LEARNING-BASED ATTACK AND DEFENSE ON RECOMMENDER SYSTEMS

by

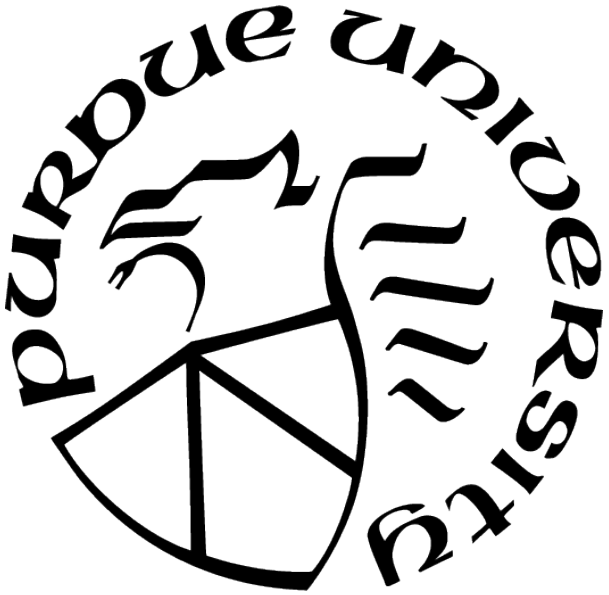
Agnideven Palanisamy Sundar

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science



Department of Computer and Information Science

Indianapolis, Indiana

August 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Xukai Zou, Chair

Department of Computer and Information Science

Dr. Feng Li

Department of Computer and Information Technology

Dr. Qin Hu

Department of Computer and Information Science

Approved by:

Dr. Shiaofen Fang

To my parents and my sister.

ACKNOWLEDGMENTS

I would like to express my sincere thanks to Dr. Feng Li for his encouragement and direction, which has undeniably impacted every step of this work. Thank you for the skills and knowledge you have imparted to me.

I owe my deepest gratitude to Dr. Xukai Zuo for his continuous guidance, encouragement, and mentorship. I am gratefully indebted to him for his patience and support.

I would also like to thank Dr. Qin Hu for agreeing to serve on my committee and for her guidance in completing this work.

Last but not least, I would like to thank my mother Vasanthi Sundar, my father Sundar Palanisamy, and my sister Koushicaa Sundar for their unparalleled love and support; they have always believed in my abilities.

TABLE OF CONTENTS

LIST OF TABLES 9

LIST OF FIGURES 10

ABSTRACT 11

1 INTRODUCTION 12

1.1 How to deceive a customer by attacking the rating and review systems of a website? 13

1.1.1 Attacking the review system of a website: Fake Review Injection . . 13

1.1.2 Attacking the rating system of a website: Shilling Attack 13

1.2 Contributions 14

1.2.1 Fake Review Injection Scenario 14

1.2.2 Shilling Attack Scenario 14

1.3 Thesis Outline 15

2 LITERATURE REVIEW 16

2.1 Fake Review Injection 16

2.1.1 Traditional Machine Learning Models. 16

2.1.2 Neural Network Models. 16

2.1.3 Graph-based models. 17

2.2 Shilling Attacks 17

2.3 Multi-Armed Bandits 18

3 A DYNAMIC DEFENSE AGAINST SPAM REVIEWERS ON RECOMMENDER SYSTEMS- A GRAPH EMBEDDING-BASED DEEP LEARNING APPROACH 19

3.1 Introduction 19

3.2 Problem Definition 20

3.3 Framework 21

3.3.1 Dynamic User Proximity Graph Generation 23

3.3.2 Spammer Identification . . . 23
Individual-Content Similarity . . . 24
Review Rating Count . . . 24
Edge Weight and ICS Threshold . . . 25
3.3.3 Embedding and Clustering . . . 26
Graph Embedding . . . 26
Clustering Fake Reviewers . . . 27
3.4 Experiments . . . 27
3.4.1 Datasets . . . 27
3.4.2 Ground Truth . . . 28
3.4.3 Evaluation Metrics and Baselines . . . 29
Unsupervised Clustering . . . 29
Baseline Comparison . . . 30
3.4.4 Analysis on System Parameters . . . 30
Choice of Epsilon . . . 30
Impact of Window Size . . . 31
Impact of ICS threshold . . . 32
3.5 Conclusion . . . 33

4 AN OVERVIEW ON SHILLING ATTACKS AND THEIR DEFENSES . . . 34
4.1 Introduction . . . 34
4.2 Understanding Collaborative Filtering . . . 35
4.2.1 Collaborative Filtering . . . 36
User-User-based CF . . . 36
Item-Item-based CF . . . 37
4.3 Shilling Attacks . . . 37
4.3.1 Attack Profile . . . 37
Attack Size . . . 39
Filler Length . . . 39
4.3.2 Attacks Models . . . 39

Standard Attacks . . . 41
Obfuscated Attacks . . . 44
4.4 Detection Attributes . . . 47
4.4.1 Generic Attributes . . . 47
4.4.2 Model Specific Attributes . . . 48
4.5 Detection Algorithms . . . 49
4.5.1 Supervised Approaches . . . 50
4.5.2 Unsupervised Approach . . . 54
4.5.3 Defense against Shilling Attacks . . . 56

5 A PRACTICAL SHILLING ATTACK ON RECOMMENDER SYSTEMS- A MULTI-ARMED-BANDIT-BASED REINFORCEMENT LEARNING APPROACH 58
5.1 Introduction . . . 58
5.2 Motivation . . . 59
5.2.1 Developments in Collaborative Filtering . . . 59
5.2.2 Detection Traits . . . 60
User-based Traits . . . 60
Item-based Traits . . . 61
5.2.3 Factors Influencing the Effectiveness of a Shilling Attack . . . 62
Attack Size . . . 62
Choice of Rated Items (Selected Items) . . . 62
5.3 Methodology . . . 62
5.3.1 First, instead of treating each profile individually, grouping them into categories, makes it easier to reduce the uncertainty as a group: Categorizing Injected Profiles. . . . 63
Observer Profile . . . 63
Attacker Profile . . . 64
5.3.2 Second, to ensure high efficiency, the uncertainty reduction process needs to happen simultaneously with the item selection proces: Multi-Armed-Bandit-based Uncertainty Reduction. . . . 65

5.3.3 Rating Scheme . . . 67
5.4 MAB-based Shilling Attack Scheme . . . 67
5.4.1 Setup Phase . . . 67
5.4.2 Iterative Phase . . . 68
5.4.3 Termination Phase . . . 69
5.4.4 Toy Example of MAB Shilling Attack . . . 71
5.5 Experiments . . . 72
5.5.1 Dataset . . . 73
5.5.2 Our Approach . . . 73
5.5.3 Baseline Attack Algorithms . . . 74
Bandwagon Attack . . . 74
Average Over Popular Attack . . . 74
Random Attack . . . 75
Average Attack . . . 75
5.5.4 Baseline CF Algorithms . . . 75
CoClustering Algorithm . . . 75
kNN Basic . . . 75
kNN with Means . . . 76
Non-negative Matrix Factorization (NMF) . . . 76
5.5.5 Evaluation Metrics . . . 76
5.5.6 Result Analysis . . . 76
Changes with filler size . . . 77
Changes with CF Algorithms . . . 77
5.6 Conclusion . . . 78
6 SUMMARY . . . 79
REFERENCES . . . 80
PUBLICATIONS . . . 89

LIST OF TABLES

3.1 Datasets 28

3.2 Unsupervised Method Results (@ϵ=0.8)[with ψ] 29

3.3 Comparison with baseline approaches 30

4.1 Attack Models 42

4.2 Symbol definitions 48

4.3 Supervised Classification based Detection Algorithms 51

4.4 Unsupervised Clustering based Detection Algorithms 53

LIST OF FIGURES

3.1 User Proximity Graph-based User Node Embedding 22

3.2 Users Clustered in different ϵ values 31

3.3 Impact of Window Sizes 32

3.4 Impact of ICS threshold 33

4.1 An illustration of user-user and item-item collaborative filtering methods. 36

4.2 An Illustration of a Shilling Attack 38

4.3 Attack Profile 38

4.4 Number of users with the target item in their top-N recommendations after shilling attack 40

4.5 Types of Shilling Attack 46

5.1 Characteristic traits of an attack which are exploited during the detection process 60

5.2 A toy example of possible observer profile combos. 64

5.3 Using MAB for observer combo selection. 68

5.4 Attacker profiles at the end of the attack. 70

5.5 Toy example of a shilling attack with 200 injected profiles. 72

5.6 Probability density of the beta distribution for four observer combos at different steps. 73

5.7 Percentage of users with target item in their top-20 recommendation list when filler size is 3%. 77

5.8 Percentage of users with target item in their top-20 recommendation list when filler size is 5%. 77

ABSTRACT

The internet is the home for massive volumes of valuable data constantly being created, making it difficult for users to find information relevant to them. In recent times, online users have been relying on the recommendations made by websites to narrow down the options. Online reviews have also become an increasingly important factor in the final choice of a customer. Unfortunately, attackers have found ways to manipulate both reviews and recommendations to mislead users. A Recommendation System is a special type of information filtering system adapted by online vendors to provide suggestions to their customers based on their requirements. Collaborative filtering is one of the most widely used recommendation systems; unfortunately, it is prone to shilling/profile injection attacks. Such attacks alter the recommendation process to promote or demote a particular product. On the other hand, many spammers write deceptive reviews to change the credibility of a product/service. This work aims to address these issues by treating the review manipulation and shilling attack scenarios independently. For the shilling attacks, we build an efficient Reinforcement Learning-based shilling attack method. This method reduces the uncertainty associated with the item selection process and finds the most optimal items to enhance attack reach while treating the recommender system as a black box. Such practical online attacks open new avenues for research in building more robust recommender systems. When it comes to review manipulations, we introduce a method to use a deep structure embedding approach that preserves highly nonlinear structural information and the dynamic aspects of user reviews to identify and cluster the spam users. It is worth mentioning that, in the experiment with real datasets, our method captures about 92% of all spam reviewers using an unsupervised learning approach.

