithm recommendation, it is reasonable to assume that there is no performance estimate for any algorithm. In this case, CF4CF cannot properly work, since it would have no data to be used as input by the CF model. This issue is parallel to the cold start problem in traditional CF. Here we propose to deal with this problem by using subsampling landmarkers (SL). Thus, in order to build the active dataset representation, our method replaces the performance estimation of the algorithms on the complete dataset with SL (i.e. performance estimates obtained on samples of the data), applying the same rating conversion procedure described earlier. Formally, let us consider the complete ranking of the subsampling landmarkers $SL_{d_i} = (a_j)_{j=1}^{M}$ for a specific dataset d_i. Since we aim to use some of these values to serve as initial ratings for the CF model, we first sample the ranking SL_{d_i}. Considering how the number of ratings provided has the potential to directly affect the performance of CF models, it is important to understand the effect of sampling different amounts of ratings. We address this issue by using a parameter $N_{SL} \in [1, ...M-1]$ in our experiments. Finally, the sampled ranking SL_{d_i} is converted into ratings $R_{SL_{d_i}}$, also using Equation 1.

3.4 Turning Predicted Ratings into a Ranking

After obtaining the active dataset representation $R_{SL_{d_i}}$, it is possible to use the previously trained CF model to produce the predictions for the remaining algorithms, represented as $\hat{R}_{d_i}$. Recall that CF algorithms only consider items to be eligible for recommendation if the active user has not provided any feedback to them yet. Hence, in CF4CF, only algorithms for which no SL is calculated (and posteriorly provided to the CF model as rating) can be considered for prediction. However, the algorithm selection problem requires a complete ranking of algorithms to be predicted. To deal with this issue, we propose to aggregate the predictions with the initial ratings. Hence, the full ratings predicted are provided by $R_{d_i} =< \hat{R}_{d_i}, R_{SL_{d_i}} >$. To convert the ratings into rankings, these are sorted in decreasing order then ranked.

4 EXPERIMENTAL SETUP

4.1 Baselevel

The baselevel component deals with the traditional CF problem, being exactly the same for CF4CF and standard MtL. Here, three dimensions are considered: datasets, algorithms and evaluation measures. The experiments use 38 datasets from different domains, namely Amazon Reviews, BookCrossing, Flixter, Jester, MovieLens, MovieTweetings, Tripadvisor, Yahoo! and Yelp. Table 1 summarizes the characteristics of all domains and datasets used.

The CF algorithms used are variations of MF methods: BPRMF [20], which performs a pairwise classification task, optimizing AUC using Stochastic Gradient Descent (SGD); WBPRMF [20], which is a variation of BPRMF that includes a sampling mechanism that promotes low scored items; SMRMF [28], which is another variation of BPRMF, but it replaces the optimization formula in SGD by a soft margin ranking loss inspired by SVM classifiers; WRMF [14] which uses ALS (Alternating Least Squares) instead of SGD and introduces user/item bias to regularize the process; and lastly the baseline algorithm MostPopular which ranks items by how often they have been seen in the past. Since these algorithm deal with a Top-N recommendation problem, they are evaluated with NDCG (to assess ranking accuracy) and AUC (to evaluate classification accuracy) using 10-fold cross-validation. To prevent bias towards any algorithm, hyperparameter optimization was not used.

4.2 Metalevel

CF4CF uses only algorithm performance as input data. While the results obtained from the baselevel are used as training data, the prediction stage requires to calculate subsampling landmarkers. SL are obtained from a random sample of 10% of the original instances. These samples are submitted to the same baselevel evaluation procedure to obtain performance estimates. For the MtL approach, each dataset is simply described by the state of the art metafeatures [5] (Section 2). Algorithm performance is used to create rankings of algorithms to be used as targets for this predictive procedure. Thus, MtL is addressed using Label Ranking (LR) [15, 26]. Recall that CF4CF is designed to use any CF algorithm. However, in order to provide the fairest comparison possible between MtL and CF, this work uses two algorithms with the same bias: user-based CF [21] and kNN for LR [24], both based on Nearest Neighbours. These algorithms are referred to as CF4CF and KNN-LR, respectively. The baseline is Average Rankings.

