Efficiency of Shilling Attacks in Modern Recommenders

Brian Murtagh
Pontificia Universidad Católica de Chile
bmurtagh@uc.cl

Pablo Brancoli
Pontificia Universidad Católica de Chile
pbrancoli@uc.cl

Sergio Gazali
Pontificia Universidad Católica de Chile
sagazali@uc.cl

Abstract

Among the various problems of study and social impact regarding recommender systems, there are attacks aimed at biasing their results in favor or against certain items. In particular, shilling attacks are studied, which consist of the insertion of false profiles and ratings to the database.

Various attack strategies are implemented to know their effectiveness in promoting the recommendations of an item by a set of modern recommenders. An attacker's goal is to manipulate a recommender system such that the attacker-chosen target items are recommended to many users. To achieve this goal, our attack injects fake users with carefully crafted ratings to a recommender system. We developed different algorithms of attacks in search of what algorithm is the better and which recommender system is the most affected by the attacks.

Our experimentals results on Movielens dataset show that some attacks are most effective than anothers, and that more complex recommender systems like neuronal networks, are more resistants to attacks. All these conclusions are based on the HR@10 metric, which indicates the percentage of real users, to whom the target item is recommended within the top k selected items.

Keywords

recommender systems, shilling attacks, algorithms

ACM Reference Format:
Brian Murtagh, Pablo Brancoli, and Sergio Gazali. 2021. Efficiency of Shilling Attacks in Modern Recommenders. In *RecSys: Term Project, December 13, 2021, Santiago, Chile.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/1234567.1234567

1 State of the art

The goal of this investigation is to provide a general understanding of the impact that different attacks can have on a set of modern recommender systems. Thus, it is necessary to establish a basic understanding of this two elements.

RecSys, 2021-2, Santiago, Chile

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1234567.1234567

Recommender systems are algorithms designed to facilitate the access to information, by filtering from a wide set of items, a selected subset based on a specific criteria. Modern recommender systems aim at handing the users personalised content which best fit their preferences and necessities. The goal is to model, based on past interactions, the user's interest to recommend the content that maximizes his or her utilities. Traditional recommender systems include neighborhood-based, matrix factorization based and graph based models. Though in recent years neural network based recommender systems are gaining popularity given their nonlinear transformation and representation learning capacities. Deep learning is becoming a technology trend in the field.

However, recommender systems introduce several challenges and risks, often with implication at a social level. One of those is their vulnerability to attacks. An attack on these seeks to 'push' or 'nuke' an item's recommendations. This is to increase or decrease, respectively, the likelihood of a certain item of being recommended by the system. A common way to perpetrate this is by implementing a 'data poisoning attack', also known as 'shilling attack'. Shilling attacks consists on the creation of fake users on the target system, to then insert fake ratings or interactions on specific items of the dataset. Different algorithms implement different strategies by which they select the items to interact and the type of interaction to have with each item.

A specific set of recommender systems and shilling attacks were selected to perform this investigation. The recommender systems were provided in a library implemented by Microsoft, and each of the attack algorithms were implemented by the development team. Below, a description of each of these can be found.

1.1 Recommender Systems

- NCF[5]: Neural Collaborative Filtering is a Deep learning algorithm with enhanced performance for implicit feedback. Replaces the user-item inner product of a normal Collaborative Filtering algorithm with a neural architecture.
- RBM: Restricted Boltzmann Machines is a neural network based algorithm for learning the underlying probability distribution for explicit or implicit feedback.
- RLRMC[6]: Riemannian Low-rank Matrix Completion is a matrix factorization algorithm that uses Riemannian conjugate gradients optimization for small memory consumption.

1.2 Shilling Attack algorithms[3,4]

- Most Popular: This algorithm is based on the idea that creating a new user and giving positive rankings to the most

popular items in the data set, increases the probability that the user resembles other users, having an impact on their recommendations. In addition, it is proposed to assign ratings to items called fillers (I_f), chosen randomly. These are assigned with $[r_{min}, 2]$, which vary depending on their average ratings. What is sought is to generate correlations of ratings with other users.

