Evaluating the Robustness of Collaborative Filtering Recommender Systems against Attacks

Andrew Zhou
Department of Data Science
University of Washington
Seattle, WA 98195
ajz55@uw.edu

Kenny Zhang
Department of Statistics
University of Washington
Seattle, WA 98195
zhehaoz@uw.edu

Shruthi Kundapura
Department of Computer Science
University of Washington
Seattle, WA 98195
skunda@uw.edu

1 Introduction

Recommender Systems (RS) which have become an essential part of many online applications are known to be vulnerable to attacks. Among the different attacks studied, we are focusing on understanding the impact of shilling attacks on different recommender systems. Shilling attacks are initiated by injecting carefully crafted fake user profiles with chosen items and ratings into the system. There are two main incentives to these attacks. The first incentive is to prevent users from using a specific recommender system or E-commerce system, by making the system generate "fake" or "bad" recommendations to its users using these attacks. This attack format is called the data pollution attack. The second incentive is to greatly increase the chances that an attacker-chosen target item is recommended to many users of the system, which would increase publicity and sales of the chosen target item. Shilling attacks follow this format to promote target items, which is what we will evaluate in this work.

Our goal is to analyse the five shilling attack methods((Li et al. 2017)[8]): Random Attack, Average Attack, Bandwagon Attack, Segmented Attack, Sampling Attack on different recommender systems. Recommender models chosen for evaluation are: User-based KNN model, Item-based KNN model, Latent Factor model, Neural Matrix Factorization model (NeuMF)(Dziugaite et al. 2015)[3]). The intent of these evaluations is to discover which of these attack types are more effective on what kind of recommender system, and also to identify properties of the target items which may influence the impact of these attacks.

Lastly, we will evaluate the attacks on most recently proposed causal based recommender system model (Wang et al)[11] with an exposure model. This new model is different from typical collaborative filtering recommender system, by assuming that all users will not be equally likely to be exposed to all movies. To analyse the impact of attacks on this model we will build an exposure model with Poisson factorization to predict which movies the user will watch. The main motivation behind this evaluation is our intuition that the layer of movie exposures information added by Poisson factorization, will make this system more robust to attacks against other recommenders.

To our knowledge, this is the most comprehensive analysis on shilling attacks for different recommender models. We provide multiple metrics for evaluations and a summary conclusion on all type of attacks. We also implemented our own optimization for the causal recommender systems and provide the attack data generation procedure.

2 Related Works

Deng et al. 2016[2] provides a description of data poisoning or shilling attacks which uses fake user profiles with carefully crafted ratings to promote target item and (2013)[12] describes profile pollution attack where the intent of injecting fake information is to perturb results obtained from services using personalization algorithms. We have used [8] as a reference to implement 5 kinds of traditional

shilling attacks: Random, Average, Bandwagon, Sampling, Segment attack. This categorization of attack models is done on the basis of way in which selected filler items are chosen and also on its ratings in fake profile. Lam et al. (2004)[7] presents with an idea to measure the effectiveness of these traditional attacks on kNN user-based and item-based collaborative filtering RS models, using metrics like Prediction shift, Expected TopN Occupancy and Mean Absolute Error(MAE). Similar approach is described Sandvig et al.(2007)[10] with conclusion that user-based kNN is more susceptible to attacks than item-based kNN. We have used these metrics in our evaluation approach.

Huang et al. 2021[5] gives an account of custom attack model built relying on the complete knowledge of RS algorithm used. This paper provides a systematic study of data poisoning attacks on deep learning based recommender systems where the attack is formulated as a non-convex optimization problem under the restriction of a small amount of fake user injections. This paper serves as a baseline for studying our traditional shilling attacks on Neural Matrix Factorization model. Lin et al. (2020)[9] also provides a description of building a custom shilling attack framework called AUSH using Generative Adversarial Network mainly for deep learning based RS. The model architecture described is hard to implement and computationally expensive but we can borrow some idea from this paper.

Wang et al. (2020) [11] proposed a new model on causal recommender systems which deals with the unobserved confounders that affect both which items the users decide to interact with and how they rate them. They add a phase of exposure model that will decide which item user will interact with before providing predicted ratings for all items. Then the weight from exposure model is applied to the probablistic matrix factorization which we can estimate latent factors from a maximum a posteriori estimator. This weight on exposure inspires our paper and we thought the exposure part will be more robust towards shilling attacks as it is harder for target items to go through the exposure model.

However, all these works does not give a clear comparison of how these traditional (Random, Average, Bandwagon attack), easy to launch attacks models have an impact on traditional neighborhood-based RS model versus matrix-factorization-based RS model versus deep-learning based RS model versus exposure based model. So, in this work we want to create some of the traditional shilling attack and evaluate its impact on different Collaborative Filtering (CF) based RS. Lam et al.(2004)[7] provides a summary of attack on traditional k Nearest Neighbour(kNN) based CF algorithms. We would want to extend this and check the impact of traditional shilling attack model (Random, Average, Bandwagon) on the following 5 categories of collaborative filtering algorithms: User-User kNN based CF, Item-Item kNN based CF, Latent factor based model, Neural Network Matrix Factorization(NMF) and exposure based model on [11]. With this study on these different CF algorithms, we mainly want to answer the following questions:

- Which of the systems are more prone to attack?
- Has these systems over the time become more vulnerable or robust?
- What is the impact of these attacks on an exposure based RS system?