Evaluation of algorithm selection results is done at two levels: meta-accuracy and impact on the baselevel performance. While the first aims to assess how similar are the predicted and real rankings of algorithms, the second investigates how the algorithms recommended by the metamodels actually perform in terms of recommendation prediction. To assess the meta-accuracy, this work uses the ranking accuracy measure Kendall's Tau and leave-one-out cross-validation. The impact at the baselevel is assessed by the average performance on a specific baselevel evaluation measure for different thresholds t. These thresholds refer to the number of algorithms considered from the predicted ranking. Hence, if $t = 1$, only the performance of the first algorithm is used; if $t = 2$, then the best performance of the first 2 algorithms is used, and so on.

5 EXPERIMENTAL RESULTS

5.1 Rating Matrix Sparsity

The first analysis focuses on understanding the effect of parameter $N_{ratings}$, which represents the number of ratings sampled by dataset to build the matrix. For such, different matrices were created by sampling the complete matrix and then used to induce CF4CF models. The results of the variation of the threshold in terms of

Table 1: Summary of the datasets used in the experiments. Values within square brackets indicate lower and upper bounds in a specific characteristic. In this table, k and M stand for thousands and millions, respectively.

Domain	Dataset(s)	#Users	#Items	#Ratings	Ref.
Amazon	App, Auto, Baby, Beauty, CD, Clothes, Food, Game, Garden, Health, Home, Instrument, Kindle, Movie, Music, Office, Pet, Phone, Sport, Tool, Toy, Video	[7k - 311k]	[2k - 267k]	[11k - 574k]	[17]
Bookcrossing	Bookcrossing	8k	29k	40k	[32]
Flixter	Flixter	15k	22k	813k	[31]
Jester	Jester1, Jester2, Jester3	[2.3k - 2.5k]	[96 - 100]	[61k - 182k]	[11]
Movielens	100k, 1m, 10m, 20m, latest	[94 - 23k]	[1k - 17k]	[10k - 2M]	[13]
MovieTweetings	RecSys2014, latest	[2.5k - 3.7k]	[4.8k - 7.4k]	[21k - 39k]	[9]
Tripadvisor	Tripadvisor	78k	11k	151k	[27]
Yahoo!	Movies, Music	[613 - 764]	[4k - 4.6k]	[22k - 31k]	[29]
Yelp	Yelp	55k	46k	212k	[30]

Kendall's Tau ranking accuracy for NDCG and AUC metatargets are presented in Figure 2.

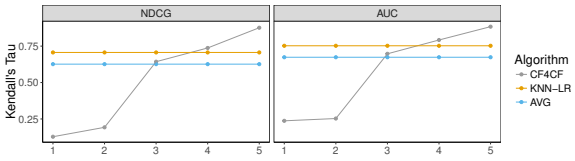

Figure 2: Ranking accuracy for different $N_{ratings}$.

According to the results, CF4CF performs similar or better than the baseline and standard MtL for $N_{ratings} = 4$. Thus, CF4CF is able to provide good recommendations using only 4 ratings per baselevel dataset. However, the results also show that CF4CF is only better than MtL for $N_{ratings} = 5$, meaning the full rating matrix is the only to consistently beat the selected MtL approach.

5.2 Meta-accuracy

This analysis assesses the effect of the number of sampled landmarkers (N_{SL}) in the CF4CF overall predictive performance. This experiment aims to understand how many SL are required to have competitive performance against MtL. Once again, the results of the variation of the threshold in terms of Kendall's Tau for both metatargets are presented in Figure 3.

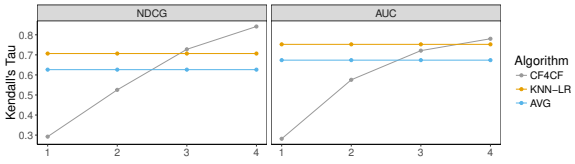

Figure 3: Ranking accuracy for different N_{SL}.

