Co-rating Attacks on Recommendation Algorithms

Manfred Moosleitner
Department of Computer Science
Universität Innsbruck, Austria

Günther Specht
Department of Computer Science
Universität Innsbruck, Austria

Eva Zangerle
Department of Computer Science
Universität Innsbruck, Austria

firstname.lastname@uibk.ac.at

ABSTRACT

Online shops, streaming services, and booking systems use algorithms to recommend items from their stock to users. These recommendations are often calculated based on the interactions of other users with the items, e.g., buying a product or watching a movie. This creates an attack point where the outcome of recommendation algorithms can be purposefully manipulated by manually or automatically created user interactions, aimed to raise or lower the relevance of specific items. We study the attackability of recommender algorithms by simulating a series of attacks on six recommendation algorithms, using three attack strategies, and two opposing attack objectives. We run these experiments with varying numbers of co-ratings per attack and evaluate the overall item ranking and an average change in average rank. Our results show that the effort required of and the efficiency reached by the attacks greatly depends on the strategy, objective, and recommendation algorithm. Additionally, the calculated average change in average rank provides an indicator about the attackability of recommendation algorithms. We find that neighborhood- and cluster-based algorithms show a higher vulnerability against attacks compared to algorithms based on matrix factorization.

Keywords

Recommender Systems, Co-Rating Attacks, Attack Measurements, Manipulation, Bias

1. INTRODUCTION

Recommender systems are ubiquitous in today's online world as they provide users of, for instance, online shops, video, and music streaming services with recommendations of items that might be interesting to them [21, 23]. Such recommendations are computed based on interactions of users with items. An interaction can be, e.g., a user who rates an item with five stars, often called *collaborative filtering* [22]. Based on interactions between users and items, in collaborative filtering recommender algorithms, rating predictions are frequently computed based on user similarity, which is captured by co-rated items of a pair of users. This opens a possibility for manipulations of the computed recommendations, as the ratings stem from the user base and their interactions with the system. One type of possible manipulations is *shilling attacks* [11], where additional interactions are injected into the system to alter the recommendations computed by the system. One famous example of such a shilling attack at Amazon was reported on December 7th in 2002 by the British online news service "The Register"[1]. On Amazon, users are provided with recommendations in the form of "Customers who viewed this article, also viewed ...". To manipulate these recommendations, a group of attackers interacted with two different books multiple times, aiming to make the second book appear in the recommendation section of the first book, although it diverged in content and genre. It is unknown how many people were involved exactly and how often the two books were co-viewed, but this example shows vulnerabilities that can be abused to manipulate the outcome of recommender systems.

For this paper, we focus on *co-rating attacks*, a specific form of shilling attacks. A co-rating attack means that a single attack in an attack series consists of two ratings, e.g., one rating of user B for item X and one rating of user B for item Y, to influence the rating behavior of the targeted system, possibly for one specific user A. We call user A the *target user* and user B the *auxiliary user*. Similarly, we call item X the *target item* and item Y the *auxiliary item*, and the attack aims to raise the predicted ratings for the target item X. The contribution of this work is two-fold: (i) we provide a systematic evaluation of the vulnerability of recommendation algorithms against co-rating attacks and the effort required for such attacks to be successful, and (ii) we propose a new metric to measure the attackability and hence, the vulnerability (respective resilience) of recommendation algorithms against different co-rating attack strategies.

2. RELATED WORK

In the following, we discuss work related to attacks on recommender systems and measures that aim to quantify these attacks.

Lam et al. [12] classified three different areas where attacks on recommender systems can happen. Firstly *exposure*, where systems get breached, and private data from

32nd GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), September 01-03, 2021, Munich, Germany.

[1]https://www.theregister.co.uk/2002/12/07/sodomites_overrun_amazon_com/

users is leaked. Secondly, the authors describe *sabotage* as an attack to hinder the service of the targeted system at all and give denial of service as an example. Thirdly, they mention the area of *bias*, where the attacks focus on purposefully changing the ratings of the recommender system. The former two types, exposure, and sabotage are out of the scope of this work, as they are more in the area of classic security topics. We focus on the manipulation of bias instead. Bias in our context can be, e.g., that a popular item is given a higher relevance when calculating the recommendations, purely due to its popularity. This is in contrast with the main purpose of recommender systems, namely, to suggest items to users that fit their personal taste.