3 Data Set

To accomplish our project, we will be using the MovieLens data set from the GroupLens Research Project at the University of Minnesota. The data set contains data on different movies, movie information, user ratings, and user information.

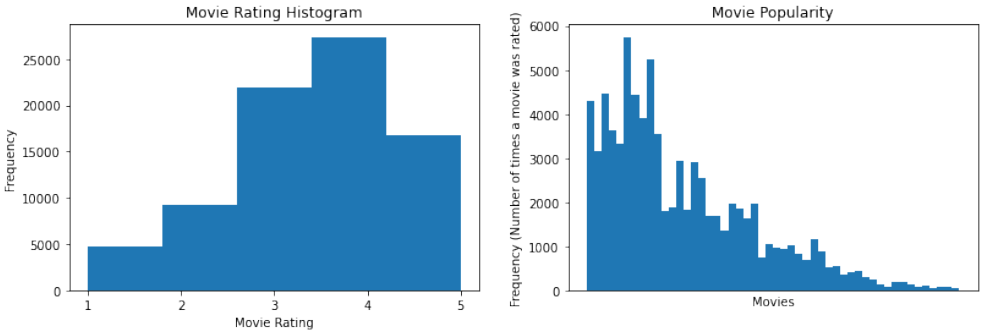
This data set is available at https://grouplens.org/datasets/movielens/, and is available for research uses only. (Harper et al. 2015)[4]

3.1 Data Profile

We used the 100k MovieLens data set. It is noted that the data set contains 100,000 ratings from 1,000 users and 1,700 movies. After opening the data set for ourselves, we had a table of 100,000 rows and 4 columns. The 4 columns are the features of each review: user_id, item_id, rating, and timestamp. Each row is a movie rating with the 4 column features.

After closer inspection of the data set, there were actually 943 unique users and 1,682 unique movies in the 100k data set. We performed an 80/20 train-test split on the data, which results in 80,000 rows

in the training set and 20,000 rows for the testing set. This train-test split has been performed on the data set before hand and the same split will be used for all analysis completed in this project.



The left figure is a Movie Rating Histogram. It shows the distribution of ratings the user's have given for all movies. The right figure is a Movie Popularity histogram. It shows the distribution of how many ratings a certain movie has received.

Running some quick data analysis on the dataset, we found that user's gave ratings in a left skewed distribution. Most user's gave a rating of 4, and there are very few movies that got a rating of 1 in comparison. It is also noted that the number of times a movie is rated (popularity) is also strongly skewed. There are a lot of movies with very few ratings (less than 10, also called long tailed items), meanwhile some movies have a lot of ratings (in the thousands).

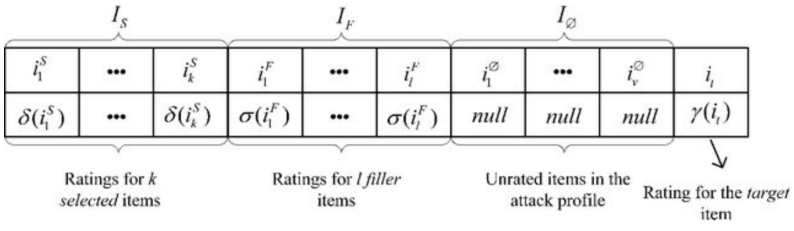
4 Shilling Profile and Attack Models

4.1 Shilling Profile

Let I_S denote the selected item set, I_F the filler item set, I_ϕ the unrated item set, I_T the target item set,and given a Recommendation Problem, a Shilling Profile (SP) is defined as:

$$SP = I_S + I_F + I_\phi + I_T$$

where:
I_S is set of items identified by the attacker to exploit the owned knowledge of system to maximize the effectiveness of the attack; for instance set of popular items.
I_F is set of randomly selected items for which rating scores are assigned from normal distribution of ratings to make the attack imperceptible.
I_ϕ is set of items without ratings in the fake user profile.
I_T is set of target item(s) is to push or nuke.
The SP composition varies based on attack strategies. Below figure represents basic structure of shilling profile:



4.2 Random Attack

Random attack assigns a rating to filler items by randomly choosing a rating from the distribution of ratings in the system. The knowledge required to mount such an attack is quite minimal, as the overall rating mean in many systems can be determined by an outsider(For instance in e-commerce sites the mean rating is available on portal). However, this attack has been found to be particularly ineffective in comparison to others.

4.3 Average Attack

Average attack assigns rating which corresponds to mean rating for $item_i$ across the item data collected. Mean ratings of an item are available on most of the recommendation websites, which can be used by the attacker to initiate this attack. This attack has been tested to be more effective in comparison to Random attack.

4.4 Bandwagon Attack

Bandwagon attack uses the most popular items as the selected items and assigns maximal ratings to them, while fillers are assigned ratings in the same manner as random attack. This attack exploits the pattern in which the popular items are being recommended to users and uses that to recommend the target item to similar users.

4.5 Segmented Attack

Segmented attack assigns maximal ratings to the selected items and minimal ratings to the filler items. This attack targets a segment of users by using the knowledge of what items are preferred by these users and use these items as selected items by assigning maximum rating.