1. INTRODUCTION

There is an abundance of data, resources, and knowledge available on the internet in the current day and age, making it impossible for users to reach the most relevant information to suit their needs. Many websites use recommender systems to make the searching process easier for users. There are different recommender systems; we will be focusing on the explicit rating-based systems, which use the ratings provided by some users to make recommendations to the other users. The users also directly decide the quality of an item based on ratings; the general rule of thumb is that high ratings are considered good. On top of using ratings, users also use the reviews provided by other users to estimate product quality.

Let us now discuss the influence of ratings and reviews on a customer. For our investigations, let's consider the case of e-commerce websites, which provide both ratings and reviews to support a customer's decision. For example, consider a customer trying to buy a flashlight from an e-commerce website; once the customer enters the search term, "Flashlight", an array of items is displayed. The order in which the items are displayed depends on the website. Now, the customer uses the rating and the reviews for each item to decide which one to purchase. When it comes to the rating, the selection is based on two factors: 1) The number of ratings for an item, 2) The actual rating value. Usually, the customer prefers the item with the maximum number of ratings and the highest ratings. Once they select a product based on rating, then for further scrutiny, the customers check the reviews written for the item. The reviews associated with the highest rating and the lowest rating are valued more by the customer. Usually, such reviews help new customers understand the experiences of other customers who have already used the product. The more elaborate the review, the more valuable it is considered.

The other way in which the rating of an item impacts a customer is through recommendations. Even if the customer does not explicitly search for an item, it can still be recommended to a customer based on their past purchases. Only the top-rated items relevant to a customer end up being recommended to the customer. Now that we have seen how ratings and reviews influence a customer's decision let us look at ways in which an adversary can alter the ratings, reviews, and, subsequently, the recommendations made to a customer.

1.1 How to deceive a customer by attacking the rating and review systems of a website?

Now, let's consider the same situation from the perspective of an attacker. The attacker can have one of the three motives:

1) To promote an item favorable to the attacker,

2) To demote a rival item, and

3) To disrupt the trustworthiness of the website.

The lack of access to the back end forces the attackers to alter the rating and reviews written for items to accomplish their goals.

1.1.1 Attacking the review system of a website: Fake Review Injection

Spammers post unauthentic reviews for their benefit, manipulating users to buy products of poor quality. In 2008, Liu et al. published a paper analyzing the influence of opinion spam on online reviews [1]. Since then, multiple types of research have been conducted to find ways to attack and protect the review system. When only a few fake reviews are injected, its impact on the system will be minimal. If many fake reviews are written, it alters the credibility of the product. About one-third of the total reviews on Amazon and one-fifth of all reviews on Yelp are claimed to be fake reviews [2].

When spammers work together as a group or when the same spammer maintains multiple user accounts, its corresponding impact on a business is much more significant. Such spammer groups exhibit certain characteristic traits that differentiate them from authentic users. Some paymasters on crowdsourcing platforms like RapidWorkers and ShortTask, purchase fake reviews for their products on sites like Amazon and Yelp.

1.1.2 Attacking the rating system of a website: Shilling Attack

An item can be promoted by increasing the overall rating of the item. To do so, the attacker injects many fake profiles into the website. These fake profiles are used to give a high rating to the target item, which needs to be promoted. These fake profiles can also be used to provide low ratings to rival items to reduce their overall rating. If the ratings assigned

by these fake profiles are carefully curated, then the recommender system will promote the target item to many authentic users. Such an attack on the recommender system is known as the shilling attack.

A shilling attack is a special type of attack that injects multiple fake profiles into the recommender system. Shilling attacks can be classified based on intent as a push or nuke attack, where a product is either promoted or demoted, respectively, to gain an economic advantage over competitors. Over the years, multiple attack profiles and models have been developed [3]–[9]. Simultaneously, many detection techniques and algorithms have emerged to counter such attacks [10]–[15]. Almost all of the attack models use the same attack profile while generating malicious users. The attack models' differences are attributed to how the individual elements of the attack profiles are formed. The simplicity of existing attack models makes them easy to detect, but such simplistic attack models are not used in practice.

1.2 Contributions

The motive of this thesis is to act as an impetus for furthering the research in the practical implication of attacks against Recommender Systems by conducting the following works.

1.2.1 Fake Review Injection Scenario

Though the act of writing a fake review is trivial, recent developments in the domain have made this type of attack truly notorious. Attackers have come up with sophisticated yet practical procedures to increase the severity of the attack. In this work, we build a Dynamic Deep Learning-based clustering of spam reviewers using graph-embedding methods. We target high-impact fake reviewers who work in groups.

1.2.2 Shilling Attack Scenario

Unlike Fake Review Injection, the practical applicability of existing Shilling Attack works is minimal. This impracticality makes it easier to detect such attacks because of their prominent traits. Our work develops a sophisticated Reinforcement Learning-based attack process that improves attack reach while maintaining high similarity with authentic users.

1.3 Thesis Outline

This thesis contributes to the recommender system attack scenario in the following ways:
1) The next chapter, Chapter 2, discusses the State-of-the-art research works on the topics of Fake Review Injection and Shilling Attacks.
2) We will discuss our Deep learning-based clustering method to detect and remove spam reviews from the recommendation system in Chapter 3 .
3) We focus on understanding the existing types of shilling attacks, their defenses, and the common traits that make them detectable in Chapter 4.
4) We build a practical Reinforcement Learning-based shilling attack that can avoid being detected by the most common detection techniques in Chapter 5.
5) Chapter 6 summarizes this Thesis work.

2. LITERATURE REVIEW

In this chapter, we will be looking at various research works related to *Fake Reviews Injection* in Recommender Systems, *Shilling Attacks* in Recommender Systems, and their defenses. We also focus on how the Deep Learning and Reinforcement Learning approaches used in our work are utilized in other researches.

2.1 Fake Review Injection

In 2008, Liu et al. published a paper analyzing the influence of opinion spam on online reviews [1]. Since then spam detection has been a topic of rigorous study and research.

2.1.1 Traditional Machine Learning Models.

Earlier researchers exploited the text and behavioral features for detecting individual fake reviewers. [16] utilized the rating behavior of users to detect spam. [17] used an approach in which the burstiness in the review is exploited. [18] models the spamicity of authors. Rating deviation, review count, and the ratio of first reviews were some of the attributes used by most earlier researchers [18]. These works mainly focused on detecting individual fake reviewers, whereas our work focuses on detecting fake reviewers exhibiting group behavior.

Most of the initial works in group spam also relied on features like time window, group content similarity, and group deviation as attributes [19], [20]. These methods did not adopt an embedding approach to learn the features. Our work focuses on finding the nonlinear relationships between group members.

2.1.2 Neural Network Models.

Some of the more recent works in the deceptive opinion spam detection have been adopting the neural network-based approaches [21], [22]. [23] employs neural network on a document-level representation learning method. [24] relies on a data-driven approach instead

of depending on expert knowledge. [25] proposed a method based on the differences in the experience of the reviewers. These techniques are more reliant on the text-based features for its detection process while our approach takes both the linguistic-based and meta-data based features.

2.1.3 Graph-based models.

[26] approached the spam detection problem as a graph problem; the review classification was based on trustiness, honesty, and reliability scores. [27] only depends on the topology of the graph for fraud detection. [28] is a joint deep embedding technique that aims at grouping users and items in the same low dimensional latent space. [29] adopts a neighborhood-based approach to classify the spam reviewers. [30] applies logistic regression on the review data after learning the reviewer and product node embeddings. These approaches ignore the dynamic aspect of the reviewer - product relationship. Our approach efficiently integrates the linguistic features into the graph model while simultaneously utilizing the dynamic aspects associated with reviewer behavior.

2.2 Shilling Attacks

Shilling attack techniques came into existence from the early 2000s. The initial attack models focused more on disrupting the RS's performance rather than pushing or nuking a target product. In [31], Random attack and Average attack models were used to check the effectiveness of the attacks on User-User and Item-Item based CF. A more sophisticated model with better results in promoting the product was used in [32]. This attack, known as the bandwagon attack, chose the popular items to be the selected items. Reverse-bandwagon and love/hate attacks were two effective nuke attacks in [33]. Attacking only a segment of the RS instead of the entire items list was executed in [34]. Some recommenders gave possible ratings for its items, which was used in probe attack, [35]. As soon as shilling attacks were discovered to be possible, simultaneously, there were many research works to detect such attacks. A series of obfuscated attack strategies were created to avoid detection. In [36], user

shifting and target shifting techniques were used to hide some of the attack profiles and target items. Bhaumik et al. in [37] used a combination of the random, average, bandwagon, and segmented attacks to avoid detection. Both [38] and [39] designed the attack profiles to be similar to the most popular users and most popular items, respectively. Compared to other methods, our method works online and considers other possible features present in the RS.

2.3 Multi-Armed Bandits

Multi-Armed bandits have recently been used in many applications. We discuss some of those applications here. In the healthcare segment, [40] employs an adaptive model and allocate more samples to provide better treatment options. In the finance segment, [41] use MAB for making online portfolio choices. In [42], the authors make an algorithm to choose between earning an immediate profit and learning for future profit when the demand information is incomplete. In RS, [43] apply MAB on large-scale RS even when no prior information about the user is available. In [44], the authors use MAB to maximize the influence of a product by selecting the optimal seed profiles for promotion. The authors of [45] utilize MAB for selecting the proper response for dialogue+ in online learning systems. In [46], MAB is used for anomaly detection by interacting with human subjects to learn ground truth. In the telecommunication segment, [47] use MAB for the best wireless network selection by multiRadio Access Technology. In our work, we use MAB to reduce the uncertainty related to optimal item selection.

3. A DYNAMIC DEFENSE AGAINST SPAM REVIEWERS ON RECOMMENDER SYSTEMS- A GRAPH EMBEDDING-BASED DEEP LEARNING APPROACH

A version of this work and some preliminary experiment data of this chapter has been published in IEEE GLOBECOM 2020 [48].