Algorithm 1 Most Popular Attack

```
Input: T top most popular items from the data set, X target item,
  F filler items, P new users and dataset
  for userID in range(P) do
      for item in T do
          dataset append ('userID': userID,'itemID': item ,'rating':
  4)
      end for
      dataset append ('userID': userID,'itemID': X ,'rating': 5)
  end for
  randomSet = userIds − T − X
  randomSet = random.sample(randomSet, F)
  for userID in range(P) do
      for item in randomSet do
          if rating mean of item > 3) then
              dataset append ('userID': userID,'itemID': item ,'rat-
  ing': 2)
          else
              dataset append ('userID': userID,'itemID': item ,'rat-
  ing': 1)
          end if
      end for
  end for
  ///
  Train Model
  ///
```

Algorithm 2 Probe Attack

```
Input: T top most popular items from the data set, X target item,
  F filler items, P new users and dataset
  for userID in range(P) do
      for j in range(5) do
          randomItem = random.choice(T)
          dataset append ('userID': userID,'itemID': randomItem
  ,'rating': 4)
      end for
      dataset append ('userID': userID,'itemID': X ,'rating': 5)
  end for
  ///
  Train Model
  ///
  ranked = model predict n items
  for row in ranked) do
      dataset append ('userID': row.userID,'itemID': row.itemID
  ,'rating': rating mean of the item in the initial dataset)
  end for
  randomSet = userIds − T − X
  randomSet = random.sample(randomSet, F)
  for userID in range(P) do
      for item in randomSet do
          if rating mean of the item in the initial dataset > 3 then
              dataset append ('userID': userID,'itemID': item ,'rat-
  ing': 2)
          else
              dataset append ('userID': userID,'itemID': item ,'rat-
  ing': 1)
          end if
      end for
  end for
  ///
  Train Model
  ///
```

- Bandwagon: This algorithm follows the same idea of the Most Popular Attack, with the only difference that filler items are assigned with random rankings.

- Average: This algorithm also follows the same idea of the Most Popular Attack, but unlike this one, it only ranks well the target item, with r_{max}. On the other hand, the fillers items are ranked with the average of their previous rankings.

- Probe Attack: This attack adds base items randomly gathered from the most popular items to the attack users, to later obtain recommendations from the same system. Then it incorporates rankings of these recommendations to attack users.

2 Dataset

We used the Movielens dataset for the investigation. This is a traditional dataset in the field of recommender systems, but it was still functional to the purpose of this work. Given that the study is aimed at attack algorithms and the impact that they have on different systems, the dataset was an instrument to carry on the attacks. In other words, it was not of interest to innovate on the dataset, or to apply the algorithms to a new industry or specific field. Because of this, we chose a known database to serve as a means to develop the study without the introduction of major external variables to the results.

The main statistical indicators that describe the database are exposed in table 1.

3 Research Methodology

The methodology used to achieve the goals of this investigation was, in broad terms, to search and read recent papers and online resources, then select specific attack algorithms and recommender systems against which we would perform the attacks, and finally, a stage of parameter tuning to get results of interest. All the attacks performed were 'push attacks' that had the same target item whose

recommendations were intended to be increased.

The consulted papers served as a means to understand the state of the art, and different attack techniques. This stage of the investigation also included the search of implemented code online, mainly on repositories published by authors of investigations in the field. This helped us define the aim of our investigation and make use of already implemented technologies, as the recommender systems used for our study.

After the definition of the recommender systems and attack algorithms to investigate, the first implementation took place. We developed the code for a Popular Attack against an NCF system. We then performed a sensibility analysis of the parameters of this attack, which led to an optimal configuration that generated the biggest impact on the recommendation. Note that this configuration is not intrinsic to the attack algorithm, but is also defined by the recommender that is under attack.

The next step was to iterate and perform the same exercise of implementation and tuning for each attack algorythm, but this time doing so against each selected recommender system.

After this, we carried out an attack-size analysis. The way to do this was to select, for each recommender system, the best configuration of each attack, and then measure it's impact depending on the amount of fake users injected. This fake population was tested with a size of 1%, 3%, 5% and 10% of the total population.

