Greedy Ensemble Selection for Top-N Recommendations

Tobias Vente[1], Zainil Mehta[1], Lukas Wegmeth[1] and Joeran Beel[1]

[1]Intelligent Systems Group, University of Siegen, Germany

Abstract

Despite the pivotal role ensembling played in the success of BellKors' Pragmatic Chaos Team in winning the Netflix Prize challenge in the early 2000s, it never became a standard practice in recommender systems. In contrast, ensembling, particularly greedy ensemble selection, has become a standard practice in machine learning to enhance performance compared to a single model. Despite the success of greedy ensemble selection in classification and regression tasks, it has not been adapted for top-n prediction tasks. Hence, in this study, we aim to analyze the potential of greedy ensemble selection to boost the performance of recommender system models for top-n prediction tasks. We adapt the concept of greedy ensemble selection for top-n prediction tasks, train and optimize ten factorization- and neighborhood-based models on five datasets, and compare the performance of the ensemble to that of the individual models. Our experiments reveal that greedy ensemble selection always performs better than a single model and enhances performance by an average of 8.8% on NDCG@5, 8.6% on NDCG@10, and 16.3% on NDCG@20 compared to the single best model.

Keywords

Ensembling, Recommender Systems, Algorithm Selection, Automatic Algorithm Selection

1. Introduction

Ensembling played a pivotal role for *BellKors' Pragmatic Chaos Team*, enhancing their recommender system to win the Netflix Prize challenge in the early 2000s [1, 2]. Despite its success in the competition, ensembling did not become a standard practice in the field of recommender systems. Today, mainly hybrid recommender systems rely on ensembling trained on different data or to aggregate predictions [3, 4]. Thereby, often combining collaborative filtering with content-based models to cancel out the weaknesses of individual models like the cold-start problem [5].

In comparison, in machine learning, ensembling, particularly greedy ensemble selection, is a standard practice [6], enhancing performance by as much as 37% in best-case scenarios and improving robustness [7]. Moreover, in automated machine learning, ensembling significantly enhances performance to the extent that some tools prioritize ensembling over further hyperparameter optimization [8].

However, despite the success of greedy ensemble selection for regression and classification, it has never been adapted for top-n ranking prediction tasks [6]. While ensembling has proven effective in machine learning, it remains a largely overlooked approach in the field of recommender systems. In recommender systems, researchers continue to debate whether the field makes progress, yet the focus primarily remains on continuously developing more sophisticated algorithms [9]. Instead of implementing a new, more complex recommender system algorithm, we want to focus on ensembling already existing algorithms.

Therefore, we want to analyze the potential of greedy ensemble selection for top-n prediction tasks and answer the question: **RQ:** How does the ensembling of factorization- and neighborhood-based models impact performance and robustness compared to a single optimized model?

In this work, we adapt greedy ensemble selection for top-n prediction tasks to assess its potential for enhancing ranking performance. We focus on ten fast and easy-to-train factorization- and neighborhood-based models. We then evaluate the ensemble output of these ten models on five datasets using *NDCG@k*, with k set to 5, 10, and 20, aiming to quantify the performance and robustness improvements compared to single optimized models.

Our contribution is the implementation of greedy ensemble selection for top-n ranking prediction tasks, along with a comprehensive analysis of its performance impact and robustness compared to single optimized models. Our results indicate that greedy ensemble selection improves performance by an average of 8.8% on NDCG@5, 8.6% on NDCG@10, and 16.3% on NDCG@20 compared to the single best model on five datasets. Additionally, while no single model performs best across all datasets, greedy ensemble selection consistently performs best, representing the most robust recommender with regard to performance.

The implementation of greedy ensemble selection, along with the code and necessary documentation to reproduce all experiments, is publicly available in our GitHub repository[1].

2. Related Work

The use of ensembling techniques in recommender systems is not new and has been covered in the literature [4, 3, 5, 10].

Today, primarily hybrid recommender systems use ensembling to mitigate the weaknesses of individual algorithms [3, 5, 4]. Thereby, hybrid recommender systems require knowledge of the strengths and weaknesses of different algorithms to ensemble them effectively. In contrast to hybrid recommender systems, our work focuses on ensembling recommender system algorithms without manually selecting complementary algorithm combinations.