3.1 Introduction

In recent times, the number of online product/service providers have increased rapidly. User feedback plays a crucial role in influencing their buyers. This factor has led to a widespread increase in fraudulent behavior: fake reviews and ratings to sway customers. Spammers post unauthentic reviews for their benefits, manipulating users to buy products of low quality. About one-third of the total Amazon reviews and one-fifth of all Yelp reviews are claimed to be fake reviews [2].

When spammers work together as a group or when the same spammer maintains multiple user accounts, its corresponding impact on a business is much more significant. Some paymasters on crowdsourcing platforms like RapidWorkers and ShortTask, purchase fake reviews for their products on sites like Amazon and Yelp. These crowdsourced reviews end up exhibiting group behavior even without the spammer's knowledge. Such groups exhibit group behavioral traits that are valuable in spammer detection. Most of the existing detection methods rely only on review text and reviewer linguistic behavior as attributes, which can be easily modified by spammers to go undetected ([1],[16]).

The features that cannot easily be changed by a spammer are the 'time of post' and the 'graph topology' associated with the review graph. The attributed bipartite review graph consists of the users as the source nodes and the products as the destination nodes. The review timestamp, rating, and content are the attributes on the edges. But the existing graph-based spammer detection approaches do not utilize the nonlinear relationship between different users or the dynamic aspect of user behavior ([26], [27]).

Paymasters post requests on crowdsourcing platforms for a specific number of spam reviews. Most spammers tend to respond to these requests as early as they can, leading to a series of spam reviews posted within a short time frame. Likewise, spammer groups also post reviews within a particular time frame to increase their attack efficiency. This dynamic aspect needs to be captured to differentiate between the fake reviewer groups that work on the same product.

At the same time, some of the linguistic features like self-similarity can be used to differentiate between the spammers and authentic users. The majority of authentic user misclassifications can be avoided by efficient incorporation of such linguistic characteristic traits into the graph model.

To address these issues, we propose a scheme to detect spam reviewers through deep network embedding. The network embedding learns the underlying representation of user nodes while preserving the local and global spam reviewer network structure. This work aims to represent all the users in the same low dimensional latent space, such that the spam users get clustered closer together while the regular users get evenly distributed.
In summary, the contributions of this chapter are the following:
• We propose a novel scheme to detect fake reviewers through deep dynamic structure learning on an extrapolated bipartite graph using unsupervised learning.
• We propose an unsupervised clustering approach that can capture about 92% of all the fake reviewers in the experimental dataset by utilizing a neighborhood-based graph.
• By exploiting the similarity between different reviews posted by the same reviewer, we adequately and correctly classify a large number of authentic users.

3.2 Problem Definition

We first define how to construct the homogeneous user proximity graph from the heterogeneous bipartite user-product graph, consisting of users linked to the products they have reviewed.

Given a bipartite multigraph, $G = (U, V, E)$, where U is the set all user nodes, V is the set of all product nodes, and E is the set of all edges representing the reviews. $E = \{e_{i,j}\}$ is

the review written by user u_{i} for product v_{j}. If a user leaves multiple reviews on the same product, then multiple edges are present between the user and the product.

DEFINITION 1. *(Dynamic Network) A series of dynamic network snapshots is a set of bipartite graphs* $(G^1, ..., G^T)$*, where* $G^t = (U, V, E^t)$*,* $1 \leq t \leq T$ *represents how the edges are connected from U to V at time t. A dynamic sliding window generates another series of graphs* $(G^{\triangle_1}, G^{\triangle_2}, ...)$*, where* $G^{\triangle_x} = G^x \cup G^{x+1} \cup ... \cup G^{x+n}$*,* $x \in t$ *is combination of the graph snapshots within the given window size n.*

In a user-product graph, each network snapshot signifies the reviews posted in one day, and the dynamic sliding window represents multiple consecutive days depending on the window size n. For example, if $n = 4$, then $G^{\triangle_1}$ would be from Monday to Thursday, $G^{\triangle_2}$ is Tuesday to Friday, $G^{\triangle_3}$ is Wednesday to Saturday, and so on.

DEFINITION 2. *(User Proximity Graph) Given a graph G = (U,V,E), such that two source vertexes* u_x *and* u_y *are connected to the same destination vertex* v_{j} *through edges* $\mathrm{e}_{x\mathrm{j}}$ *and* $\mathrm{e}_{y\mathrm{j}}$ *respectively. Then a new graph* $G = (U, E)$ *with only the user-nodes is formed by introducing an edge* e_{xy} *between the two source nodes* u_x *and* u_y*.*

If a product v_k has a review from only one user or does not have any reviews at all within the given time frame, then the product v_k does not contribute to any edges in the final graph G.

3.3 Framework

For our discussions, we are also going to treat spammers from crowdsourcing websites as spammer groups. This consideration holds because the paymasters in crowdsourcing websites act analogous to the leaders of the spam groups. They assign review tasks to the spammers (group members) in crowdsourcing sites, and the spammers end up posting reviews as early as they can to earn rewards. Most of the spammers from crowdsourcing sites work on multiple tasks to maximize their earnings, which further asserts our consideration.

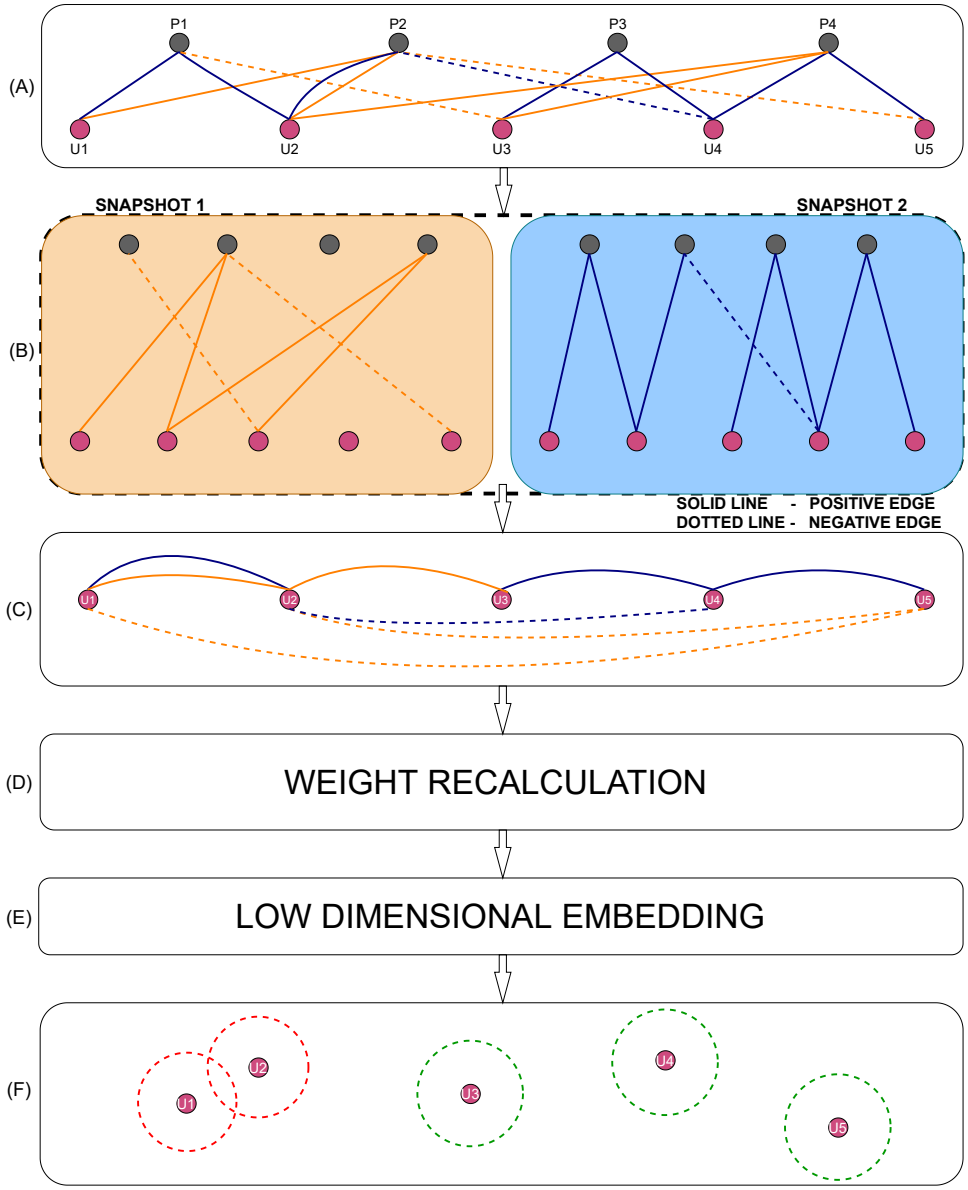


Figure 3.1. User Proximity Graph-based User Node Embedding

The two major tasks addressed in our method are clustering spam users based on review similarity and avoiding authentic user misclassification based on linguistic behavior. Our proposed scheme works as follows:

3.3.1 Dynamic User Proximity Graph Generation

In general, a review network is expressed as a bipartite graph, where the source nodes are the users, and the destination nodes are the products for which they wrote the reviews. From this graph, we form links between users to establish their review similarity. We consider the review time window as the metric for this purpose. Members of a review group tend to work together on a target product in a short time interval [19]; this feature is captured by dividing the graph into dynamic network snapshots. Edges between two users are formed if and only if both the users have reviewed the same product in the same time window.

Fig.3.1 depicts the steps involved in learning the embeddings from the initial graph. For ease of understanding, each snapshot in the figure represents one time window. The different colored edges indicate the different time windows in which those reviews were written. In (A), which is the bipartite user-product graph, if a user gives a positive review to a product, it is depicted by a solid line, and a dotted line depicts a negative review. (B) is the connections between products and users in different time windows, and (C) represents the links present in the user-user graph formed from (B). In (C), when two users give a similar rating to the same product, they are connected by a positive edge. But if the ratings are dissimilar, then they are connected by a negative edge. (D) and (E) are the weight calculation and the embedding process to find the latent representation of the users, and (F) is a toy 2D representation of the users in the latent space.