4.6 Sampling Attack

Sampling attack assigns a rating to a filler items by selecting samples from the data. The knowledge required for this attack is quite maximal. This attack requires a lot of ratings from the data set, and that is not easily obtained. This attack is good at simulating real users and very difficult to detect. However, having the generated data being very similar to the original data set, this attack is quite ineffective.

4.7 Attack Summary

Following table shows the attack profiles for each of the different attack models:

Attack Type	Target Item	Filler Item	Selected Item
Random attack	Max Rating	Random ratings	$\emptyset$
Average attack	Max Rating	Average ratings for each item	$\emptyset$
Bandwagon/Popular attack	Max Rating	Random ratings	Max Rating
Segmented attack	Max Rating	Min ratings	Max Rating
Sampling attack	Max Rating	Sampled ratings	$\emptyset$

5 Recommender System Models

5.1 Problem Setup

Let $\mathcal{U}$ be the set of users and $\mathcal{I}$ be the set of items. In an explicit feedback system, we observe some number of entries r_{ui} of the user-item matrix for $u \in \mathcal{U}$ and $i \in \mathcal{I}$ and we want to find out $\hat{r}_{ui}$ to estimate the entries that are not observed. When the direct rating data is hard to collect and we only have data on clicks, likes or other implicit feedback data, an implicit feedback recommender system can also be built. We will mostly focus on explicit feedback recommender system in this paper.

5.2 User-based and Item based KNN model

User-based KNN algorithm finds the k most similar users that have rated the target item within the neighborhood and computes rating of a target item i from user u by using the below mentioned formula:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \in N_i^k(u)} sim(u,v) * (r_{vi} - b_{vi})}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

where $N_i^k(u)$ is k similar neighbours (users) that have rated i; r_{vi} rating of i by each neighbour v. b_{ui} is baseline rating.

Item-based KNN algorithm finds the k most similar items to target item using nearest neighborhood approach and computes rating of a target item i from user u by using the below mentioned formula: Rating of an item i by user u is given by,

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in N_u^k(i)} sim(i,j) * (r_{uj} - b_{uj})}{\sum_{j \in N_u^k(i)} sim(i,j)}$$

where $N_u^k(i)$ is k similar items to item i; r_{uj} rating of similar item j; b_{ui} is baseline rating.

5.3 Latent factor model using ALS (Alternating Least Squares)

One of the most popular and successful methods in collaborative filtering to predict item rating is to use latent factor model based on matrix factorization. The Netflix Competition also shows that this kind of method can be to be more accurate in terms of prediction than the nearest neighbor techniques. (Koren et al. 2009)[6] Given some rank f, the dimension of the latent factor, each item i is associated with an item vector $q_i \in \mathbb{R}^f$ and each user u is associated with a user vector $p_u \in \mathbb{R}^f$. The estimate $\hat{r}_{ui}$ is the dot product of p_u and q_i. Instead of treating the problem as a singular value decomposition problem, which is expensive to compute, we will use an optimization perspective. The goal of the problem is to learn $\{p_u : u \in \mathcal{U}\}$ and $\{q_i : i \in \mathcal{I}\}$ from

$$\min_{p^*, q^* \in \mathbb{R}^f} \sum_{u,i \in \kappa} (r_{ui} - p_u^\top q_i)^2 + \lambda(||p_u||^2 + ||q_i||^2)$$

where κ is the set of (u, i) such that r_{ui} is observed and λ is regularization parameter, p^*, q^* are all the p, q vectors (Koren et al. 2009)[6]. One method to solve the previous optimization problem is Alternating Least Squares (ALS).

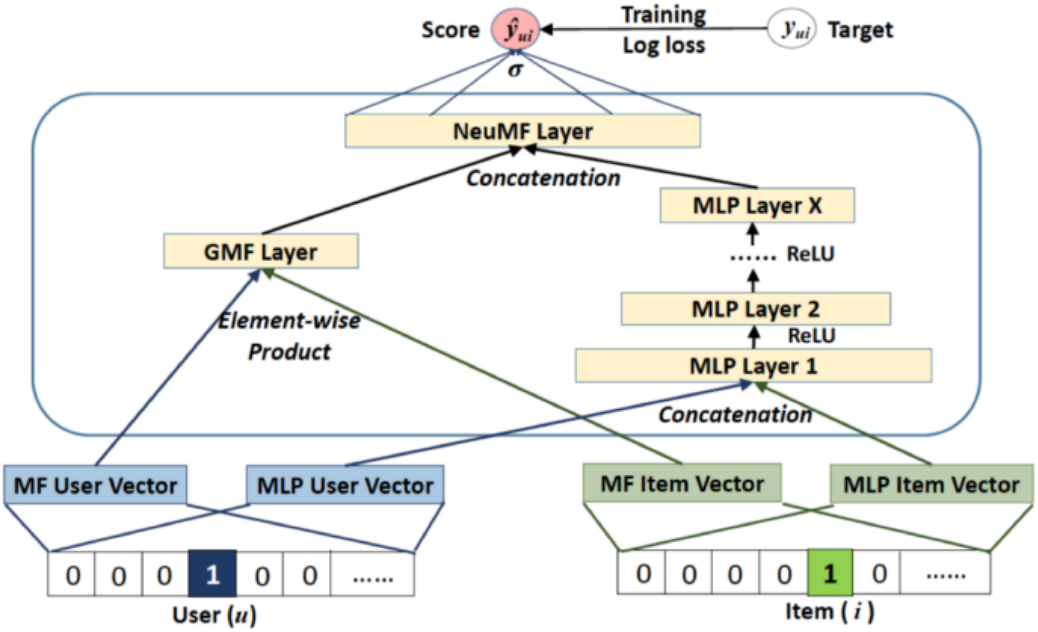
Algorithm 1 ALS algorithm for latent factor model