This process, allowed for a comprehensive analysis of the parameters of the attacks and their impact on it's results, as well as a general overview of the different algorithms and their capacity to perform effective 'push' attacks against a set of recommender systems.

3.1 Targeted Item

For the evaluation of shilling attacks is custom to pick a set of target items by random or unpopular choice. Random target items are sampled uniformly at random from the whole item set, while unpopular items are collected randomly from those items with low amount of interactions or reviews. In the following experiments we decided that the target item across the experiments was fixed and single, rather than having a set of multiple target items, so the various results are more comparable with each other. Therefore, all the attacks conducted tried to push the same movie. We selected an average item ($id = 3$) with 64 reviews which average out to a rating of 3.1.

Table 1: Database Statistical Indicators

Indicator	Value
Distinct users	943
Distinct items	1597
Mean rated items by user	46.9
Mean amount of reviews per item	79.5
Standard deviation rated items by user	75.7
Mean rating users by item	46.9
Standard deviation rating users by item	68.3
Dataset sparsity	0.9502

3.2 Metric[1]

To measure all the different attack settings and scenarios, we used a standard metric, that was also used in the papers that were consulted in the first phase of our investigation. This metric is 'Hit Ratio at k' (HR@k). It indicates the percentage of real users, to whom the target item is recommended within the top k selected items. Therefore, let R_u^k be the set of top k recommendations for a user u. If the target items appears in R_u^k, for user u, the scoring of $HR@k$ has a value of 1; otherwise it's 0. Then the equation it's as follows:

$$HR@K = \sum_{u \in U_T} \frac{H_k}{|U_T|}$$

For this investigation, we have set k to 10 ($HR@10$), given the trend among similar investigation, so result comparison is easier.

For comparison reasons, a Baseline amount is given which portraits the HR@10 the target item would have gotten if no attack was ever done over the system, allowing to compare more easily the results across multiple recommender systems.

4 Parameter Analysis

For the purpose of this research, we did not focused on the optimization of the recommender systems, so we maintain their parameters as given by Microsoft for the purpose of movie recommendation. In the other side, the parameters of the attacks were tuned by brute force of multiple values. We used three main parameters over the attacks: the first being size of the attack, which means how many new users are we creating, and it varies from 1, 3, 5, 10% of the initial population, we chose these values because they represent a representative attack size and not so crazy for a large data set.

- Attack Size: It defines how many fake users the attack is going to add to the system, respect to the total amount of real users in the dataset. It is intuitively that the bigger the attack size, the more fake users are in the system, so the greater the impact of the attack over the recommender system. We chose our attack size in a range of 1, 3, 5, 10% of the initial population, mainly because this set of attack sizes represent a possible scale in a real case scenario.

- Top K: It represents how many items are selected from the top rated items from the dataset to then be incorporated in the rankings of the fake users, this set is represented as I_S. It varies from 5, 10 and 20. Inside research showed that larger values tended to opaque the target item and the variance in the top items would increase, so it would not generate as much correlation. Average attack does not have I_S.

Table 2: Parameter tuning with HR@10 at attack size of 3%.

Attack Type	Filler Size	Recommender System								
		NCF			RBM			RLRMC		
		Top 5	Top10	Top 20	Top 5	Top10	Top 20	Top 5	Top10	Top 20
Popular	0	0,53%	0,53%	0,53%	0,74%	0,53%	1,38%	2,65%	2,86%	**3,61%**
	10	0,85%	0,85%	0,85%	0,74%	0,53%	0,95%	2,12%	2,23%	1,91%
	20	**1,48%**	1,48%	1,48%	1,59%	**1,91%**	1,70%	1,70%	1,59%	2,01%
Bandwagon	0	0,53%	0,53%	0,53%	0,32%	0,32%	0,32%	1,80%	1,91%	0,42%
	10	0,85%	0,85%	0,85%	0,32%	0,21%	0,21%	1,38%	1,91%	0,42%
	20	**1,48%**	1,48%	1,48%	**1,27%**	0,85%	0,95%	**2,76%**	2,97%	0,42%
Average	0		**1,06%**			0,42%			**3,39%**	
	10		0,95%			0,64%			3,08%	
	20		0,32%			**0,95%**			1,91%	
Probe	0	0,42%	0,53%	**1,38%**	0,64%	0,74%	**1,70%**	**4,35%**	4,35%	3,08%
	10	0,11%	0,21%	0,74%	0,21%	0,53%	0,74%	0,85%	1,17%	1,06%
	20	0,32%	0,21%	1,06%	0,53%	0,53%	1,27%	0,95%	0,32%	0,32%
Baseline			0,32%			0,74%			1,06%	