3.3.2 Spammer Identification

Now, with the links between users established, the objective is to alter edge weights such that the spammer groups are closely positioned in the latent space. We modify the weight of the edges between the nodes based on two spam indicators: Review Rating count(RRC) and Individual-Content Similarity(ICS). ICS is a well studied linguistic spam indicator considered

to be a dominant trait in a spammer behavior. Unlike in previous works ([26], [49]), we use ICS for edge weight calculation.

Individual-Content Similarity

Often, a spammer ends up slightly modifying one of their previous reviews and re-posts it on the same or different product, owing to the lack of firsthand experience. We call this feature the individual content similarity. Typically, the ICS of a spammer is much higher than the authentic user. We calculate the ICS of a user:

$$ICS(u_{\mathrm{i}}) = \frac{\sum_0^k desc(cosine(r_{\mathrm{i}_x}, r_{\mathrm{i}_y}))}{k}, x, y \in r_{\mathrm{i}} \tag{3.1}$$

Where r_{i_x} and r_{i_y} are two different reviews written by the same user. ICS of a user is calculated by taking the average of the top k cosine similarities of each pair of reviews written by the user. The value of k is dependent on the number of reviews posted by the user. The reason to be using the top k values instead of all the values is to ensure that there exists a prominent gap between the ICS of the spam users and authentic users. By selecting only the top k values, most of the non-fake reviews posted by the spammer are not taken into consideration for ICS calculation, thereby avoiding a decrease in their ICS value. This also ensures the efficiency of ICS threshold. The definition and integration of the ICS threshold into the weight calculation is explained next.

Review Rating Count

The number of occurrences in which two users have reviewed the same product is indispensable in revealing if they work in connivance. If the frequency of such occurrences is low, then there is a possibility of it being a coincidence.

Likewise, if the rating behavior of two users is not similar to each other, they are not part of the same review group and should be placed farther away in the latent space. Both these factors are expressed by the Review Rating Count. RRC is calculated as:

$$RRC(u_{\mathrm{i}}, u_{\mathrm{j}}) = \begin{cases} s(r_{\mathrm{i}}, r_{\mathrm{j}}) - d(r_{\mathrm{i}}, r_{\mathrm{j}}) & \text{if } \frac{d(r_{\mathrm{i}}, r_{\mathrm{j}})}{s(r_{\mathrm{i}}, r_{\mathrm{j}})} \leq \gamma \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

Where $s(r_{\mathrm{i}}, r_{\mathrm{j}})$ and $d(r_{\mathrm{i}}, r_{\mathrm{j}})$ are the total number of positive edges (similar ratings) and negative edges (dissimilar ratings) between users u_{i} and u_{j} respectively. γ is a threshold value beyond which the RRC becomes zero. In other words, a high number of dissimilar ratings, as opposed to similar ratings, indicate that the two users are not correlated. If multiple edges exist between two users, they are replaced by a single edge after RRC calculation.

Edge Weight and ICS Threshold

Here, we discuss a way to avoid authentic user misclassification. The weight of the edge between users u_{i} and u_{j} is a combination of RRC and ICS values. The directed edge weight between user u_{i} and user u_{j} is given by:

$$w(\mathrm{e}_{\mathrm{i,j}}) = \begin{cases} RRC(u_{\mathrm{i}}, u_{\mathrm{j}}) + \alpha(ICS(u_{\mathrm{i}})) & \text{if } ICS(u_{\mathrm{i}}) \geq \psi \\ \eta & \text{otherwise} \end{cases} \tag{3.3}$$

α is the factor used to scale the min-max value of ICS to match the value of RRC. ψ is the ICS threshold value, and η is a small constant.

This edge weight calculation leads to most of the spammers having high edge weights, clustering them closer together in the latent space. But unfortunately, some of the authentic users who review many products may also get clustered, leading to their misclassification. Using ICS threshold is a way to utilize the linguistic features to avoid such misclassifications. ICS of authentic users is usually low. If a user has an ICS lower than the threshold value,

the user is potentially authentic. Then the weights of the user's edges are equated to the small constant η, chosen such that it is the smallest value in the set of all edge weights.

It is important to note that this is a directed edge from u_{i} to u_{j}. The edge weight from u_{j} to u_{i} is calculated using the same technique.

3.3.3 Embedding and Clustering

Graph Embedding

The number of edges in the User Proximity network is much higher than the bipartite graph. Embedding places each of the users in a low dimensional latent space. The distance between each user has a direct correlation to the weight calculated between the users. Most of the network embedding techniques are not efficient for large-scale networks, and also don't work with directed or weighted edges ([50],[51]).

The embedding technique used in our method is Large-scale Information Network Embedding (LINE) [52], which is highly scalable. LINE takes into account both the first and the second-order proximity of the network. For the sake of being self-sufficient, a brief description of the Loss functions used in the LINE method is presented here.

The first-order proximity preserves the direct relationship between user nodes. In LINE, the first-order proximity is preserved by minimizing the objective function:

$$O_1 = -\sum_{\mathrm{i,j}\in E} w(\mathrm{e_{i,j}}) \log p_1(u_{\mathrm{i}}, u_{\mathrm{j}}) \tag{3.4}$$

Where $p_1(u_{\mathrm{i}}, u_{\mathrm{j}})$ is the joint probability between the user nodes u_{i} and u_{j}. The first order proximity is only applicable for undirected graphs and hence the first half of the edge weight equation needs to be modified as follows to fit the model.

$$\begin{aligned} w(\mathrm{e_{i,j}}) =& RCC(u_{\mathrm{i}}, u_{\mathrm{j}}) + \alpha(MCS(u_{\mathrm{i}}, u_{\mathrm{j}})) \\ &+ \alpha(ICS(u_{\mathrm{i}})) + \alpha(ICS(u_{\mathrm{j}}))] \end{aligned} \tag{3.5}$$

The second-order proximity preserves the neighborhood similarity between user nodes. In second-order proximity, each vertex has two roles: context for other vertexes and the vertex

itself. The vertexes with identical distribution over the contexts are embedded closer together. The second-order proximity is preserved by minimizing the objective function:

$$O_2 = -\sum_{\mathrm{i,j}\in E} w(\mathrm{e_{i,j}}) \log p_2(u_\mathrm{i}|u_\mathrm{j}) \tag{3.6}$$

Where $p_2(u_\mathrm{i}|u_\mathrm{j})$ is the conditional distribution of each vertex over the contexts. The directed edge approach, eqn.3.3, is used for edge weight calculation while preserving the second-order proximity. The proximity calculations and optimization methods were adopted from LINE [52].

Clustering Fake Reviewers

After obtaining the low dimensional vector representation of all the user nodes, the spammers exhibiting group behavior need to be clustered together. If two users have a relatively high edge weight between them, they would exist closer to each other in the latent space. In our approach, spammers tend to have such high edge weights and can be clustered based on the distance between them. The fake review group detection is carried out using DBSCAN [53], a density-based clustering method. The reason for selecting this algorithm over other methods is its ability to recognize the number of fake reviewer clusters without having to input the block count explicitly. Algorithm 1 explains the process in detail.

3.4 Experiments

3.4.1 Datasets

The dataset used for the experimental evaluations consists of more than half a million unique reviews from users on Amazon. 300 of the products in the dataset are considered to be target products as they have requested fake reviews from the crowdsourcing platform RapidWorkers. This dataset was first collected and used by Kaghazgaran et al. for their work in [29] and [49]. Table 3.1 shows the details about the dataset.

Algorithm 1: Fake Review Group Clustering

Input : Latent representation of weighted User - Proximity Graph G, distance ϵ and minPoints

Output : Clusters of Spammers

DBSCAN:

Initialize: Arbitrary starting data point;

1: Determine neighborhood of the point based on distance ϵ;

2: If number of points in neighborhood is higher than minPoints, all points in neighborhood becomes part of the cluster. Otherwise point is marked as noise point.;

3: Identify n such core points that have more number of neighbors than minPoints ;

4: The non-core points are either assigned to nearby clusters if they are ϵ distance from the cluster or marked as noise;

5: The points which are part of the cluster are fake reviewers and the noise is authentic user.

return *Clusters of Fake Review Groups*

Table 3.1. Datasets

Total Reviews	580,000
Total Reviewers	12,212
Target Products	300
Reviews on Target Products	21,162
Number of Spam Reviewers	3,116

3.4.2 Ground Truth

This dataset consists of 12,212 reviewers, which is a mix of both authentic users and spammers. To differentiate between authentic and spam users, we consider the case where a user has reviewed two or more of the 300 target products between 2016-17. Given that the dataset contains more than 135,000 products, the odds of an authentic user reviewing two target products is low. We can safely assume that the users who have reviewed at least two target products are either members of the crowdsourcing sites or spam reviewers from other sources, labeling about 3,116 users as spam users.

3.4.3 Evaluation Metrics and Baselines

The evaluation of the Unsupervised learning method is going to be based on Precision, Recall, and F1-Scores. As the baseline algorithms, we are using Wang et al. [26] and Fraud Eagle [54] methods. The reason to be choosing these methods is that these are also graph-based approaches. Wang et al. define variables to quantify the quality of reviewers, reviews, and products based on 'trustiness of reviewers', 'honesty of reviews', and 'reliability of stores'. Fraud Eagle framework treats the spam detection problem as a network classification problem by employing a propagation-based algorithm. Our approach captures the nonlinear structural information as well as the dynamic features. Either of which is not utilized in either of the baseline algorithms.

Unsupervised Clustering

In the unsupervised clustering process, DBSCAN [53] is applied to the output of the network embedding. The two hyperparameters which influence the number of clusters that are formed are the min-sample, which is set to 2, and the ϵ value. The choice of ϵ determines the algorithm's efficiency. For lower values of ϵ, the clusters that are formed consist of users that have a high probability of being a spammer, but not all of the spammers are detected. As we increase the value of epsilon, the number of spammers getting clustered increases. Table 3.2 gives the precision, recall, and F1 score of the unsupervised clustering method at ϵ=0.8. In this experiment, we chose an ICS threshold such that only 40% of the users have ICS greater than the threshold ψ.