As in our work, researchers have applied ensembling techniques that do not require manual model selection for ensembling. For example, they have used standard machine learning ensemble techniques, such as bagging and boosting to recommender systems [11, 10], allowing the ensembling of a diverse set of models without manual model selection. However, their work mostly focuses on rating prediction tasks. Furthermore, bagging and boosting require the modification of training data. We focus on post-hoc ensembling, only taking model predictions into account.

[1]https://github.com/ISG-Siegen/greedy-ensemble-selection-for-top-n-recommendations

RobustRecSys: Design, Evaluation, and Deployment of Robust Recommender Systems Workshop @ RecSys 2024, 18 October, 2024, Bari, Italy.

tobias.vente@uni-siegen.de (T. Vente);
zainil.mehta@student.uni-siegen.de (Z. Mehta);
lukas.wegmeth@uni-siegen.de (L. Wegmeth);
joeran.beel@uni-siegen.de (J. Beel)
0009-0003-8881-2379 (T. Vente); 0009-0002-0556-9493 (Z. Mehta);
0000-0001-8848-9434 (L. Wegmeth); 0000-0002-4537-5573 (J. Beel)

Additionally, researchers have analyzed ensembling for various other aspects of recommender systems. Researchers ensemble predictions of models trained on different datasets containing different user feedback types [12, 13]. Researchers focus on ensembling techniques to ensemble models optimized for different objectives [14, 15]. Researchers implement ensembling techniques specifically designed and tested for certain domains, applications, or with a limited number of base models [16, 14, 17]. Or optimize ensembling techniques to specific datasets to showcase the capabilities of ensembling without focusing on the generalization [18]. However, these approaches have limitations: they often only work with models from the same algorithm, require multiple data inputs, necessitate optimization for multiple objectives, or focus ensembling for rating predictions.

Recent work has focused on greedy ensemble selection for recommender systems [19, 20]. In this work, the authors applied greedy ensemble selection to rating prediction tasks by treating them as regression problems, thus utilizing the standard greedy ensemble selection approach for regression. However, this method does not offer solutions for top-n ranking predictions, leaving a gap in the application of greedy ensemble selection in recommender systems.

3. Greedy Ensemble Selection

Greedy ensemble selection, as implemented in machine learning, cannot be directly applied to top-n recommendations in recommender systems. Greedy ensemble selection for classification applies majority voting on predictions. However, majority voting fails for top-n recommendations since the number of repeating item recommendations across users is often insufficient. Similarly, taking the mean for regression tasks is not applicable, as top-n recommendations deal with ranked lists instead of single numeric values returned by each model. Therefore, we focus on aggregating and re-ranking the prediction scores of multiple models to generate an ensemble of their outputs.

To apply greedy ensemble selection, we assume we have a set of trained and optimized models P, each predicting k' items and their validation performance. We aim to aggregate the k' predictions of every model $p_n \in P$ into one ranked list of length k. This requires the length k' of every predicted list p_n to be at least the same length as k.

In ensembling, we can utilize more predictions (k') than the desired output list length (k). There is a chance that a prediction ranked at position $k+1$ or beyond still holds relevance or contributes valuable information, even though it does not make it into the top k predictions of the model. Taking k' predictions into account enables the ensemble process to utilize a broader range of data, potentially improving the performance of the final ensembled recommendations. Furthermore, utilizing k' predictions does not increase the prediction cost of the base models since all models score all predictions anyway before selecting the top k.

We normalize all prediction scores of k' and multiply each prediction score by the validation performance of the respective model p_n (Algorithm 1). The normalization ensures that all models have an equal impact in the ensembling, while the validation performance multiplication weights the impact based on the models' performance. Consequently, the impact of models with good validation performance will be increased relative to those of poorly performing models.

Then, we initiate the greedy search by examining all subsets of $p_n \in P$ and aggregating the k' predictions by aggregating and re-ranking the prediction scores of all in the subset included models (Algorithm 1). If items appear in the k' lists of multiple models, the item scores are summed to reflect their collective relevance across models. We then select the top-k predictions of the ensembled list. The performance of every subset of all models $p_n \in P$ is evaluated on the validation set. The best-performing ensemble of models is then selected as the final result of the greedy ensemble selection, and their top-k predictions are returned.

4. Experiments

We conducted all of our experiments with ten different factorization- and neighborhood-based algorithms and greedy ensemble selection on five datasets. The hardware includes AMD EPYC 7452 CPU processors, each with 32 cores and a CPU frequency ranging from 2.35 to 3.35 GHz.