The results show that CF4CF performs better than the baseline for $N_{SL} = 3$, for both NDCG and AUC metatargets. However, its performance is similar to MtL only when $N_{SL} = 3$ for NDCG and $N_{SL} = 4$ for AUC. Furthermore, CF4CF outperforms standard MtL for NDCG when $N_{SL} = 4$. Thus, CF4CF is a suitable alternative to this MtL approach, when 4 SL are used to characterize the active dataset.

5.3 Impact on the baselevel performance

The impact of CF4CF on the baselevel performance is presented in Figure 4. The results in this figure are for $N_{SL} = 4$. Notice that the performances presented are dependent on the baselevel scale - this means that 0% and 100% refer to the minimum and maximum amount of baselevel performance.

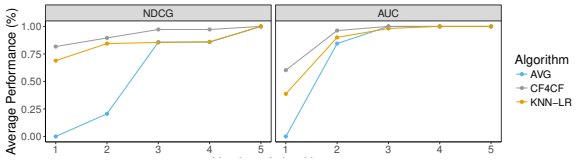

Figure 4: Impact on the baselevel performance.

The experimental results show that CF4CF outperforms both the baseline and standard MtL for $t \in \{1, 2, 3, 4\}$ and $t \in \{1, 2\}$ regarding NDCG and AUC, respectively. This means CF4CF makes better predictions than the competing approaches for the first thresholds in each problem. Thus, the results show CF4CF's predictions yield better baselevel performances for higher positions in the ranking.

6 CONCLUSIONS

We introduce a novel algorithm selection approach, CF4CF, which follows a CF approach to recommend rankings of the most suitable CF algorithms for a new dataset. The procedure uses the algorithm performance as rating information to induce a metamodel and uses subsampling landmarkers converted into ratings to predict the most suitable CF algorithms for a new dataset. According to the experimental results, several conclusions can be drawn: (1) CF4CF is better at predicting rankings of CF algorithms than standard MtL, (2) the CF algorithms recommended by CF4CF have higher impact on the baselevel performance than those recommended by standard MtL and (3) subsampling landmarkers can provide good initial ratings. Considering that CF4CF uses only performance data and subsampling landmarkers to tackle the algorithm selection problem, we have shown that this research direction is a suitable path for CF algorithm selection. Future work directions include: to investigate alternatives to leverage data for training and testing, further extend the experimental setup to other recommendation problems and algorithms, to study the tradeoff between accuracy and time required to create metafeatures and to leverage both metafeatures and ratings in a hybrid CF algorithm selection solution.

Acknowledgments. This work is financed by the Portuguese funding institution FCT - Fundação para a Ciência e a Tecnologia through the PhD grant SFRH/BD/117531/2016. The work is also financed by European Regional Development Fund (ERDF), through the Incentive System to Research and Technological development, within the Portugal2020 Competitiveness and Internationalization Operational Program within project PushNews (POCI-01- 0247-FEDER-0024257). Lastly, the authors would also like to acknowledge the support from Brazilian funding agencies (CNPq and FAPESP) and IBM Research and Intel.