Table 3.2. Unsupervised Method Results (@ϵ=0.8)[with ψ]

Window Size	15 Days	30 Days	45 Days	60 Days
Precision	0.968	0.976	0.974	0.971
Recall	0.867	0.913	0.887	0.878
F1-macro	0.927	0.949	0.940	0.929

As we can observe from the table, the precision, recall, and F1 score reaches a peak at a window size of 30 days. This means that most of the fake review groups and crowdsourced manipulators work within this time frame.

Baseline Comparison

The users in both [26] and [54] are given an anomalous score, which specifies the extent to which a user deviates from authentic behavior. The drawback of these approaches is that the performance of these methods is highly dependent on the quality of the data available. If all the products in the dataset had multiple reviews, then there will be an increase in the performance efficiency of these methods. But the dataset used is sparse in this sense, that is, not all the products in the dataset have multiple reviews. Table 3.3 shows its comparison against our approach.

Table 3.3. Comparison with baseline approaches

Approach	Precision	Recall	F1-macro
Wang et al.[26]	0.58	0.73	0.54
Fraud Eagle[54]	0.61	0.78	0.67
This work	0.97	0.91	0.94

3.4.4 Analysis on System Parameters

Choice of Epsilon

Epsilon value is the minimum distance between nodes for them to be clustered together. The choice of epsilon determines how many users get clustered. To select the best epsilon value for the dataset, we define a method to set the upper and lower bounds. Fig. 3.2 shows how there is an exponential increase in the number of users that are being clustered after the value $\epsilon = 1.10$. Beyond this ϵ value, the users who are evenly distributed because of the ICS threshold also get clustered. To avoid this, we set the upper bound for ϵ value to be 0.1 lesser than the this $\epsilon(=1.10)$ value . In the dataset used, the upper bound was set to be

ϵ=1.0. We set the ϵ to a value where 10% of the total users get clustered as lower bound, safely assuming that at least 10% of all the users are fake users, based on the knowledge about the dataset and prior literature [2]. In this study, we work with epsilon values between 0.65 and 0.8, which gives the maximum efficiency.

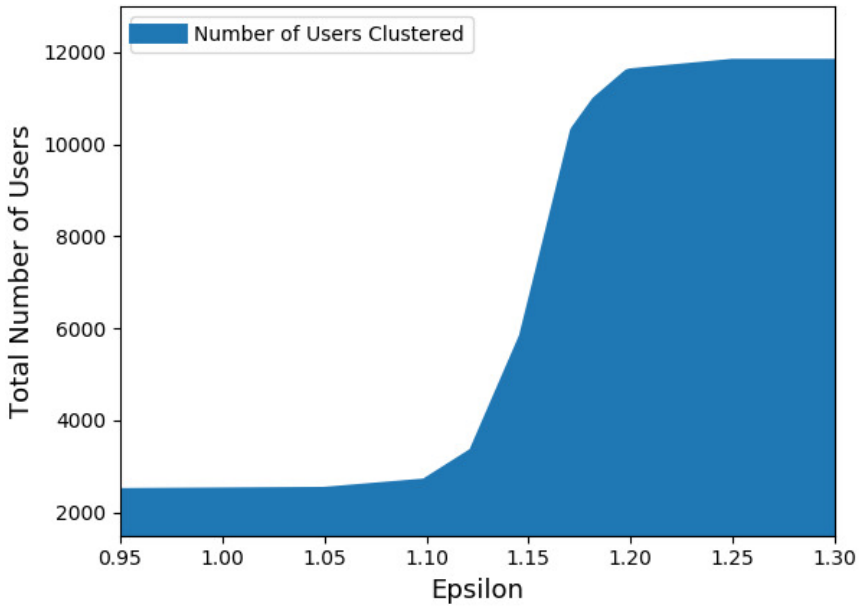


Figure 3.2. Users Clustered in different ϵ values

Impact of Window Size

We have considered four window sizes for experimentation. Most of the group activity or crowdsourced attack occurs within a time interval of one to two months. We test the classification performance of our approach in 15-day, 30-day, 45-day, and 60-day windows. As can be observed from the table. 3.2, and the fig. 3.3, the performance is at its best in the window size of 30 days. If we further increase the window size, there is a drop in the number of users that are being clustered. It could be possible that the ratings which are within a 30-day window have the maximum influence in the overall rating of the product, forcing the members of a group or crowdsourced workers to post a review within that time frame.

Fig.3.3 is a graph depicting the number of users that are being clustered for the different window sizes.

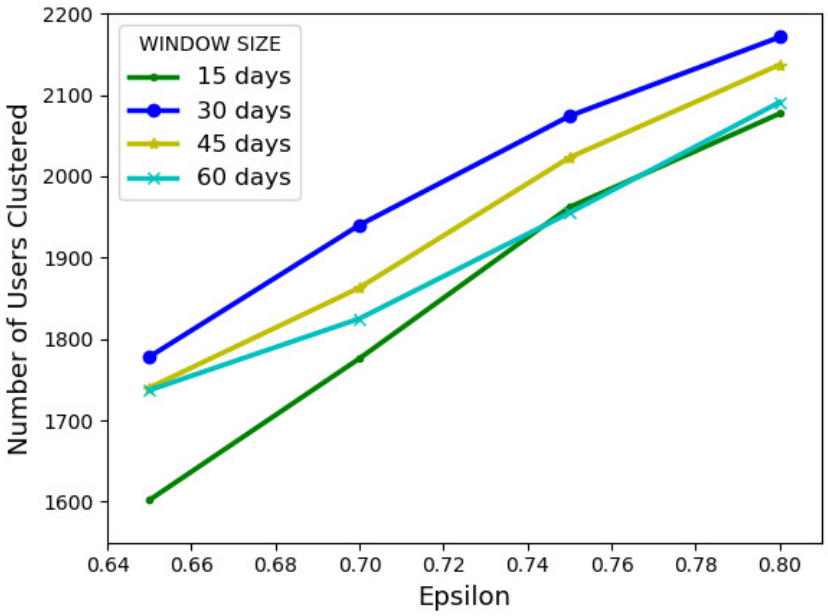


Figure 3.3. Impact of Window Sizes

Impact of ICS threshold

With the ICS threshold, many of the authentic users get their edge weight reduced to a small constant, evenly distributing them in the latent space. Such a distribution causes the authentic users to be farther away from all other users, leading to them becoming outliers during the clustering process. Fig. 3.4 shows the jump in precision and recall values with ICS threshold. In this experiment, we chose an ICS threshold such that only 40% of the users have ICS greater than the threshold ψ. If we arrange the ICS values of all the users in ascending order into a set, the ICS value which splits the set in a 60:40 ratio is taken as the ψ value. This is feasible because of the relatively small number of spammers.

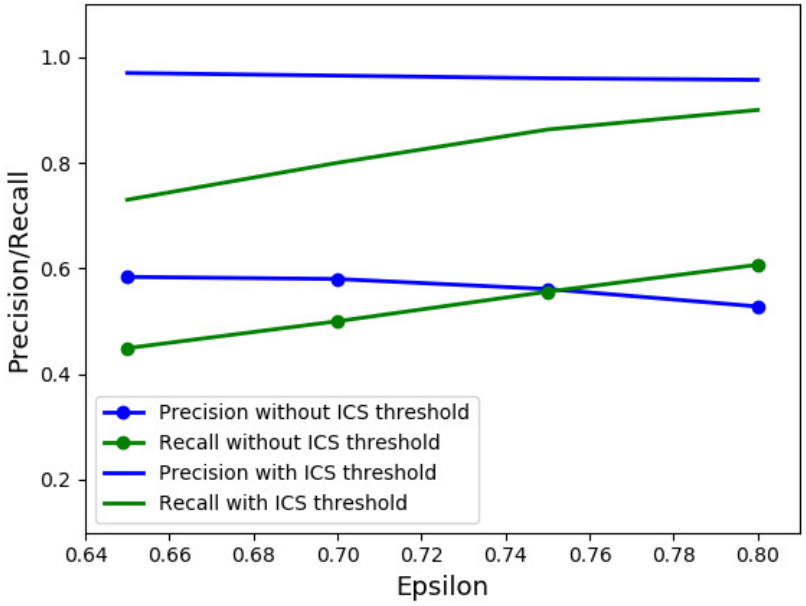


Figure 3.4. Impact of ICS threshold

3.5 Conclusion

In this chapter, we have explored how group behavioral traits of fake reviewers can be exploited to cluster spammers together in a low dimensional latent space, that works both for fake review groups and crowdsourced review manipulators. The framework takes into consideration the dynamic aspect of the user-product spatial-temporal relationships and converts it to a user proximity graph, which includes: (i) Recalculated weights to incorporate the rating behavior of the user based on dynamic attributes, (ii) utilization of linguistic characteristics of a reviewer to identify connivance among users; and (iii) application of a threshold to prevent the majority of the authentic reviewers from being misclassified. Our results are encouraging, showing the impact of window size in clustering the fake reviewers, further asserting our intuition to include the dynamic aspect of a reviewer's behavior. In the future, we are planning to expand our scheme to work across multiple domains.

4. AN OVERVIEW ON SHILLING ATTACKS AND THEIR DEFENSES

A version of this chapter has been published as a survey in IEEE Access 2020 [55].

4.1 Introduction

We live in the information age where there is an overload of information generated by individuals, companies, and governments. The internet has become a common platform for all of this information to be shared and stored. Multiple e-commerce platforms have come into existence, selling all kinds of products and services. With this information overload, it has become increasingly difficult for online users to find content relevant to them. As a means of addressing this problem, many websites are utilizing the recommender system [56]. The recommender system is an information filtering mechanism to provide customers with products/services based on their requirements.

Multiple Recommender System approaches are employed to cater to different kinds of needs in different websites. Over the years, there has been a drastic growth in the methods used to improve recommendation results for different purposes [57]–[63]. Recommendation systems can be broadly classified into two types, content-based [64]–[69] and collaborative filtering-based [70]–[75].