4.1. Experimental Pipeline

In our experimental pipeline, we apply five-fold cross-validation to all five datasets, randomly splitting each fold into three sets: 60% for training, 20% for validation, and 20% for testing. With the training and validation sets, we optimize all included algorithms with two hours of random search to select the best hyperparameter configuration. With the test set, we evaluate the final performance of the single models as well as the greedy ensemble selection. This combination of five-fold cross-validation and random search allows each algorithm to be finely tuned on every subset of the data while mitigating the effects of randomness in data splits and hyperparameter selection [21].

We measure performance using *NDCG@k* for $k = 5, 10, 20$ to evaluate top-n ranking predictions for different list lengths. The *NDCG@k* model performance on the validation set, obtained from the random search optimization process, is later used to weight the model predictions (Section 3).

4.1.1. Datasets

We include five distinct datasets of different sizes in our experiments and refer to Table 1 for a detailed overview. We transform convert datasets with user ratings, specifically Movielens-1M, Movielens-100k, and CiaoDVD, into binary user feedback datasets as it is done in related work [4, 3, 5]. Furthermore, we prune all datasets such that all included users and items have at least five interactions, commonly known as five-core pruning [22, 23, 24]. Table 1 shows all included datasets' statistics after preprocessing.

Table 1
Data set statistics after five-core pruning and user feedback transformation. Split between the implicit (first part) and explicit (second part) feedback data sets.

Name	Interactions	Users	Items	Sparsity
Citeulike-a[25]	200,180	5,536	15,429	99.77%
Hetrec-Lastfm[26]	71,355	1,859	2,823	98.64%
CiaoDVD[2]	23,467	1,582	1,788	99.17%
MovieLens-1M[27]	835,789	6,038	3,307	95.81%
MovieLens-100k[27]	81,697	943	1,203	92.8%

Table 2
NDCG@10 performance of ten factorization- and neighborhood-based models, along with greedy ensemble selection, across five datasets. The best results for individual models are indicated in bold, while the overall best performance is highlighted in bold and underlined. The relative performance increase is calculated based on the performance of Popularity.

Algorithms	CiaoDVD	CiteULike-A	Hetrec-LastFM	MovieLens-1M	MovieLens-100k	Rel. Performance Increase
ALS	0.02	0.066	0.147	0.234	**0.232**	84%
BPR	0.013	0.027	0.082	0.119	0.173	9%
ImplicitMF	0.022	0.109	0.159	0.189	0.184	75%
ItemItem-BM25	0.024	0.106	0.168	0.234	0.221	**99%**
ItemItem-Cosine	0.01	0.082	0.169	0.217	0.21	82%
ItemItem-TFIDF	0.016	0.094	0.168	0.225	0.218	90%
ItemKNN	0.013	0.096	**0.175**	0.216	0.212	88%
LogisticMF	0.018	0.058	0.135	0.161	0.191	49%
UserKNN	**0.026**	**0.112**	0.157	**0.235**	0.225	**99%**
Popularity	0.016	0.009	0.07	0.142	0.142	0%
Greedy Ensemble	**0.03**	**0.117**	**0.183**	**0.247**	**0.243**	**108%**

4.1.2. Algorithms

We include ten factorization- and neighborhood-based recommender systems algorithms in our experiments. The algorithm implementations are from the *Implicit* [28] and *LensKit* [29] recommender systems libraries. The algorithms from *Implicit* are *Alternating Least Squares (ALS)*, *Logistic Matrix Factorization (LogisticMF)*, *Bayesian Personalized Ranking (BPR)*, and *Item-Item Nearest Neighbors* with distance metrics *Cosine Similarity*, *TF-IDF*, and *BM25*. The algorithms from *LensKit* are *Implicit Matrix Factorization (ImplicitMF)*, *User-User Nearest Neighbors (UserKNN)*, *Item-Item Nearest Neighbors (ItemKNN)*, and most *Popular*.

4.2. Greedy Ensemble Selection

We run greedy ensemble selection using various prediction input list lengths (k') to examine the impact of predictions ranked higher than k on the ensembling process (Section 3). We set k' to 5, 10, 15, 25, 50, 75, 100, 125, and 150. This wide range of k' values helps us identify trends in the impact of longer input list lengths. All ensemble configurations are evaluated on the validation set. Ultimately, we select the ensemble configuration that performs best on average across all folds, with the optimal k' value.