Jannach et al. [9] categorize attacks into three *attack dimensions*. They termed the first category *push attacks*; their purpose is to increase the predicted ratings for a specific item. The attacks in the second category are *nuke attacks*, which aim to lower the predicted ratings for a specific item. Please note that we refer to these attacks as *pull attacks*. The last category described serves the purpose of rendering the recommender system ineffective and unpredictable. One additional dimension for attacks lies in the strategy when selecting the auxiliary user and item. Mobasher et al. [15] give a detailed overview of different strategies, which focuses on the selection of the auxiliary items by using statistics about the data. Some of the strategies described choose auxiliary items from the same category and genre as the target item, other strategies use the popularity of the item or choose them randomly. In contrast, we focus on the selection of the auxiliary user.

Early works [11, 18, 16, 19] and more recent publications [4, 1] feature attacks only on variations of the K-Nearest-Neighborhood (KNN) algorithms. In this work, we additionally consider algorithms based on matrix factorization (MF) [14], singular value decomposition (SVD) [10], co-clustering [6], and "popularity differential" [13].

Burke et al. [3] state that a popular way to measure the robustness of a recommendation algorithm against attacks is the average prediction shift, which measures the average shift of the ratings predicted by the algorithm across all users. The prediction shift reflects whether an attack has the intended effect, but not how large the impact is. As a solution, they propose using the average hit ratio, which counts the number of target items that appear in the list of recommendations of the target user.

O'Mahony et al. [20] analyze the performance of attacks using HitRatio and prediction shift, but only on a KNN-based recommender. Lam et al. [12] provide an overview of different possibilities to attack a recommender system, but they do not present a quantitative analysis. We use a similar approach as O'Mahony et al. [19], but evaluate the attacks for multiple recommendation algorithms and multiple attack strategies, and measure the effect of the intensity of an attack.

3. METHODOLOGY

The general idea of our work is to capture the *effort* required for an attack to be successful and the *impact* of co-rating attacks on the predicted ratings. The required effort can be measured by the number of co-ratings used in an attack, i.e., the number of interactions of the auxiliary user with the target item and the auxiliary item in the form of two rating entries in the dataset. We consider a push attack to be successful if the targeted item reaches the top-10 recommendations, as this increases the visibility of the item and previous research has shown that a list of ten recommendations can be a sensible choice in regards to set attractiveness and choice difficulty [2]. For pull attacks, we consider an attack to be successful if the rank of the target item, after the attack, is at least ten ranks lower than before the attack. This would pull the targeted item out of the top-10 and decreases its visibility. We define the *impact* of an attack as the change in rank of the target item in the average item ranking.

For our experiments, we rely on the MovieLens 100K dataset [8][2] as the basis, which is an established dataset widely used in the recommender systems research community [18, 6, 13, 19, 16, 20, 14, 1].

Our experiments are organized in individual *steps*. A step can be viewed as the current state of the experiment, including the dataset, the training of the algorithm, and the predicted ratings. For each recommender algorithm, we start with zero co-ratings, train the algorithm on the unmodified dataset, and calculate the rankings per item for each user and the average rank per item over all users. We refer to this as *step 0*, and subsequently, add a single co-rating to the dataset (and hence, run an attack); we refer to this as *step 1* and repeat this procedure. In each step, we compute the change in the rank and the average rank for the target item. We hypothesize that a noticeable effect should be reached within 100 iterations, therefore using this number as the maximum number of steps in our experiments.

We analyze the experiments along the four dimensions *recommendation algorithm*, *attack strategy*, *attack purpose*, and *the number of steps*, where the *attack purpose* is defined as whether the goal of the attack is to lower or raise the rank of the target item, whereas the *attack strategy* describes how the auxiliary user and the auxiliary item are selected.

The configurations for single experiments have been composed such that they cover a broad spectrum of the full experiment space. We describe these dimensions in the following.

3.1 Recommendation Algorithms

As the main focus of this work lies in examining how different recommendation algorithms behave when exposed to co-rating attacks, we evaluate six recommendation algorithms[3]. We selected two k-Nearest Neighbor [17] variations, *KNNBasic* and *KNNWithMeans*[4], where the neighborhood is built using the k-most similar users with respect to the user ratings. In our case, *mean squared difference* [17] was used to calculate the similarity. The kNN-based algorithms were chosen because they are fairly simple, are commonly used as a baseline, and it should be easy to find a working attack strategy for them.