Content-based filtering recommends products to users by comparing the content of the products to the users' profiles. The downside of using content-based filtering is the over-specialization; they tend to recommend only the products that are very similar to what has already been consumed by the user which wasn't the case with collaborative filtering. The collaborative filtering recommender system works by analyzing the past behavior of a user. The key idea is that users with similar behavior have similar needs and interests. Recommendations made using collaborative filtering depend on relationships between the users and items. Unfortunately, due to its openness and dependency on user ratings, collaborative filtering is prone to shilling attack, also known as a profile-injection attack.

Shilling attack [3], [76]–[79] is a particular type of attack where a malicious user profile is inserted into an existing collaborative filtering dataset to alter the outcome of the recommender system. The injected profiles explicitly rate items in such a way that the target item is either promoted or demoted. It has been a topic of study for over a decade, and multiple survey papers have covered different parts of this domain. In [80], Mehta et al. focus exclusively on robust collaborative filtering techniques and not on detection techniques or attack strategies. In [81], the types of attacks and the detection techniques discussed are limited. In 2014, [82] produced one of the most comprehensive surveys on the topic, but it presents details on the attacks only until 2011. The survey in [77] focuses only on the statistical measures used in the detection and the basic shilling attack methods. Kaur et al. [83] perform experimental evaluation comparing the most commonly used shilling attack methods. In [84] and [76], the discussions do not consider the different detection attributes used in supervised and unsupervised detection methods. Both [78] and [79] briefly discuss the various attack and detection methods. There is no discussion on robust algorithms, and the detection methods are not categorized.

This chapter aims to be a comprehensive survey of different attack models and detection attributes for shilling attacks on collaborative filtering recommender systems. Since shilling attacks are more prominent in explicit rating systems, this chapter's scope is limited to methods that work on explicit rating systems where the user explicitly gives one rating for each item. Shilling attacks are possible in both nearest-neighbor-based and matrix factorization-based recommender systems; it is predominantly tested in nearest-neighbor settings, which will be used in our explanations.

4.2 Understanding Collaborative Filtering

The function of a recommender system is to suggest items that may be of interest to a website's users. This suggestion is based on the other items that the users have rated or purchased on the website. The recommender system intends to make these recommendations to let the users explore items that may have been otherwise missed. For instance, users may be recommended with movies they have never heard of, based on the other movies they have

rated. At the same time, a recommender system also suggests items that a user might need, reducing the effort needed to find it. For example, batteries are recommended to users with a flashlight in their online shopping cart, making the user experience more pleasant and easier.

4.2.1 Collaborative Filtering

It is the most commonly used system in practice. It can be broadly classified as User-User-based and Item-Item-based.

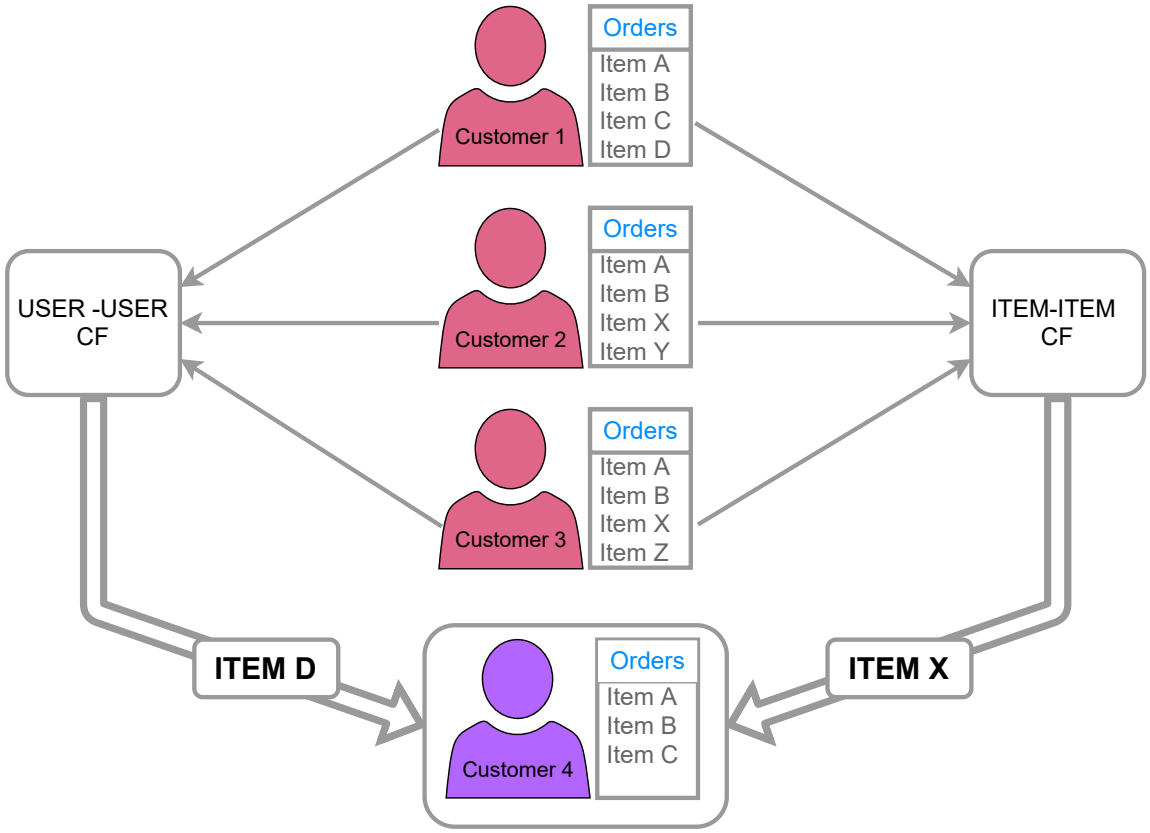


Figure 4.1. An illustration of user-user and item-item collaborative filtering methods.

User-User-based CF

[85] This CF works by finding users who have purchased/rated a similar set of items and recommends the items purchased by one user to the other. To illustrate, if users u_1 and

u_2 purchased items i_1, i_2 and i_3, then these two users are considered to be similar to each other. When user u_1 purchases another item i_4, then this item will be recommended to user u_2 based on user-user CF.

Item-Item-based CF

[86] This type of CF forms relationships between items based on how often those items are purchased together. Consider that the item sets $\{i_1, i_2, i_3\}$, $\{i_1, i_2, i_4\}$, and $\{i_1, i_2, i_5\}$ are purchased by users u_1, u_2, and u_3 respectively. When another user u_4 purchases item i_1, then item i_2 will be recommended to the user. If two items are found together frequently in the purchase history of multiple users, then those items will be strongly related in the item-item based CF.

Collaborative filtering can be interpreted as a way to extract relationships and similarities based on how users interact with items in an online platform. Fig. 4.1 illustrates the difference in the outcome of the two types of CF.

4.3 Shilling Attacks

Shilling attacks can be classified based on intent as a push or nuke attack, where a product is either promoted or demoted, respectively, to gain an economic advantage over competitors. Fig.4.2 gives an example of the impact of a shilling attack on a recommender system. Here, item X is the target item that is promoted by the shilling attack. Over the years, multiple attack profiles and models have been developed [3]–[9]. Simultaneously, many detection techniques and algorithms have emerged to counter such attacks [10]–[15]. Almost all of the attack models use the same attack profile while generating malicious users. The attack models' differences are attributed to how the individual elements of the attack profiles are formed.

4.3.1 Attack Profile

The attack profile is segmented into four sets: Selected items I_s, Filler items I_f, Null items $I_\emptyset$, and the Target item(s) I_t. I_t is the set of items or an individual item which needs

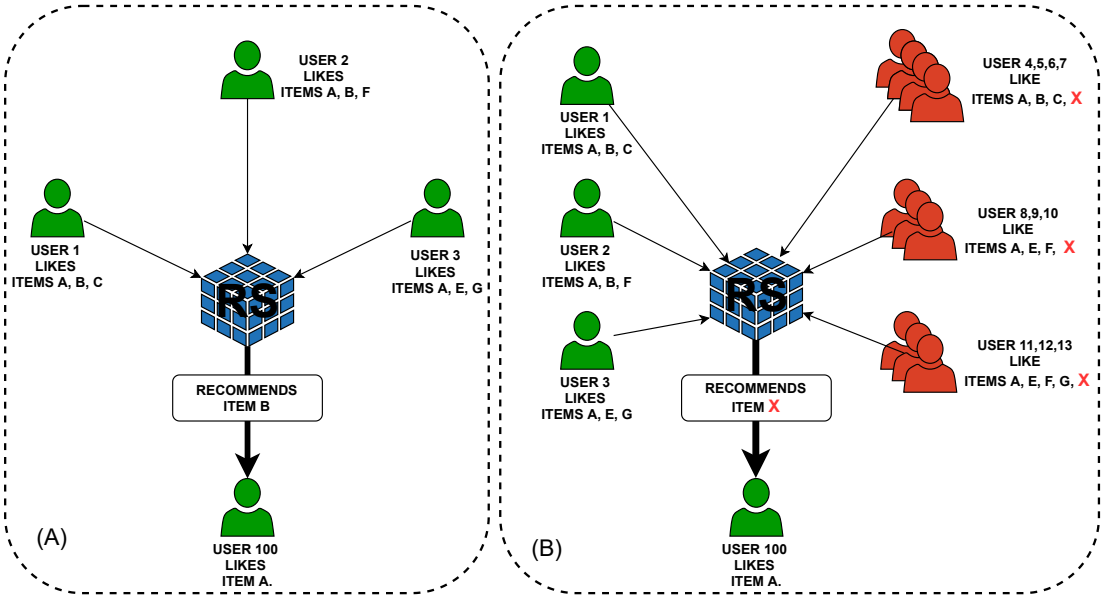


Figure 4.2. An Illustration of a Shilling Attack

to be pushed or nuked. I_s is the set of items carefully chosen so that a malicious profile has a similarity with the maximum possible number of genuine users. The efficiency of an attack is decided by how many users are recommended with the target item. I_s plays a crucial role in attack efficiency. I_f is the set of filler items chosen and rated in such a way that the malicious profiles can camouflage with the genuine profiles. $I_\emptyset$ is the set of items that are not rated by the malicious user [87]. Fig.4.3 illustrates an attack profile.