5. Results

Our experiments reveal that greedy ensemble selection enhances performance by an average of 8.8% on *NDCG@5*, 8.6% on *NDCG@10* (Table 2), and 16.3% on *NDCG@20* compared to the single best model. Since *NDCG@10* is the most commonly used evaluation metric with a cutoff of $k = 10$ and the trends are consistent across all k values, our analysis will focus on the *NDCG@10* results.

In general, the algorithm performance ranking varies across datasets. While Popularity always yields the lowest *NDCG@10* score (Table 1), the best-performing algorithm changes. UserKNN performs best on CiaoDVD, CiteULike-A, and MovieLens-1M, while ItemKNN performs best on Hetrec-LastFM and ALS on MovieLens-100k.

In contrast to the single algorithm performances, greedy ensemble selection consistently outperforms all algorithms across all datasets for all *NDCG@k values* and adverts the algorithm selection problem. Greedy ensemble selection effectively identifies and aggregates a subset of models that outperforms the single best model, resulting in the highest

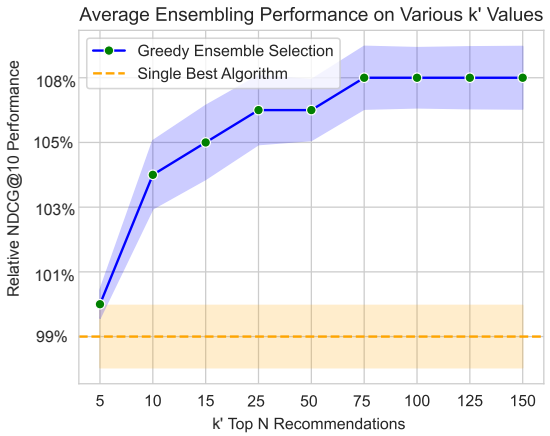

Figure 1: Performance differences of greedy ensemble selection with varying input prediction list lengths (k') compared to the virtual single best algorithm, averaged over five datasets. The x-axis represents the input prediction list lengths (k'), and the y-axis shows the *NDCG@10* performance. The shaded band represents the confidence interval of 95% for NDCG@10.

NDCG@k scores on all five datasets. This approach reliably enhances performance and presents robust results compared to the single best model respectively.

On average, the overall performance advantage of greedy ensemble selection compared to the single best algorithm is 8.6% on *NDCG@10* (108% vs. 99% for UserKNN, Table 2), but varies across datasets. Greedy ensemble selection shows a performance increase as high as 15.4% on CiaoDVD (0.03 vs. 0.026 for UserKNN, Section 5) and as low as 0.7% on Hetrec-LastFM (0.175 vs. 0.174 for ItemKNN, Section 5). On datasets like CiteULike-A, Movielens-1M, and MovieLens-100k, the performance increase is approximately 5%.

Longer prediction input lists of length k' (Section 3) improve the overall model performance (Fig. 1). Predictions that do not make it into the final k predictions of the single models still contribute valuable information to the ensemble process. While ensembling $k' = k$ predictions already enhance performance, increasing k' can further improve results. We tested using up to 150 predictions per user from each model and observed that the ensemble's performance plateaued beyond $k' = 100$ predictions (Fig. 1). Additionally, increasing k' beyond this point incurs higher computational costs during the ensembling process without yielding significant performance gains.

6. Discussion

To comprehensively answer our research question: How does the ensembling of factorization- and neighborhood-based models impact performance compared to a single optimized model? We conclude that greedy ensemble selection of factorization- and neighborhood-based models enhances the performance, on average, up to 16.3% compared to the single best model averaged over all datasets.

Our experiments show that greedy ensemble selection enhances performance and avoids the need for manual algorithm selection. By ensembling a subset of all available algorithms, greedy ensemble selection consistently achieves better results than any single algorithm across all included datasets. However, ensembling introduces an additional step in the recommender systems pipeline.

Despite its performance boost, greedy ensemble selection for top-n recommendations is expensive compared to single factorization- and neighborhood-based models. In addition to adding complexity to the pipeline, ensembling requires the training and optimization of multiple models to utilize their predictions for the top-n recommendations. Increasing the overall complexity and computational cost.