- Filler Size: This parameters represents how many items are selected as filler as previous studies has shown[1], 2 that adding a random set of items helps incorporate the fake users in the system without diluting much its impact. This parameter is represented by I_F and it varies from 0, 10 and 20, mainly given that in Top K we occupy values that are not so large since they would affect the correlation.

It's important to mention that there is a difference between the probing attack and the rest, which has an impact on the Top K parameter. In this case, Top K does not determine the amount of popular items to rate, but the amount of top recommended items that are taken by each fake user to then rate, replicating the behavior of a real user.

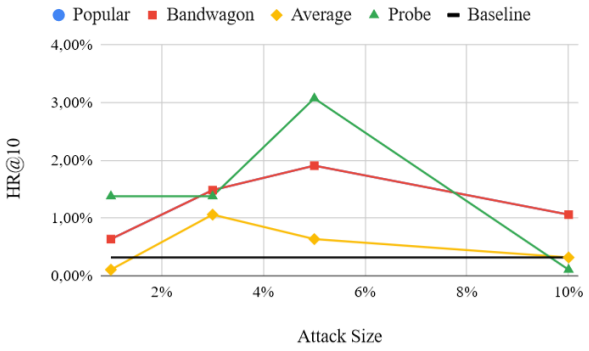

Figure 1: HR@10 for different attack sizes against NCF

5 Results

The experiments described in section 3 resulted in a significant amount of configurations and attack scenarios. All of them were measured using HR@10, and the results were condensed in a Table 2, for the parameter tuning of each attack, and three plots, which illustrate the attack size variations for the best configurations of each attack algorithm (one plot for each recommender system).

Attacks over NCF

In Table 2, it's possible to appreciate that there is not a clear relevance of I_S size respect to I_F. Popular and Bandwagon attacks preferred a high-filler, low-popular configuration reaching the same results, meaning the difference in filler ranking assignment did not provided much difference. Average configuration is as expected with no filler items, giving that the ranking of random items can disperse a user impact. The Prove attack chose a empty filler set as well and high amount of recommended items. The best attack is Most Popular with an efficiency of 1.16%, given the baseline of 0,32%.

In Figure 1, we cannot appreciate Popular attack given that it obtain the same results as Bandwagon. The best attack is not clear, as Prove attack reached a 3% HR@10 in an attack size of 5%, showing a high sensitivity over the amount of fake users, and counter-intuitively as the bigger the attack size, the hit rates get lower with Prove attack plumbing below the baseline.

Attacks over RBM

For the RBM system, the first three attacks have a clear preference over $I_F = 20$, which could imply in a better integration in the system with more random recommendations. An eye-catcher is the preference of the Average attack of a high-filler configuration, giving a clue that filler items are very relevant in RBM systems. The Prove attack chose a empty filler set again as well and high amount of recommended items. The best attack is Most Popular with an efficiency of 1.17%, given the baseline of 0,74%, very similar to NCF results.