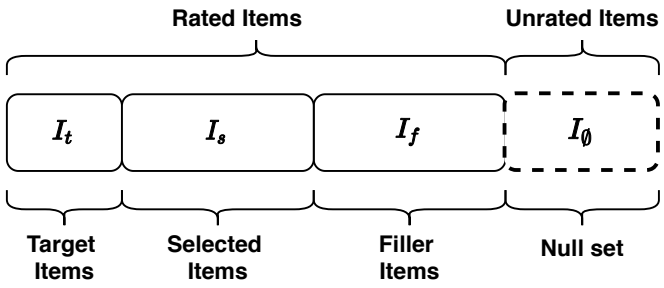


Figure 4.3. Attack Profile

Attack Size

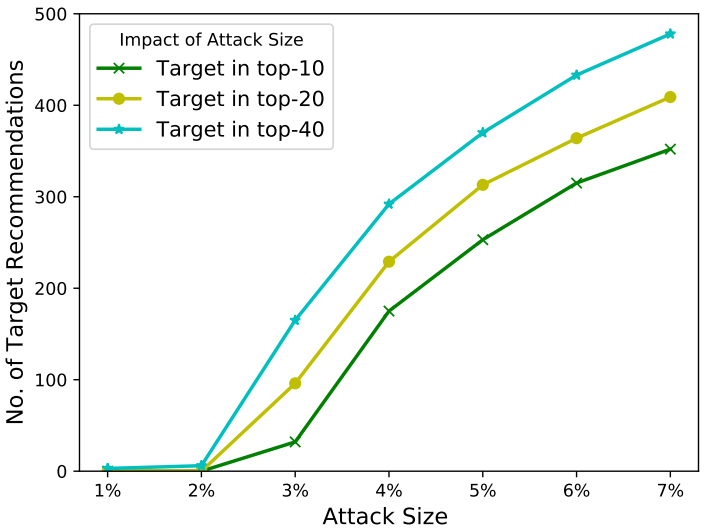
The number of injected profiles and the number of items rated per profile considerably influences an attack's reach. The number of injected profiles, also known as attack size, should be large enough to have any impact on the system. Fig.4.4a shows the increase in the reach of the target item with respect to the attack size. The MovieLens dataset [88] with 100,000 movie ratings from 943 users on 1682 items were used for the generation of this graph. A random attack discussed in the next section, with various attack sizes (1% to 7% of the number of authentic users), was implemented. A movie with an average rating of 1.9 calculated from 31 authentic ratings was chosen as the target item. Before the attack, the target item was not part of the top-40 recommendations made to any of the authentic users using a kNN-based algorithm. Fig. 4.4a shows the number of users who have the target item in their top-10,20 and 40 recommendations after the attack. The graph shows that the target item reaches more people as the attack size increases. The number of filler items per attack profile was fixed at 2% of the total number of items.

Filler Length

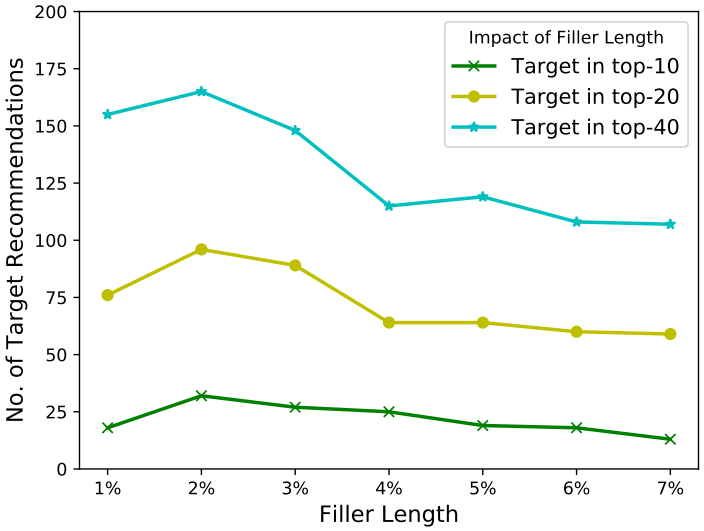
The number of filler items rated per injected profile is known as the filler size. Fig.4.4b shows the impact of increasing the number of rated items, also known as filler length, on the target recommendation. For the evaluation of this graph, the number of injected profiles was fixed at 3% of total users. From this graph, it can be seen that increasing the filler length can be detrimental to attack efficiency, implying that high filler length can cause the attack profiles to be less similar to authentic users.

4.3.2 Attacks Models

Based on the attackers' motivation and knowledge, multiple attack models have been developed over the years. All these attacks can be categorized either as a high-knowledge attack or a low-knowledge attack. Low-knowledge attacks are more practical and have a higher chance of having a real-world impact, but the efficiency of such attacks is also low. On



(a) Reach of the target item in various attack sizes



(b) Influence of filler length for a fixed attack size

Figure 4.4. Number of users with the target item in their top-N recommendations after shilling attack

the other hand, high-knowledge attacks can have a massive effect on Recommender Systems' performance, but they are harder to pull off. From a practical standpoint, an inside job is a viable option to execute a high-knowledge attack, the chances of which are negligible. So, in real-world applications, a moderately efficient low-knowledge attack poses a more significant threat than a highly efficient high-knowledge attack. Based on how the selected items and filler items are chosen, multiple attack models exist which can further be classified as standard or obfuscated, based on the attacks' ability to go undetected.

Standard Attacks

These are the attack models that do not make an exclusive attempt to go undetected in a recommender system. Many detection algorithms have a higher chance of detecting the shilling attack profiles injected using these attacks.

Random Attack [3], also known as the RandomBot attack, is the simplest form of shilling attack. In this model, the items rated by the attack profile are chosen at random except for the target item. The ratings for these items is around the system overall mean. The target item gets the maximum or minimum rating based on whether it is a push or a nuke attack. Some attacks are intended to disrupt the trustworthiness of a recommender system, known as random vandalism [84]. Being the most straightforward attack, it is also the least effective. The purpose of a random attack is usually more effective in disrupting the performance of a Recommender System rather than promoting the target item. The ease of execution of random attacks is because of its low-knowledge requirement. All that the attacker needs are the overall system mean which can be easily empirically calculated. Being the simplest attack, it is not very effective.

Average Attack [3] is similar to the random attack in terms of the item selection process. The randomly chosen items are rated based on the rating distribution of the individual items. Each filler item is assigned the mean rating of that item. This attack is feasible only if the attacker has immense knowledge about the dataset on which the recommender system is built. The effectiveness of this model is proportional to the attacker's knowledge. Though

Table 4.1. Attack Models

	I_s		I_f		I_t	
Attack Models	**Selection**	**Rating**	**Selection**	**Rating**	**Rating**	**Knowledge**
Random		$\emptyset$	Random	System mean	max/min	Low
Average		$\emptyset$	Random	Item mean	max/min	High
Bandwagon	Popular	max/min	Random	System mean	max/min	Low
Reverse Bandwagon	Unpopular	min	Random	System mean	-/min	Low
Segment	Segmented	max/min	Random	System mean	max/min	Moderate
Love/Hate		$\emptyset$	Random	max	-/min	Low
User Shifting		$\emptyset$	Random	Sys mean+random	max/min	Low
Target Shifting		$\emptyset$	Random	System mean	max-1/min+1	Low
AoP		$\emptyset$	Top X% of popular	Item mean	max/min	Low
PIA	Power items	Item mean		$\emptyset$	max/min	High
PUA	Power users' items & ratings			$\emptyset$	max/min	High

the only difference between random attack and average attack is the filler ratings, the average attack's effectiveness is much better.

Bandwagon Attack [4], [6] is the type of attack where the profiles generated by attackers are filled with popular items with high ratings. The attack profiles are naturally closer to a large number of users. The target item is given the highest rating. This attack can be further divided into bandwagon-random and bandwagon-average depending on the rating scheme used for the filler items. Bandwagon also falls under the low-knowledge attack category since the attacker only needs publicly available data.

Probe Attack [89] is not an attack that can be generalized for all systems. Some recommender systems project a predicted rating score for each of the items. The attacker uses this detail to rate the items, enabling it to be similar to other users. The attacker gives genuine ratings to some seed items. Then, when the recommender suggests more items, the attacker forms the rated items list based on these items. This scheme ensures that the attack profiles stay close to its neighbors. It also enables the attacker to learn more about the system.

Segmented Attack [90] targets a specific group of users who are likely to purchase the target item in an e-commerce setup. Segment attacks are usually deployed in item-based collaborative filtering. The rated items and the ratings are based on the attacker's knowledge about the segment. The significant advantage that this method has over other methods is its ability to reach potential customers. For example, if the target item is a book in the science fiction genre, then the selected items will also be from the same genre. Such selection increases the chances of the target book reaching more fans of science fiction. Since the attack is deployed only in a segment of the system, the impact is high.

Love/Hate Attack [5] is a highly effective nuke attack. Here, the attacker randomly chooses filler items and gives them the highest ratings and the least rating to the target item. Despite the simplicity of this model, the effectiveness is surprisingly high. Though it was predominantly designed for nuke attacks, it can also be used for a push attack by altering the ratings. Push attack is not as effective as a nuke attack. Table 4.1. comprehensively summarizes the differences in various attack models.

Reverse Bandwagon Attack [5], [6] is the exact reversal of a bandwagon attack. This attack is used to nuke the target product by giving low ratings to the items with high negative reviews and giving the least rating to the target item. It is also a low-knowledge attack, just like the bandwagon attack. Though it is highly similar to the bandwagon attack, the efficiency of the reverse bandwagon attack is slightly better.

Obfuscated Attacks

To go undetected from detection algorithms, attackers try to obfuscate their attack signature. Many models incorporate slight modifications to the standard attack techniques to achieve obfuscation. Fig. 4.5 shows which of the standard attacks have influenced which of the obfuscated ones. The dotted lines indicate a direct influence between the attacks. The ones that are not derived from specific standard attacks can be incorporated with any standard attack. Though obfuscation might slightly reduce the impact of the attack, it is better than being detected.