Nevertheless, the research community appears willing to accept higher computational costs for better performance, as evidenced by more sophisticated algorithms and the growing use of deep-learning approaches. While greedy ensemble selection involves an exhaustive search, further research could optimize the ensembling process for greater efficiency.

Currently, there is an ongoing debate in the field about whether recommender systems are truly making progress [9]. Much of the current research focuses on developing new (deep-learning) approaches, which do not necessarily outperform well-optimized traditional models. Revisiting and adapting ensembling for top-n recommendations, particularly with easy-to-train traditional recommender system algorithms, could open a new research direction. By adapting and improving advanced ensembling methods, recommender systems could significantly enhance their performance, especially for top-n predictions.

6.1. Future Work

Future work can investigate the contribution of more advanced deep-learning algorithms to the ensembling process. This includes assessing the models' potential performance enhancements and comparing the overall performance to state-of-the-art deep-learning methods. Furthermore, analyzing more efficient strategies to build an effective ensemble is valuable. Finally, examining the impact of ensembling across different domains within recommender systems helps better understand domain-specific trends.

References

[1] A. Toscher, M. Jahrer, R. M. Bell, The bigchaos solution to the net□ix grand prize (????).

[2] D. H. Wolpert, Stacked generalization, Neural Networks 5 (1992) 241–259. doi:https://doi.org/10.1016/S0893-6080(05)80023-1.

[3] E. Çano, M. Morisio, Hybrid recommender systems: A systematic literature review, Intell. Data Anal. 21 (2017) 1487–1524. URL: https://doi.org/10.3233/IDA-163209. doi:10.3233/IDA-163209.

[4] R. Burke, Hybrid Systems for Personalized Recommendations, volume 3169 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, p. 133–152. URL: http://link.springer.com/10.1007/11577935_7. doi:10.1007/11577935_7.

[5] R. Burke, Hybrid recommender systems: Survey and experiments, User Modeling and User-Adapted Interaction 12 (2002) 331–370. doi:10.1023/A:1021240730564.

[6] P. Gijsbers, M. L. Bueno, S. Coors, E. LeDell, S. Poirier, J. Thomas, B. Bischl, J. Vanschoren, Amlb: an automl benchmark, Journal of Machine Learning Research 25 (2024) 1–65.

[7] J. Heinermann, O. Kramer, Machine learning ensembles for wind power prediction, Renewable Energy 89 (2016) 671–679.

[8] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, A. Smola, Autogluon-tabular: Robust and accurate automl for structured data, 2020. URL: https://arxiv.org/abs/2003.06505. arXiv:2003.06505.

[9] M. Ferrari Dacrema, P. Cremonesi, D. Jannach, Are we really making much progress? a worrying analysis of recent neural recommendation approaches, in: Proceedings of the 13th ACM Conference on Recommender Systems, RecSys '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 101–109. URL: https://doi.org/10.1145/3298689.3347058. doi:10.1145/3298689.3347058.

[10] A. Bar, L. Rokach, G. Shani, B. Shapira, A. Schclar, Improving Simple Collaborative Filtering Models Using Ensemble Methods, in: D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, Z.-H. Zhou, F. Roli, J. Kittler (Eds.), Multiple Classifier Systems, volume 7872, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 1–12. URL: http://link.springer.com/10.1007/978-3-642-38067-9_1. doi:10.1007/978-3-642-38067-9_1, series Title: Lecture Notes in Computer Science.

[11] R. Boim, T. Milo, Methods for boosting recommender systems, in: 2011 IEEE 27th International Conference on Data Engineering Workshops, 2011, pp. 288–291. doi:10.1109/ICDEW.2011.5767667.

[12] A. da Costa Fortes, M. G. Manzato, Ensemble Learning in Recommender Systems: Combining Multiple User Interactions for Ranking Personalization, in: Proceedings of the 20th Brazilian Symposium on Multimedia and the Web, WebMedia '14, Association for Computing Machinery, New York, NY, USA, 2014, pp. 47–54. URL: https://doi.org/10.1145/2664551.2664556. doi:10.1145/2664551.2664556.

[13] A. F. da Costa, M. G. Manzato, Exploiting multimodal interactions in recommender systems with ensemble algorithms, Information Systems 56 (2016) 120–132. URL: https://www.sciencedirect.com/science/article/pii/S0306437915300818. doi:10.1016/j.is.2015.09.007.