SlopeOne [13] is also a simple algorithm, where the ratings for a user A are computed using the ratings of other users who share rated items with A. E.g., user A rated item X and Y, and user B only rated item X. The rating difference of user B is added to the rating of user A's rating for item X to predict the rating of user A for item Y.

[2]`https://grouplens.org/datasets/movielens/100k/`
[3]For the implementation, we relied on the Python Surprise library `https://github.com/NicolasHug/Surprise`
[4]Here the predicted rating is added to the mean rating across all users.

CoClustering [6] was selected because it does not only use user- and item-clusters, but also the co-clusters which should make it harder to find a working attack strategy compared to KNN-based approaches.

Singular Value Decomposition (SVD) [10] aims to reconstruct the rating matrix ($user \times items$) from a matrix representing the user's latent factors and a second matrix, representing the latent factors of the items. *Non-Negative Matrix Factorization (NMF)* [14] also relies on matrix factorization, which is used by Netflix [7] and Youtube [5].

To assess the difference in the performance of the chosen algorithms, the algorithms were evaluated using five-fold cross-validation using the MovieLens dataset and the configuration parameters as they were used in our experiments. The resulting Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) of the predicted ratings in Table 1 show that the performances of the algorithms are similar.

	RMSE		MAE	
	mean	std	mean	std
KNNBasic	0.9785	0.0053	0.7105	0.0036
KNNWithMeans	0.9507	0.0032	0.7491	0.0023
SVD	0.9369	0.0044	0.7386	0.0040
NMF	0.9628	0.0038	0.7569	0.0030
SlopeOne	0.9447	0.0024	0.7423	0.0025
CoClustering	0.9655	0.0050	0.7561	0.0059

Table 1: Five-fold cross-validation of the evaluated algorithms.

3.2 Attack Strategies

In the following, we investigate three different co-rating attack strategies. Our first attack strategy is the *baseline approach (BASIC)*, where the auxiliary user is created as a fresh new user, i.e., a user that has not provided any ratings so far. The second attack strategy is the *user activity approach (ACT)*, where the auxiliary user is selected based on the rating activity of the user—the dataset is analyzed at runtime, and the most active user is determined and misused as the auxiliary user, based on the data in step 0. The third attack strategy is the *user similarity approach (SIM)*, where the user most similar to the target user is selected as the auxiliary user. The core idea here is that user similarity metrics are also used in collaborative filtering recommender algorithms [22] and hence, similarly choosing the auxiliary user seems promising. As for the user similarity computation, we rely on the cosine similarity of the user rating vectors. For all three attack strategies, the same auxiliary user is used in all steps. It is important to note that we examine the effect of co-rating attacks on recommendation algorithms alone and not whole recommender systems, thus using the full knowledge about the dataset in the attack strategies.

In all approaches, an arbitrary user was chosen as target user—i.e., we aim to measure the impact of the co-rating attacks on the movie rankings for this user. For our experiments, the user with user-id 1 was picked as the target user. The movie "The Truth About Cats & Dogs (1996)" (movie id 111) was randomly selected as the auxiliary item from the set of movies the target user rated with five stars. The movie "Scream of Stone (Schrei aus Stein) (1991)" (movie id 1682) was selected as the target item, because it was the newest movie in the dataset (highest movie id) and because the movie has not been rated by the target user yet. Table 2 shows some statistics about the data before co-ratings are added and the chosen user and movies. The data collected shows that the target-user is quite active, with more than double the number of ratings than the average user and more than four times the number of ratings than the median over all users. We can also see that the mean ratings and the standard deviation for the target user are higher than the mean rating for all users. The data for the auxiliary movie shows that the movie is rated more often, but only a little lower, than the average movie. The target movie has only a single rating in the original data, which should be beneficial for our attacks.

An attack is now realized by the auxiliary user awarding high ratings for both movies (and hence, co-rating these) until the target item appears in a prominent position in the target user's list of recommended movies.

For running the experiments, in the first step, the raw dataset is used to train the recommender algorithms to be evaluated. These trained models are then used to predict the ratings for all items for all users, from which we infer a ranking of all items for each user. These per-user rankings are then used to compute the average rank for each item over all users. In the next step, one co-rating is added to the dataset. Subsequently, the extended dataset is used for computing updated rankings for all items for all users and the average rank over all users. This procedure is repeated 100 times for each algorithm, providing the rank of the target item for the target user and the average rank over all users, and also the change in rank between the individual steps, to show the effect of the attack from one step to the next.