procedure ALS
  for iteration $\leftarrow$ 1 to MAX_ITER **do**
    for each item i **do** ▷ update item matrix $\mathbf{Q}$
      $r_i \leftarrow$ all the preferences for item i
      $q_i \leftarrow \left(\mathbf{P}^\top \mathbf{P} + \lambda \mathbf{I}\right)^{-1} \mathbf{P}^\top r_i$
    end for
    for each item i **do** ▷ update user matrix $\mathbf{P}$
      $p_u \leftarrow$ all the preferences for item i
      $p_u \leftarrow \left(\mathbf{Q}^\top \mathbf{Q} + \lambda \mathbf{I}\right)^{-1} \mathbf{Q}^\top r_u$
    end for
  end for
  return $\mathbf{X}, \mathbf{Y}$
end procedure

5.4 Neural Matrix Factorization

The Neural Matrix Factorization (NMF) model consists of 3 separate models. First model being Generalized Matrix Factorization (GMF) model that is a linear model that captures the user-item interactions. Second model being Mutliple-layer Perceptron (MLP) model that captures higher dimensional interactions. Finally, the Neural Matrix Factorization (NMF) layer concatenates the GMF and MLP layers and uses a sigmoid activation function to obtain an probability value between 0 and 1 (Xiangnan He et al.)[1]. This probability indicates the likability of an item by user. Values that are closer to 1 mean the user is more likely to like the item, hence to be recommended to the user. Since the model output is a probability but not a rating, we

were unable to compute RMSE (Root Mean Squared Error) and prediction shift for this model.



Neural Matrix Factorization Architecture (Xiangnan He et al.)[1]

5.5 Causal Recommender System

We use a Causal Recommender System proposed by Wang et al. [11]. The main idea is to use deconfounder exposure model prior to latent factor factorization to adjust for the non-uniform distribution of missingness of ratings. We derive our own probablistic matrix factorization and optimization procedure to implement the model. Let $\Omega_{R_{u,i}}$ be the set that ratings are available for user u and item i. We define the loss function of causal model to be

$$L = \frac{1}{2}\left(\sum_{u=1}^{N}\sum_{i=1}^{M}(R_{ui} - U_u^\top V_i - \gamma_u \hat{a}_{ui})^2_{(u,i)\in\Omega_{R_{u,i}}} + \lambda(\sum_u ||u_u||^2 + \sum_i ||V_i||^2)\right)$$

where $\hat{a}_{ui}$ is estimated based on Poisson matrix factorization for exposure and γ_u is new latent parameter adjust for the exposure for each user.The gradient with respect to parameter $\mathbf{U}, \mathbf{V}, \gamma$ are given below:

$$\nabla_{V_i} L = \sum_{u=1}^{N}(R_{ij} - U_u^\top V_i - \gamma_u \hat{a}_{ui})U_u^\top - \lambda V_i$$
$$\nabla_{U_u} L = \sum_{i=1}^{M}(R_{ij} - U_u^\top V_i - \gamma_u \hat{a}_{ui})V_j^\top - \lambda U_u$$
$$\nabla_{\gamma_u} L = \sum_{i=1}^{M}(R_{ij} - U_u^\top V_i - \gamma_u \hat{a}_{ui})\hat{a}_{ui}$$

Setting gradients to zero gives updates on each parameter in matrix form:

$$v_i \leftarrow \left(\mathbf{U}^\top\mathbf{U} + \lambda\mathbf{I}\right)^{-1}(\mathbf{U}^\top r_i - \mathbf{U}^\top(\gamma \circ \hat{a}_i))$$
$$u_u \leftarrow \left(\mathbf{V}^\top\mathbf{V} + \lambda\mathbf{I}\right)^{-1}(\mathbf{V}^\top r_u - \gamma_u \mathbf{V}^\top \hat{a}_u)$$
$$\gamma_u \leftarrow (r_u^\top \hat{a}_u - \mathbf{V}\mathbf{U}_u^\top \hat{a}_u)/(\hat{a}_u^\top \hat{a}_u)$$