[14] D. Carmel, E. Haramaty, A. Lazerson, L. Lewin-Eytan, Multi-Objective Ranking Optimization for Product Search Using Stochastic Label Aggregation, in: Proceedings of The Web Conference 2020, WWW '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 373–383. URL: https://doi.org/10.1145/3366423.3380122. doi:10.1145/3366423.3380122.

[15] P. Nguyen, J. Dines, J. Krasnodebski, A Multi-Objective Learning to re-Rank Approach to Optimize Online Marketplaces for Multiple Stakeholders, 2017. URL: http://arxiv.org/abs/1708.00651. doi:10.48550/arXiv.1708.00651, arXiv:1708.00651 [cs].

[16] N. H. Kulkarni, G. N. Srinivasan, B. M. Sagar, N. K. Cauvery, Improving Crop Productivity Through A Crop Recommendation System Using Ensembling Technique, in: 2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS), 2018, pp. 114–119. doi:10.1109/CSITSS.2018.8768790.

[17] H. Wu, K. Yue, Y. Pei, B. Li, Y. Zhao, F. Dong, Collaborative topic regression with social trust ensemble for recommendation in social media systems, Knowledge-Based Systems 97 (2016). doi:10.1016/j.knosys.2016.01.011.

[18] S. Forouzandeh, K. Berahmand, M. Rostami, Presentation of a recommender system with ensemble learning and graph embedding: a case on MovieLens, Multimedia Tools and Applications 80 (2021) 7805–7832. URL: https://doi.org/10.1007/s11042-020-09949-5. doi:10.1007/s11042-020-09949-5.

[19] T. Vente, L. Purucker, J. Beel, The feasibility of greedy ensemble selection for automated recommender systems, in: COSEAL Workshop 2022, 2022. URL: https://www.researchgate.net/publication/373841225_The_Feasibility_of_Greedy_Ensemble_Selection_for_Automated_Recommender_Systems.

[20] T. Vente, M. Ekstrand, J. Beel, Introducing lenskit-auto, an experimental automated recommender system (autorecsys) toolkit, in: Proceedings of the 17th ACM Conference on Recommender Systems, 2023, pp. 1212–1216.

[21] T.-T. Wong, P.-Y. Yeh, Reliable accuracy estimates from k-fold cross validation, IEEE Transactions on Knowledge and Data Engineering 32 (2020) 1586–1594. doi:10.1109/TKDE.2019.2912815.

[22] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, P. Jiang, Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 1441–1450. URL: https://doi.org/10.1145/3357384.3357895. doi:10.1145/3357384.3357895.

[23] Z. Yue, Z. He, H. Zeng, J. McAuley, Black-box attacks on sequential recommenders via data-free model extraction, in: Proceedings of the 15th ACM Conference on Recommender Systems, RecSys '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 44–54. URL: https://doi.org/10.1145/3460231.3474275. doi:10.1145/3460231.3474275.

[24] Z. Yue, H. Zeng, Z. Kou, L. Shang, D. Wang, Defending substitution-based profile pollution attacks on sequential recommenders, in: Proceedings of the 16th ACM Conference on Recommender Systems, RecSys '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 59–70. URL: https://doi.org/10.1145/3523227.3546770. doi:10.1145/3523227.3546770.

[25] H. Wang, B. Chen, W.-J. Li, Collaborative topic regression with social regularization for tag recommendation, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI '13, AAAI Press, 2013, p. 2719–2725.

[26] I. Cantador, P. Brusilovsky, T. Kuflik, Second workshop on information heterogeneity and fusion in recommender systems (hetrec2011), in: Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys '11, Association for Computing Machinery, New York, NY, USA, 2011, p. 387–388. URL: https://doi.org/10.1145/2043932.2044016. doi:10.1145/2043932.2044016.

[27] F. M. Harper, J. A. Konstan, The movielens datasets: History and context, ACM Trans. Interact. Intell. Syst. 5 (2015). URL: https://doi.org/10.1145/2827872. doi:10.1145/2827872.

[28] B. Frederickson, Fast python collaborative filtering for implicit datasets, URL https://github. com/benfred/implicit (2018).

[29] M. D. Ekstrand, Lenskit for python: Next-generation software for recommender systems experiments, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 2999–3006. URL: https://doi.org/10.1145/3340531.3412778. doi:10.1145/3340531.3412778.