3.3 Attack Purposes

We evaluate the three attack strategies for both pull attacks (aiming to lower the predicted rating of the target item) and push attacks (aiming to raise the predicted rating of the target item). For push attacks, it is sufficient to pick an item that has not been rated by the target user, as the target item. For pull attacks, in contrast, we computed the recommendation lists, based on the data in step 0, and chose the top-ranked item as target item.

3.4 Quantifying Attackability

Here we introduce our novel attackability metric, which takes into account the effort and the effect of an attack. We apply this metric to the data collected during our experiments and present the results in Section 4.3.

Besides investigating the manipulation of the rank of an item for a single user, we are also interested in the impact of the co-rating attacks on the average ratings of each item across all users. Thus, we compute an average ranking for all items (cf. Section 3.2), which is used to determine the average rank of the target item over all users. Furthermore, we aim to quantify the attackability of recommendation algorithms. Hence, we propose to consider the absolute average change in the average rank of the target item. Consider the target item being ranked at position 1000 at step 0, and in step 1 ranked at 100, then the change in ranks is 900. We compute this change across all steps and compute the average change in rank for the target item. Formally, we define $\overline{R_i}$ as the average item rank for step i, i.e., R_0 for step 0, etc. We further define $rank_i$ as the rank of the chosen target item in the corresponding average item ranking R_i. The

	Ratings							
	Min	**Q1**	**Median**	**Q3**	**Max**	**Mean**	**Std**	**Count**
Full Dataset	1	3	4	4	5	3.53	1.13	100,000
Target User	1	3	4	5	5	3.61	1.26	272
Aux. Movie	1	3	4	4	5	3.49	0.96	272
Target Movie	3	3	3	3	3	3.00	0.00	1

(a) Statistics about the ratings in the data.

	Number of Ratings							
	Min	**Q1**	**Median**	**Q3**	**Max**	**Mean**	**Std**	**Count**
Per User	20	33	65	148	737	106.05	100.93	943
Per Movie	1	6	27	80	583	59.45	80.38	1,682

(b) Statistics about the number of ratings in the full dataset.

Table 2: Statistics about the data and the chosen user and movies for different aggregations. Shown are the five-number summary, mean, and standard deviation of the rating values in Table 2a and of the number of ratings in Table 2b. The column **Count** shows the number of rows for each aggregation, and the ratings for the auxiliary movie are from 272 individual users.

change in rank is then defined as the difference between two steps: $\Delta rank_{i,i+1} = rank_{i+1} - rank_i$. The average change in rank is computed based on the sum of all $\Delta rank_{i,i+1}$ of the available steps, as shown in Equation 1, where n is the number of steps to be used.

$$\overline{\Delta} rank = \sum_{i=0}^{n-1} \frac{\Delta rank_{i,i+1}}{n} \tag{1}$$

4. RESULTS AND DISCUSSION

In our experiments, we run multiple attacks, using multiple dimensions, adding up to a total of 1,800 individual trained prediction models, which were used in 3,600 experiments. In the following, we present the results of these evaluations.

4.1 Push Attack Evaluation

For this evaluation, we analyze the rank of items in the recommendation list for each step. Here, the best achievable rank in the ranking is zero. As stated in Section 3, we consider a push attack successful if the attack changes the rank prediction of the target item to be in the top-10 of the average item ranking.

The first set of experiments investigate the BASIC attack strategy, where a new user is created as the auxiliary user and the auxiliary item was randomly selected from the set of movies that have not been rated by the target user.

Figure 1a shows that the average rank for CoClustering is influenced most as the curve races directly to the top ranks within the first few steps. The results for NMF and SVD show capricious behavior within a confined area, but do not reach a high rank. The ranks for KNNBasic and KNNWithMeans show that they are only influenced at the start but converge quickly. SlopeOne is unaffected by the attack with no changes in the rank. For NMF and SVD, the attacks can be regarded as failed, but the ranking values have a high variance, which goes in the direction of sabotage, as introduced in Section 2. Only the attack on CoClustering was successful, showing a high attackability of the basic attack strategy.