Algorithm 2 ALS algorithm for causal recommender

Given: $\hat{a}_{ui}$ for all user u and item i. All updates are restricted to $\Omega_{R_{u,i}}$.
procedure ALS
  for iteration $\leftarrow$ 1 to MAX_ITER **do**
    for each item i **do** ▷ update item matrix $\mathbf{V}$
      $r_i \leftarrow$ all the ratings for item i
      $\hat{a}_i \leftarrow$ all the estimated confounders for item i
      $\gamma \leftarrow$ parameter vector
      $v_i \leftarrow \left(\mathbf{U}^\top \mathbf{U} + \lambda \mathbf{I}\right)^{-1} \left(\mathbf{U}^\top r_i - \mathbf{U}^\top (\gamma \circ \hat{a}_i)\right)$(∘: element-wise product)
    end for
    for each user u **do** ▷ update user matrix $\mathbf{U}$
      $r_u \leftarrow$ all the ratings for user u
      $\hat{a}_u \leftarrow$ all the estimated confounders for user u
      $\mathbf{U}_u \leftarrow$ is the u^{th} row of matrix $\mathbf{U}$
      $u_u \leftarrow \left(\mathbf{V}^\top \mathbf{V} + \lambda \mathbf{I}\right)^{-1} \left(\mathbf{V}^\top r_u - \gamma_u \mathbf{V}^\top \hat{a}_u\right)$
      $\gamma_u \leftarrow (r_u^\top \hat{a}_u - \mathbf{V}\mathbf{U}_u^\top \hat{a}_u)/(\hat{a}_u^\top \hat{a}_u)$
    end for
  end for
  return $\mathbf{U}, \mathbf{V}$
end procedure

6 Evaluation metrics

Let U bet set of users, I be set of items, then Root Mean Squared Errror (RMSE), Prediction Shift (PS), and Hit Ratio for top K items are defined as follows:

6.1 Root Mean Squared Error

For each user-item pair (u, i) the real rated value in the test set is denoted by $p_{u,i}$ and the predicted rated value by the model is $\hat{p}_{u,i}$. Then the formula for RMSE is given as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(p_{u,i} - \hat{p}_{u,i})^2}{N}}$$

Where N is the total number of real rated values in the test set. This metric is the standard deviation of the prediction errors, which is used to measure how well a recommender model predicts.

6.2 Prediction Shift

For each user-item pair (u, i) the prediction shift denoted by $\triangle_{u,i} = p'_{u,i} - p_{u,i}$, where p' and p denotes prediction after and before attack. A positive value means that the attack has succeeded in making the pushed item more positively rated. Then, average prediction shift for an item i over all users is given as:

$$\triangle_i = \sum_{u \in U} \frac{\triangle_{u,i}}{|U|}$$

Average prediction shift for all items tested is given as: $\triangle = \frac{\triangle_i}{|I|}$. The prediction shift measures how much the prediction values have changed after the attack compared with the prediction before attack.

6.3 Hit Ratio for top K items

Let R_u be the set of top K recommendations for each user u. For each push attack on item i, the value of a recommendation hit for user u denoted by H_{ui}. Value of H_{ui} is 1 if $i \in R_u$; otherwise H_{ui} is 0. Hit ratio is defined as the number of hits across all users in the test set divided by the total number of users in the test set. Then, hit ratio for a pushed item i over all users for K top recommendations is:

$$HitRatio_i@K = \sum_{u \in U} \frac{H_{ui}}{|U|}$$

Average hit ratio across all target items is calculated as the sum of the hit ratio for each target item i after divided by the total number of items in test set:

$$HitRatio@K = \frac{HitRatio_i@K}{|I|}$$

This metric corresponds to the percentage users to whom one of the target item will be recommended among top 10 after the attack.

7 Results and Findings

7.1 Experimental Setup

We setup 5 different sets of target items for our attacks: **items with High Rating but Low Rating Count (long tail items), Low Rating but Low Rating Count (long tail items), Low Rating but High Rating Count, Randomly chosen target items, and Segmented target items**. The items with low rating count utmost one or two user ratings. Each target item set consists of three target items that belong in their category. For example, the High Rating Low Count set consists of three target items that have very high rating but low number users have rated this items hence low rating count and vice versa. The Random target item set consists of three randomly chosen items. The segment target item set consists of three items that were chosen to be closely related items, i.e items which belong to a particular segment. These items were chosen using cosine similarity of the genres of the movies. These target items are then used in created attack data or fake profiles.

For all 5 recommender system algorithms, we created models trained on attack data with target items pertaining to all 5 different sets of target items, totaling up to 20 trained models per algorithm. In addition, we created 5 base models (one without attack data) for each algorithm to get base ratings. With each trained attack model, we obtained 3 metric values: RMSE, Prediction Shift, and Hit Ratio. Also note that the Hit Ratio was calculated using the top 10 recommendations of a user. Neural Matrix Factorization outputs a probability value rather than rating so we were only able to get the hit ratio. The causal model is also biased towards randomly splitting data. So the RMSE and Prediction shift does not make sense. Our experiment results with attack size of 50 (5% of total users) fake profiles are shown below:

7.2 Experimental Results

		High Rating, Low Count			Low Rating, Low Count		
		RMSE	PredShift	HitRatio	RMSE	PredShift	HitRatio
Latent Factor Model	Average	0.927	0.0654	0.939	0.928	0.0277	0.848
	Bandwagon	0.926	0.0967	0.575	0.927	0.0787	0.121
	Random	0.928	-0.171	0.719	0.929	-0.173	0.339
	Sampling	0.928	-0.0613	0.0450	0.928	-0.0659	0.00435
User KNN	Average	0.934	0.508	0.602	0.933	2.398	0.687
	Bandwagon	0.933	0.562	0.559	0.935	2.39	0.632
	Random	0.933	0.515	0.573	0.933	2.210	0.629
	Sampling	0.933	0.433	0.395	0.933	1.061	0.204
Item KNN	Average	0.936	0.712	0.404	0.936	1.35	0.542
	Bandwagon	0.936	0.724	0.416	0.936	1.33	0.584
	Random	0.937	0.644	0.366	0.937	1.25	0.495
	Sampling	0.939	0.618	0.302	0.939	1.21	0.411
NMF	Average	-	-	0.0	-	-	0.0
	Bandwagon	-	-	0.0	-	-	0.0
	Random	-	-	0.0	-	-	0.0
	Sampling	-	-	0.0	-	-	0.0
Causal	Average	-	-	0.0319	-	-	0.0322
	Bandwagon	-	-	0.0326	-	-	0.0366
	Random	-	-	0.0487	-	-	0.0450
	Sampling	-	-	0.0138	-	-	0.0211