REFERENCES

[1] Gediminas Adomavicius and Jingjing Zhang. 2012. Impact of data characteristics on recommender systems performance. *ACM Management Information Systems* 3, 1 (2012), 1–17.
[2] Pavel Brazdil, Christophe Giraud-Carrier, Carlos Soares, and Ricardo Vilalta. 2009. *Metalearning: Applications to Data Mining* (1 ed.). Springer Publishing.
[3] Pavel Brazdil, Carlos Soares, and Joaquim da Costa. 2003. Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time. *Machine Learning* 50, 3 (2003), 251–277.
[4] Tiago Cunha, Carlos Soares, and André C.P.L.F. Carvalho. 2017. Metalearning for Context-aware Filtering: Selection of Tensor Factorization Algorithms. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17)*. ACM, New York, NY, USA, 14–22.
[5] Tiago Cunha, Carlos Soares, and André C.P.L.F. de Carvalho. 2016. Selecting Collaborative Filtering algorithms using Metalearning. In *ECML-PKDD*. 393–409.
[6] Tiago Cunha, Carlos Soares, and Andre C.P.L.F. de Carvalho. 2017. Recommending Collaborative Filtering algorithms using subsampling landmarkers. In *Discovery Science*. 189–203.
[7] Tiago Cunha, Carlos Soares, and André C.P.L.F. de Carvalho. 2018. A Label Ranking approach for selecting rankings of Collaborative Filtering algorithms. In *ACM Symposium on Applied Computing*. 1393–1395.
[8] Tiago Cunha, Carlos Soares, and André C.P.L.F. de Carvalho. 2018. Metalearning and Recommender Systems: A literature review and empirical study on the algorithm selection problem for Collaborative Filtering. *Information Sciences* 423 (2018), 128–144.
[9] Simon Dooms, Toon De Pessemier, and Luc Martens. 2013. MovieTweetings: a Movie Rating Dataset Collected From Twitter. In *CrowdRec at RecSys 2013*.
[10] Michael Ekstrand and John Riedl. 2012. When Recommenders Fail: Predicting Recommender Failure for Algorithm Selection and Combination. *ACM RecSys* (2012), 233–236.
[11] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. 2001. Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Information Retrieval* 4, 2 (2001), 133–151.
[12] Josephine Griffith, Colm O'Riordan, and Humphrey Sorensen. 2012. Investigations into user rating information and accuracy in collaborative filtering. In *ACM SAC*. 937–942.
[13] GroupLens. 2016. MovieLens datasets. (2016). http://grouplens.org/datasets/movielens/
[14] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *IEEE International Conference on Data Mining*. 263 – 272.
[15] Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. 2008. Label ranking by learning pairwise preferences. *Artificial Intelligence* 172, 16-17 (2008), 1897–1916.
[16] Pawel Matuszyk and Myra Spiliopoulou. 2014. Predicting the Performance of Collaborative Filtering Algorithms. In *Web Intelligence, Mining and Semantics*. 38:1–38:6.
[17] Julian McAuley and Jure Leskovec. 2013. Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text. In *ACM Conference on Recommender Systems*. 165–172.
[18] Bernhard Pfahringer, Hilan Bensusan, and Christophe Giraud-Carrier. 2000. Meta-Learning by Landmarking Various Learning Algorithms. In *International Conference on Machine Learning*. 743–750.
[19] Fábio Pinto, Carlos Soares, and João Mendes-Moreira. 2016. Towards automatic generation of Metafeatures. In *PAKDD*. 215–226.
[20] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. 452–461.
[21] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-Based Collaborative Filtering Recommendation Algorithms. In *International Conference on World Wide Web*. 285–295.
[22] Floarea Serban, Joaquin Vanschoren, and Abraham Bernstein. 2013. A survey of intelligent assistants for data analysis. *Comput. Surveys* V, 212 (2013), 1–35.
[23] Yue Shi, Martha Larson, and Alan Hanjalic. 2014. Collaborative Filtering beyond the User-Item Matrix. *Comput. Surveys* 47, 1 (2014), 1–45.
[24] Carlos Soares. 2015. labelrank: Predicting Rankings of Labels. (2015). https://cran.r-project.org/package=labelrank
[25] Joaquin Vanschoren. 2010. *Understanding machine learning performance with experiment databases*. Ph.D. Dissertation. Katholieke Universiteit Leuven.
[26] Shankar Vembu and Thomas Gärtner. 2010. Label ranking algorithms: A survey. In *Preference Learning*. 45–64.
[27] Hongning Wang, Yue Lu, and ChengXiang Zhai. 2011. Latent Aspect Rating Analysis Without Aspect Keyword Supervision. In *ACM SIGKDD*. 618–626.
[28] Markus Weimer, Alexandros Karatzoglou, and Alex Smola. 2008. Improving Maximum Margin Matrix Factorization. *Machine Learning* 72, 3 (2008), 263–276.
[29] Yahoo! 2016. Webscope datasets. (2016). https://webscope.sandbox.yahoo.com/
[30] Yelp. 2016. Yelp Dataset Challenge. (2016). https://www.yelp.com/dataset_challenge
[31] R. Zafarani and H. Liu. 2009. Social Computing Data Repository at ASU. (2009). http://socialcomputing.asu.edu
[32] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving Recommendation Lists Through Topic Diversification. In *Proceedings of the 14th International Conference on World Wide Web*. 22–32.