The second set of experiments aims at investigating the user activity attack strategy; Figure 1b shows the obtained results. We can see that the values for all algorithms rush towards the top at the start. CoClustering, KNNBasic, and KNNWithMeans quickly arrive at the top ranks. NMF and SVD show some erratic behavior and SlopeOne converges shortly after reaching rank 200. CoClustering was attacked successfully again since a rank in the top ten was clearly reached. The attacks on KNNBasic and KNNWithMeans also reach a high rank but did not reach the top-10. NMF, SVD, and SlopeOne could not be attacked successfully. Generally, we observe that the activity-based attack strategy reaches a higher attack effect than the basic attack approach.

The third set of experiments used the similarity between the users to select the target user; the results are shown in Figure 1c. We observe a tendency towards higher ranks for NMF and SVD, but the erratic behavior starts to dominate early on. In addition to CoClustering, KNNBasic, and KNNWithMeans, also SlopeOne changes to a high rank quickly. Using the similarity to select the target user leads to the highest ranks for the majority of algorithms. All algorithms except for NMF and SVD were attacked successfully.

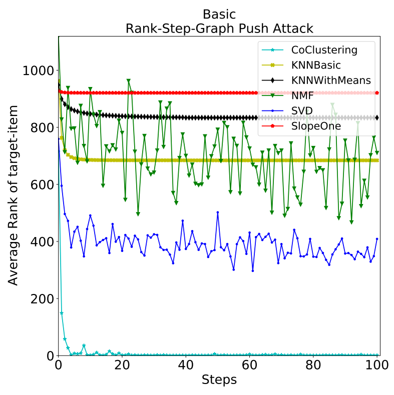
4.2 Pull Attack Evaluation

In the following, we evaluate pull attacks, where an attack is considered successful if the rank of the target item is consistently lowered by 10 or more ranks after an attack.

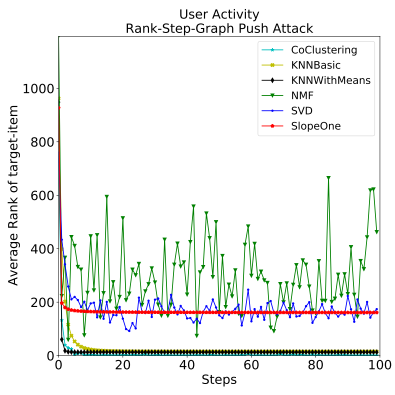
For the BASIC attack approach, a fresh user is used as the auxiliary user. As the target item, the movie with the highest id from the set of movies the target user has not rated was selected. Figure 2a shows the obtained results. SlopeOne is mostly unaffected by the pull attack. The effect on SVD was rather small and shows minor fluctuations. The average rank for KNNWithMeans moves slowly and converges early around 110. CoClustering, KNNBasic, and NMF show high vulnerability against the attacks. Most interesting is the behavior of NMF showing a clear tendency towards the lower ranks when compared to SVD, the other MF-based algorithm, whose behavior showed to be stable and unaffected by the attacks.

The next set of experiments analyzes the activity-based approach with pull attacks. Figure 2b shows that NMF and SVD are only marginally affected as both algorithms show only minor fluctuations. CoClustering, KNNBasic, KNNWithMeans, and SlopeOne show a clear tendency towards the lower ranks.

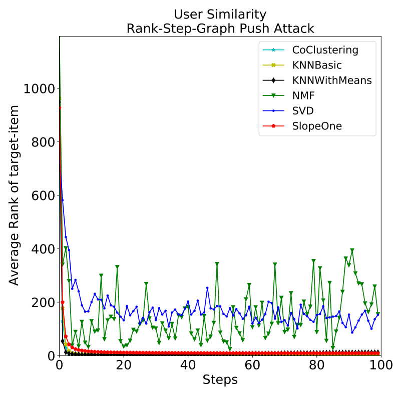
The last set of experiments used the similarity-based approach to launch pull attacks on the algorithms. In Figure 2c, we observe that the results are quite similar to the



(a) Basic attack approach.

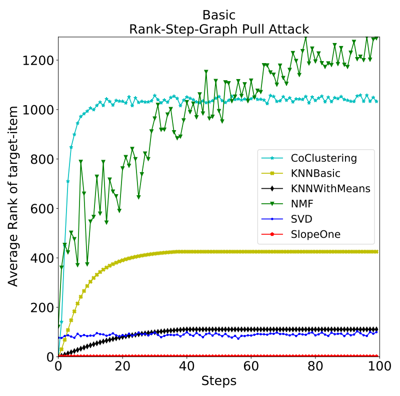


(b) User activity approach.