		Low Rating, High Count			Random		
		RMSE	PredShift	HitRatio	RMSE	PredShift	HitRatio
Latent Factor Model	Average	0.928	-0.0263	0.0137	0.928	-0.0135	0.159
	Bandwagon	0.926	0.0940	0.0290	0.929	0.105	0.0530
	Random	0.929	-0.197	0.0246	0.929	-0.200	0.0885
	Sampling	0.928	-0.0445	0.0181	0.928	-0.0634	0.00798
User KNN	Average	0.933	0.188	0.0	0.934	0.754	0.130
	Bandwagon	0.935	0.089	0.0	0.934	0.661	0.132
	Random	0.933	0.139	0.0	0.933	0.675	0.152
	Sampling	0.933	0.027	0.0	0.933	0.200	0.038
Item KNN	Average	0.936	0.304	0.011	0.937	0.785	0.169
	Bandwagon	0.937	0.289	0.013	0.937	0.818	0.271
	Random	0.938	0.301	0.015	0.937	0.771	0.21
	Sampling	0.939	0.25	0.007	0.939	0.673	0.173
NMF	Average	-	-	0.00363	-	-	0.0254
	Bandwagon	-	-	0.00363	-	-	0.0131
	Random	-	-	0.00363	-	-	0.0181
	Sampling	-	-	0.00363	-	-	0.0181
Causal	Average	-	-	0.0574	-	-	0.0426
	Bandwagon	-	-	0.0503	-	-	0.0373
	Random	-	-	0.0520	-	-	0.0470
	Sampling	-	-	0.0500	-	-	0.0272

The two tables above summarize our findings of attack performance. The table is structured in an hierarchical view. From the left, we have the 5 different models: Latent Factor Model, User-based KNN, Item-based KNN, Neural Matrix Factorization (NMF), and Causal Model. Each model was tested against the 4 different attacks: Average, Bandwagon, Random, Sampling. From the top of the table, we have 4 different target item sets: High Rating Low Count, Low Rating Low Count, Low Rating High Count, and Random. Each of the 4 target item sets have 3 metrics we used to evaluate the attacks: RMSE, Prediction Shift for All Users, and the Hit Ratio of the Target Items. These numbers are for attack size of 50 i.e by adding 50 fake profiles. (5% of total users)

		High Rating, Low Count					
		50 Target Size			100 Target Size		
		RMSE	PredShift	HitRatio	RMSE	PredShift	HitRatio
Latent Factor Model	Average	0.927	0.0654	0.939	0.929	0.107	0.924
	Bandwagon	0.926	0.0967	0.575	0.928	0.130	0.466
	Random	0.928	-0.171	0.719	0.931	-0.276	0.750
User KNN	Average	0.934	0.508	0.602	0.934	0.793	0.830
	Bandwagon	0.933	0.562	0.559	0.937	0.839	0.741
	Random	0.933	0.515	0.573	0.934	0.768	0.787
Item KNN	Average	0.936	0.712	0.404	0.936	1.117	0.692
	Bandwagon	0.936	0.724	0.416	0.940	1.083	0.692
	Random	0.937	0.644	0.366	0.938	0.985	0.570

Attack Size Table: The table above are the results for attack size 50 and 100 using the High Rating Low Count target item set. There is another table using the Random target item set in the appendix.

7.3 Findings and Discussion

7.3.1 Findings on Models

We see that KNN and latent factor models are the ones most susceptible to attacks. User based KNN, item based KNN, latent factor model have an average hit ratio across all attacks as 0.53, 0.37, 0.57 respectively for high rating low rating count target item set category, which for given test users the target items showed up in top 10 results 53%, 37% and 57% respectively for these models across all attacks. Where as neural network based recommender system, Neural Matrix Factorization is really robust to attacks, as it has almost 0% hit ratio across all attack types and target item sets. Even for low rating average high count target (LRHC) item set KNN based models have a 0.5 hit ratio.

This basically shows that KNN is most vulnerable to attacks among all the models. The Causal Recommender model is more robust than KNN and Latent Factor models and achieves similar hit ratio for all attacks. It is most robust against sampling attack and performs well on low rating low count items.

7.3.2 Findings on Attack Type

We found that average attack performs better than all the other attack types. For long tail target item set (target item with low count high rating) Average attack always has hit ratio greater than 0.55 which means it recommended one of the target to users 55% of the times. It is also notable that sampling and segment attack perform poorly across all models and target item set categories. Their hit ratio and prediction shifts are not significant which indicates these attacks are poor in getting target items in the top 10 recommendations for many users.