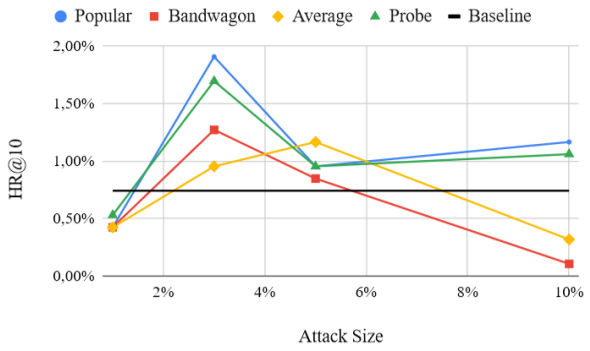

Figure 2: HR@10 for different attack sizes against RBM

The results shown in Figure 2 are widely different from the ones shown in the previous section. All attacks preform worse than the baseline in the attack size of 1%, then it reaches a high point at 3%. From there on, Average and Bandwagon drop below baseline, and Popular and Probe had a big drop but this time they maintain above base-line

Attacks over RLRMC

For the RLRMC system, Popular, Average and Probe attack have a preference over $I_F = 0$ which contradicts what we thought, but in Bandwagon there is a preference over $I_F = 20$, so is a strange phenomenon, given Bandwagon and Popular similarity . It is also curious to analyze that the bandwagon with $I_S = 20$ does not change as the item fillers change.

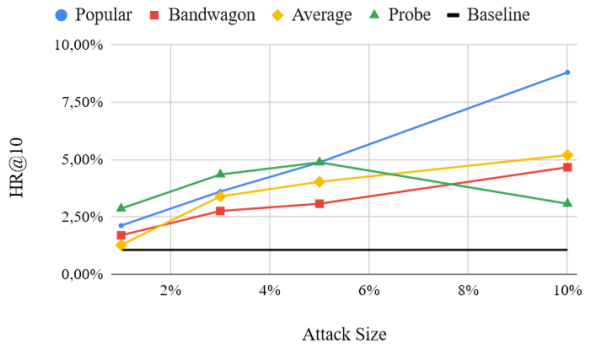

Figure 3: HR@10 for different attack sizes against RLRMC

Finally in Figure 3, the RLRMC was the one that returned the best results, reaching a global maximum of 8.8% in the case of Most Popular. Here the results finally take the shape we expected, a steady increase of all attacks as the attack value is bigger, except for Prove attack.

6 Conclusions

From the experiment results, we can observe that the various attacks preform very different over the 3 recommender systems. We were expecting that as the attack size got bigger, the higher the impact the attack would have in the system; but our results shows that that's not the case in NCF and RBM, as most attacks at 10% size decrease considerably.

As we can see, the most affected recommender was the RLRMC, reaching up to 8.8% of HR@10, this falls on the nature of the system, which due to the fact that it is a matrix factorization type recommender, changes in the matrix weights maybe affect the total output.On the other hand, NCF and RBM are less affected, this is because they are based on neural networks and Deep learning, so they are much more complex algorithms than a matrix factorization, and it is more difficult to change their values.

Lastly, we would like to comment on the relevance of the study of the several challenges that emerge from the use of recommender systems. These systems play an important role in defining which information is accessed by the population, and that gives their proper functioning a special significance. Issues such as biased recommendations, filter bubbles, recommendation of fake news, or of information that does not benefit the real interests of users, have social implications whose magnitude are probably not yet understood. This should encourage intense investigation on the matter in future years, to help use this technology in the best way possible.

7 References

(1) Burke, R.; O'Mahony, M. P.; and Hurley, N. J., "Robust collaborative recommendation. In Recommender Systems Handbook", 2015.
(2) M. P. O'Mahony, N. J. Hurley, and G. C. Silvestre, "Recommender systems: Attack types and strategies," in AAAI, pp. 334–339, 2005.
(3) I. Gunes, C. Kaleli, A. Bilge, "Shilling attacks against recommender systems: a comprehensive survey", Artif Intell Rev 42, 767–799, 2014.
(4) H. Huang, J. Mu, N. Zhenqiang Gong, Q. Li, B. Liu, M. Xu, "Data Poisoning Attacks to Deep Learning Based Recommender Systems", 2021.
(5) X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in Proceedings of the 26th international confer- ence on world wide web. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.
(6) R. Salakhutdinov, A. Mnih, G. Hinton, "Restricted Boltzmann Machines for Collaborative Filtering". ICML 2007.