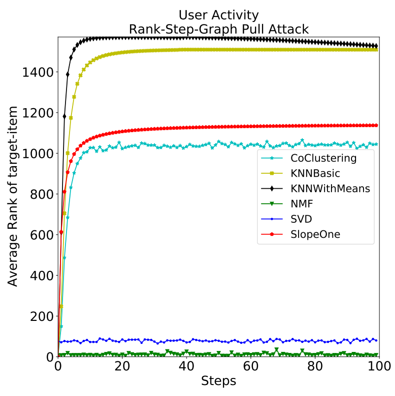


(c) User similarity approach.

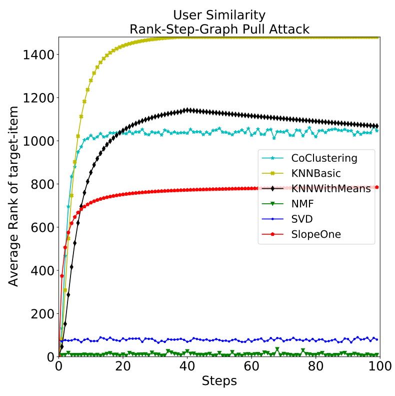
Figure 1: Push attacks: average rank for the different co-rating attack strategies.



(a) Basic attack strategy.



(b) User activity strategy.



(c) User similarity approach.

Figure 2: Pull attacks: ranks in the average ranking for the different co-rating attack strategies.

outcome of the experiment with the activity-based approach. SVD and NMF only show single spikes in the change of the average rank. For the remaining algorithms, the average rank of the target item changes rapidly to a lower rank.

4.3 Attackability Evaluation

Table 3 shows the attackability evaluation measured by the $\overline{\Delta} rank$ measure (cf. Section 3.4) for the data from our experiments for push and pull attacks, and each of the three attack strategies used.

For push attacks, we observe a relation between the attackability, i.e., if an algorithm could be attacked successfully, and the average change in average rank for the corresponding algorithms. The relation also holds for a lower observed attackability, i.e., whether an algorithm could not be attacked successfully, and lower values for the average change in average rank. For pull attacks, we also see the same relation between our observed attackability and the calculated average change in average rank. Even though the push attacks on SVD and NMF were mostly considered unsuccessful, the average change in average rank is still higher due to the erratic behavior the algorithms showed.

From this, we can state that our newly proposed metric, the average change in average rank, is an intuitive and feasible indicator for how vulnerable an algorithm is to a specific attack strategy and attack purpose, hence describing the attackability of the algorithm against co-rating attacks.

	Push attacks			Pull attacks		
Algorithm	BASIC	ACT	SIM	BASIC	ACT	SIM
CoClustering	7.14	7.14	7.14	10.43	10.51	10.52
KNNBasic	2.78	9.57	9.69	4.29	15.24	14.95
KNNWithMeans	1.15	9.45	9.45	1.11	15.42	10.78
NMF	4.09	7.40	10.50	11.79	0.01	0.01
SVD	3.50	5.91	6.13	0.23	0.03	0.02
SlopeOne	0.06	7.74	9.27	0.00	11.48	7.92

Table 3: Attackability for push and pull attacks, where brighter colors represent a lower change in average Δ rank and darker colors signal a higher change in average Δ rank.

5. CONCLUSION

We investigated the attackability of recommendation algorithms against co-rating attacks. We ran experiments with a varying number of co-ratings, approaches, and types of attack for the chosen algorithms. This added up to a total of 1,800 prediction models, which were used in 3,600 experiments. In these experiments, we investigated the effort, the number of co-ratings, and the effect of the attacks on the results of the algorithm, where we also introduce a new metric to express the attackability from the collected data, by calculating the average change in average rank over all steps. We find that the nearest neighbor and co-clustering algorithms were the least resilient algorithms. Furthermore, we observe that the matrix factorization-based algorithms show erratic behavior when attacked with push attacks, but generally showed a lower attackability when assaulted with pull attacks. Our results show that a handful of coordinated users are enough to manipulate the outcome of recommendation algorithms, thus having an impact on which products we may buy, movies we watch, or locations we visit for our next holiday. For future work, we aim to investigate the influence of the number of ratings when choosing target-user and -movie, and auxiliary user. Furthermore, we aim to test the generalizability of our approach and metric by running the experiments using different target-users and -items, and on different and larger datasets.