7.3.3 Findings on Target Items Sets

In our results target item set with High Rating Low Count always has higher hit ratio compared to other target items. This was the case for Average, Random and Bandwagon attack type. This shows that it is easy to promote a target item with high average rating but low rating count (count <= 2) for KNN based and latent factor based recommender system. On the contrary low rating average and low rating count target items have less than 0.05 hit ratio i.e less than 5% chances of getting recommended to users across all model types.

Random target items yield relatively lower hit ratio's, and mostly hovers around $5\% - 20\%$. Due to the nature of randomness of the target items, there is no guarantee the hit ratio numbers will be consistent.

7.3.4 Findings on Attack Size

As expected, the bigger the attack size resulted in higher the hit ratio. KNN based and latent factor models for all attacks has a significant increase from $5\% - 30\%$ increase in the hit ratio with the attack size of 100 (10% of total users). In general, it would be a good idea for an attacker to inject more fake user profiles. In real world scenario it is expensive and not easy to inject higher number of fake profile into a system, but in our experimental setup we were able to evaluate the impact of attack size.

8 Conclusion

Among all the models KNN and Latent factor model were the one more susceptible to attack. Among the attack types, average attack seemed to have most impact in terms of hit ratio and promoting a low rating count or long tail target item. Also, our findings conclude that the neural network model are more robust in general. Hence, there are many work going on to create custom models to attack these neural based recommender systems. If we took causal model as an improvement of the widely used latent factor model, it is more robust than latent factor models and maintains interpretability of the model.

References

[1] Xiangnan He et al. “"Neural collaborative filtering."”. In: *Proceedings of the 26th International Conference on World Wide Web*. (2017).

[2] Zi-Jun Deng, Fei Zhang, and Sandra PS Wang. “Shilling attack detection in collaborative filtering recommender system by PCA detection and perturbation”. In: *2016 International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR)*. IEEE. 2016, pp. 213–218.

[3] Gintare Karolina Dziugaite and Daniel M. Roy. *Neural Network Matrix Factorization*. 2015. arXiv: 1511.06443 [cs.LG].

[4] F. Maxwell Harper and Joseph A. Konstan. “The MovieLens Datasets”. In: 4 (2015). URL: http://dx.doi.org/10.1145/2827872.

[5] Hai Huang et al. “Data Poisoning Attacks to Deep Learning Based Recommender Systems”. In: *arXiv preprint arXiv:2101.02644* (2021).

[6] Yehuda Koren, Robert Bell, and Chris Volinsky. “Matrix factorization techniques for recommender systems”. In: *Computer* 42.8 (2009), pp. 30–37.

[7] Shyong K. Lam and John Riedl. “Shilling Recommender Systems for Fun and Profit”. In: *Proceedings of the 13th International Conference on World Wide Web*. WWW ’04. New York, NY, USA: Association for Computing Machinery, 2004, pp. 393–402. ISBN: 158113844X. DOI: 10.1145/988672.988726.

[8] Xinxin Niu Li Yang Wei Huang. “Defending shilling attacks in recommender systems using soft co-clustering”. In: *IET Information Security* (2017).

[9] Chen Lin et al. “Attacking Recommender Systems with Augmented User Profiles”. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2020, pp. 855–864.

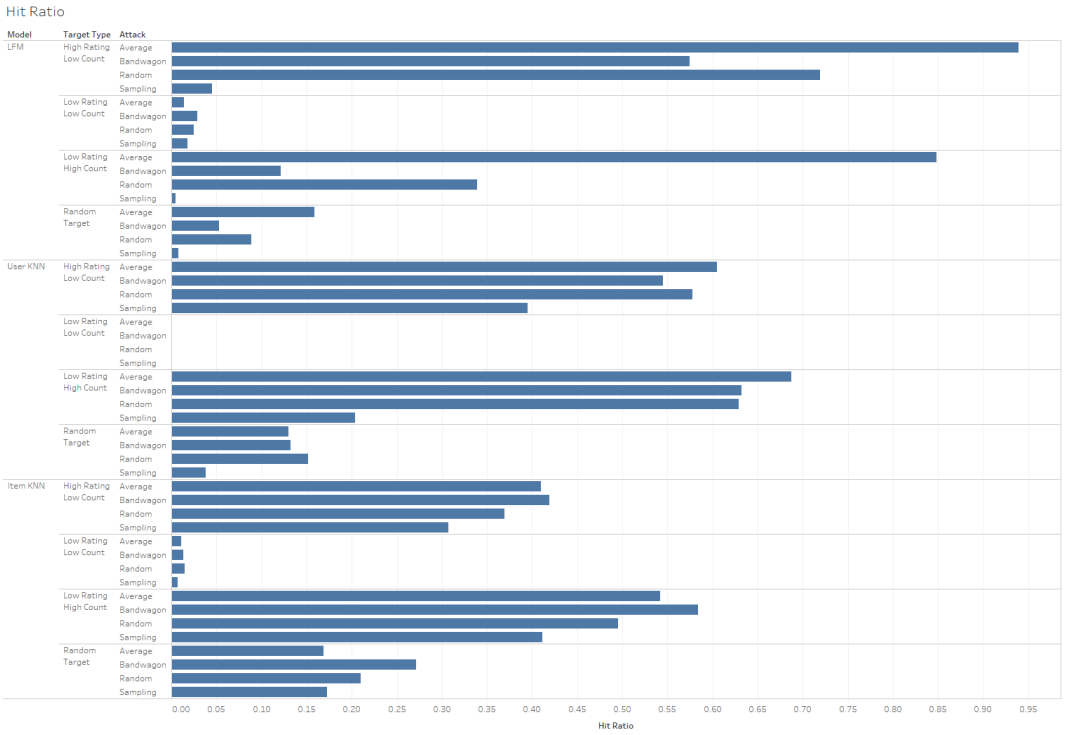
[10] J. J. Sandvig, Bamshad Mobasher, and Robin Burke. “Robustness of Collaborative Recommendation Based on Association Rule Mining”. In: *Proceedings of the 2007 ACM Conference on Recommender Systems*. RecSys ’07. New York, NY, USA: Association for Computing Machinery, 2007, pp. 105–112. ISBN: 9781595937308. DOI: 10.1145/1297231.1297249. URL: https://doi.org/10.1145/1297231.1297249.