6. REFERENCES

[1] F. Aiolli, M. Conti, S. Picek, and M. Polato. Big Enough to Care Not Enough to Scare! Crawling to Attack Recommender Systems. In *European Symposium on Research in Computer Security*, pages 165–184. Springer, 2020.

[2] D. Bollen, B. P. Knijnenburg, M. C. Willemsen, and M. Graus. Understanding Choice Overload in Recommender Systems. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, page 63–70, New York, NY, USA, 2010. Association for Computing Machinery.

[3] R. Burke, M. P. O'Mahony, and N. J. Hurley. Robust Collaborative Recommendation. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 961–995. Springer, 2015.

[4] K. Chen, P. P. Chan, F. Zhang, and Q. Li. Shilling Attack based on Item Popularity and Rated Item Correlation against Collaborative Filtering. *International Journal of Machine Learning and Cybernetics*, 10(7):1833–1845, 2019.

[5] P. Covington, J. Adams, and E. Sargin. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, page 191–198, New York, NY, USA, 2016. Association for Computing Machinery.

[6] T. George and S. Merugu. A scalable Collaborative Filtering Framework based on Co-Clustering. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4–pp. IEEE, 2005.

[7] C. A. Gomez-Uribe and N. Hunt. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Transactions on Management Information Systems*, 6(4), Dec. 2016.

[8] F. M. Harper and J. A. Konstan. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst. (TiiS)*, 5(4):19:1–19:19, Dec. 2015.

[9] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender Systems - An Introduction.* Cambridge University Press, 2010.

[10] Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37, 2009.

[11] S. K. Lam and J. Riedl. Shilling Recommender Systems for Fun and Profit. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 393–402, New York, NY, USA, 2004. ACM.

[12] S. K. T. Lam, D. Frankowski, and J. Riedl. Do You Trust Your Recommendations? An Exploration of Security and Privacy Issues in Recommender Systems. In G. Müller, editor, *Emerging Trends in Information and Comm. Secur.*, pages 14–29. Springer, 2006.

[13] D. Lemire and A. Maclachlan. Slope One Predictors for Online Rating-Based Collaborative Filtering. In *Proceedings of the 2005 SIAM Int. Conference on Data Mining*, pages 471–475. SIAM, 2005.

[14] X. Luo, M. Zhou, Y. Xia, and Q. Zhu. An Efficient Non-Negative Matrix-Factorization-Based Approach to Collaborative Filtering for Recommender Systems. *IEEE Transactions on Industrial Informatics*, 10(2):1273–1284, 2014.

[15] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams. Toward Trustworthy Recommender Systems: An Analysis of Attack Models and Algorithm Robustness. *ACM Trans. Internet Technol.*, 7(4):23–es, Oct. 2007.

[16] B. Mobasher, R. Burke, C. Williams, and R. Bhaumik. Analysis and Detection of Segment-Focused Attacks Against Collaborative Recommendation. In O. Nasraoui, O. Zaïane, M. Spiliopoulou, B. Mobasher, B. Masand, and P. S. Yu, editors, *Advances in Web Mining and Web Usage Analysis*, pages 96–118. Springer, 2006.

[17] X. Ning, C. Desrosiers, and G. Karypis. A Comprehensive Survey of Neighborhood-Based Recommendation Methods. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 37–76. Springer, 2015.

[18] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre. Collaborative Recommendation: A Robustness Analysis. *ACM Trans. Internet Technol.*, 4(4):344–377, Nov. 2004.

[19] M. P. O'Mahony, N. J. Hurley, and G. C. Silvestre. Recommender systems: Attack Types and Strategies. In *AAAI*, pages 334–339, 2005.

[20] M. P. O'Mahony, N. J. Hurley, and G. C. Silvestre. Attacking Recommender Systems: The Cost of Promotion. In *Proc. of the Workshop on Recommender Systems, in Conjunction with the 17th Eur. Conf. on Artif. Intell., Riva del Garda, Italy*, pages 24–28, 2006.

[21] M. Pichl, E. Zangerle, and G. Specht. Improving Context-Aware Music Recommender Systems: Beyond the Pre-Filtering Approach. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, ICMR '17, page 201–208, New York, NY, USA, 2017. Association for Computing Machinery.

[22] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative Filtering Recommender Systems. In *The Adaptive Web*, pages 291–324. Springer, 2007.

[23] M. Schedl, P. Knees, B. McFee, D. Bogdanov, and M. Kaminskas. Music Recommender Systems. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 453–492. Springer, 2015.