[11] Yixin Wang et al. “Causal Inference for Recommender Systems”. In: *Fourteenth ACM Conference on Recommender Systems*. 2020, pp. 426–431.

[12] Xingyu Xing et al. “Take this personally: Pollution attacks on personalized services”. In: *22nd {USENIX} Security Symposium ({USENIX} Security 13)*. 2013, pp. 671–686.

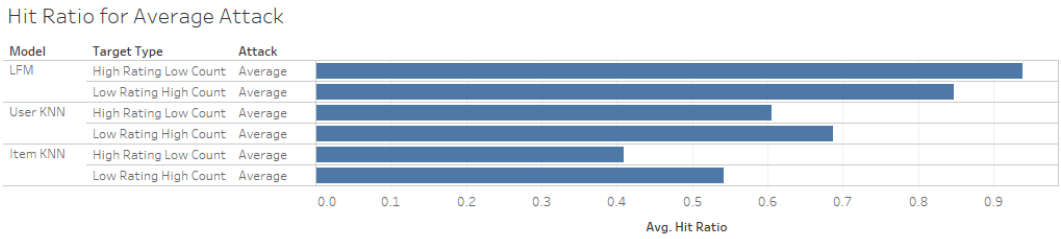
Appendix

Hit Ratio



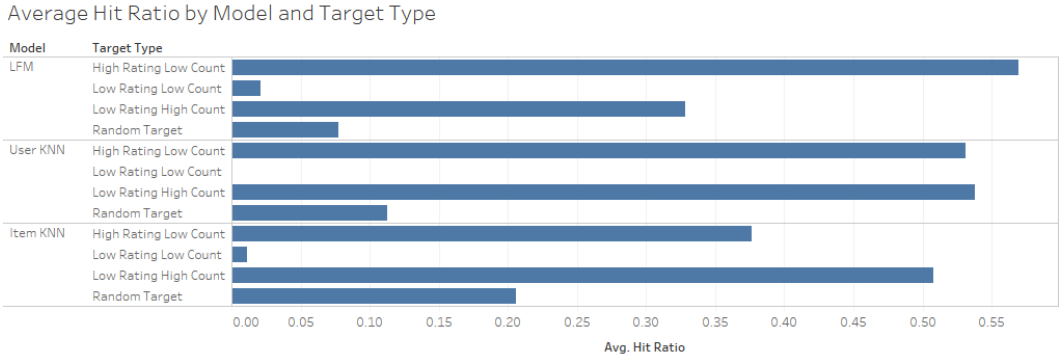
Hit Ratio for each Attack broken down by Model, Target Type, and Attack Type.

Hit Ratio for Average Attack



Average of Hit Ratio for each Attack broken down by Model, Target Type, and Average Attack.

Average Hit Ratio by Model and Target Type



Average of Hit Ratio for each Target Type broken down by Model.

Metrics for Segmented attack across all models

The table above are the results for the models against the segment attack and segment chosen target items.

	Segment Target Items		
	RMSE	PredShift	HitRatio
Latent Factor Model	0.927	-0.0687	0.0833
User KNN	0.932	-0.026	0.002
Item KNN	0.941	0.005	0.001
NMF	-	-	0.00382
Causal	-	-	0.0607

Hit ratios for Random target items for attack size = 100

The table above are the results for attack size 50 and 100 using the Random target item set.

		Random					
		50 Target Size			100 Target Size		
		RMSE	PredShift	HitRatio	RMSE	PredShift	HitRatio
Latent Factor Model	Average	0.928	-0.0135	0.159	0.930	0.0924	0.296
	Bandwagon	0.929	0.105	0.0530	0.931	0.106	0.0827
	Random	0.929	-0.200	0.0885	0.931	-0.271	0.162
User KNN	Average	0.934	0.754	0.130	0.934	0.917	0.238
	Bandwagon	0.934	0.661	0.132	0.935	0.816	0.207
	Random	0.933	0.675	0.152	0.934	0.809	0.230
Item KNN	Average	0.937	0.785	0.169	0.937	0.973	0.336
	Bandwagon	0937	0.818	0.271	0.938	0.986	0.400
	Random	0.937	0.771	0.21	0.939	0.939	0.313