Average Over Popular [91] is a technique used to obfuscate the Average Attacks. Here, the filler items are chosen from the top X% of the most popular items with equal probability. This method is much more effective than randomly choosing from the entire collection of items. The choice of X influences the detectability of the attack.

Mixed Attack [92] is done by using the random, average, bandwagon, and segmented attacks in equal proportions, simultaneously. The detection technique should have the ability to detect all of the standard attacks to be successful. The different attack methods are used to push/nuke the same target item. It helps in evading multiple detection techniques.

Noise Injection [93] adds to a Gaussian distributed random number multiplied by a constant to each rating, for a subset of injected profiles. The degree of obfuscation is dependent on the constant that is multiplied. It can be effectively applied to all of the standard attack methods to obfuscate its signature. Since the rating scheme is affected by noise injection, a slight but observable drop in the attack efficiency can be noticed.

User Shifting [93] is an obfuscation tactic where a subset of the rated item of each injected profile is modified. The ratings of this subset of items are either increased or decreased

to reduce the similarity between attack profiles. For different groups of the injected profiles, different subsets of rated items have their ratings modified.

Target Shifting [93] shifts the rating of the target item to one level lesser than the highest possible in push attacks. In nuke attacks, the target rating is shifted to one rating higher than the least possible rating. This strategy is specifically useful in evading the detection methods that penalizes users that give an extreme rating to items. If the target item is already popular, it will be harder to push while employing target shifting obfuscation. In such cases, some other obfuscation methods should be used.

SAShA [7] is an attack strategy that uses the semantic features extracted from a knowledge graph to improve the performance of standard CF attack models. A knowledge graph is a structured repository of factual, categorical, and ontological information [94]. This attack works by computing the semantic similarity between the knowledge graph derived features of the target item and all other items in the system. This information is leveraged to generate the most efficient set of filler items.

Power Item Attack [9], [95] utilizes the power items which are chosen based on three methods. Power items are defined as the set of items that can influence the largest group of items. These items effectively alter the recommendations made for other users. In PIA-AS, the top-N items with the highest aggregate similarity are chosen to be the power items. Such similarity is possible only when a considerable number of users have rated the same two items. In PIA-ID, the In-Degree centrality is the criteria for choosing the power items. The similarity of each pair of items is calculated using weighted significance and the top-N of each item is selected. PIA-NR chooses the items with the highest number of users as the power items.

Power User Attack [9], [95], similar to PIA, chooses the set of users who have the maximum influence on the broadest group of users. In PUA-AS, the top X users with the highest Aggregate Similarity are chosen as the power users. In PUA-ID, the users who participate in the highest number of neighborhoods are selected as power users, based on the In-Degree centrality concept. The power users in PUA-NR are the users with the highest number of ratings in their profile.

In [8], Chen et al. describe a method to use both rated item correlation and item popularity to generate malicious users with strong attack ability and similarity to real users. In their approach, each malicious user profile is generated individually. The rated items of a profile are selected based on a matrix of real user profiles.

As soon as the vulnerability of Collaborative Filtering to shilling attacks was discovered, various detection techniques were also constructed. We can broadly classify these techniques into supervised and unsupervised detection techniques. In literature, there is an array of detection attributes that govern these methods.

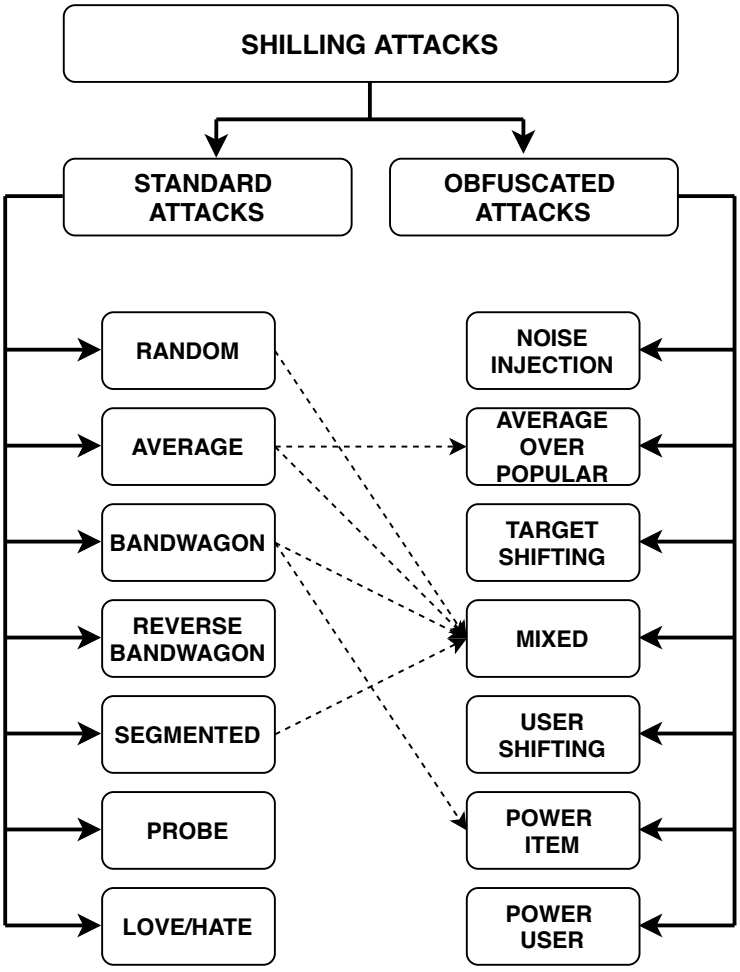


Figure 4.5. Types of Shilling Attack

4.4 Detection Attributes

The attributes that differentiate the shilling profiles from the authentic profiles are considered as the detection attributes. The detection attributes that are designed to work irrespective of the type of attack model are known as Generic attributes.

4.4.1 Generic Attributes

The attributes that are not tailored for specific attack models fall under this category. The efficiency of these attributes alters with the different attack models used. Table. 4.2 gives the definitions for the symbols used in the explanations below.

Length Variance (LengthVar) measures the difference in the length of a user's profile from the average length of a profile. Here, length denotes the number of items rated by a given user profile. Some attack profiles tend to have too many rated items, deviating substantially from an average user's length [11].

$$LengthVar = \frac{|N_u - \overline{n}|}{\sum_{k \in U} (n_k - \overline{n})^2} \tag{4.1}$$

Rating Deviation from Mean Agreement (RDMA) is the measure of rating deviation of a user on a set of target items with respect to other users, combined with the inverse rating frequency of these items [10].

$$RDMA = \frac{\sum_{\mathrm{i}=0}^{N_u} \frac{|r_{u,\mathrm{i}} - \overline{r_{\mathrm{i}}}|}{NR_{\mathrm{i}}}}{N_u} \tag{4.2}$$

Weighted Deviation from Mean Agreement (WDMA) is firmly based on the RDMA attribute. The significant difference of this attribute is that it places high weight for rating deviations for sparse items. WDMA was experimentally found out to give higher information gain [11].

$$WDMA = \frac{\sum_{\mathrm{i}=0}^{N_u} \frac{|r_{u,\mathrm{i}} - \overline{r_{\mathrm{i}}}|}{NR_{\mathrm{i}}^2}}{N_u} \tag{4.3}$$

Table 4.2. Symbol definitions

Symbols	Definitions
N_u	Number of ratings from user u
NR_{i}	Number of ratings for item i
$r_{u,\mathrm{i}}$	User u's rating for item i
$\overline{r_{\mathrm{i}}}$	Mean rating for item i
U	Total number of users
$\overline{u}$	Average length of a profile
P_u	Profile of user U
$P_{u,T}$	Target items in the profile
$P_{u,F}$	Filler items in the profile
I_u	Set of items rated by user u
U_u	Partition of the profiles of user u
$\lvert U_u \rvert$	Number of profiles of user u

Weighted Degree of Agreement (WDA) captures the cumulative differences of a user's rating of an item from the item's average rating, divided by the number of ratings for the item. WDA is empirically the same as the numerator of the RDMA [11].

$$WDA = \sum_{\mathrm{i}=0}^{N_u} \frac{|r_{u,\mathrm{i}} - \overline{r_{\mathrm{i}}}|}{NR_{\mathrm{i}}^2} \tag{4.4}$$

4.4.2 Model Specific Attributes

The problem with using only the generic attributes is that sometimes it is unable to distinguish malicious profiles from the authentic users, especially when the authentic user exhibit unusual behavior. Attack specific attributes were constructed to overcome these shortcomings. These detection attributes discover the partitions in user profiles so that their behaviors exhibit similarity to one particular attack model.

Mean Variance (MeanVar) is used to detect average attacks. It partitions the attack profiles into three parts: the items with extreme ratings (target items), all other rated items in profiles (filler items), and unrated items. This attribute works by computing the

mean-variance between all the filler items and the overall average. A low variance would indicate the possibility of an average attack [11].

$$MeanVar = \frac{\sum_{\mathrm{j} \in P_{u,F}} \left(r_{u,\mathrm{j}} \overline{r_u}\right)^2}{|P_{u,F}|} \tag{4.5}$$

Filler Mean Target Difference Model (FMTD) targets the segmented attack model. This attribute relies on the difference between ratings of the items in target partition and the items in filler partition [11].

$$FMTD = \left| \frac{\sum_{\mathrm{i} \in P_{u,T}} r_{u,\mathrm{i}}}{|P_{u,T}|} - \frac{\sum_{k \in P_{u,F}} r_{u,k}}{|P_{u,F}|} \right| \tag{4.6}$$

Filler Average Correlation (FAC) focuses on detecting the random attack model. When a random attack is executed, then the ratings given to the items are chosen at random. This attribute calculates the correlation between the ratings in the profile and the average ratings of the items. The correlation is expected to be low for random attacks [12].

$$FAC = \frac{\sum_{\mathrm{i} \in I_u} (r_{u,\mathrm{i}} - \overline{r_{\mathrm{i}}})}{\sqrt{\sum_{\mathrm{i} \in I_u} \left(r_{u,\mathrm{i}} - \overline{r_{\mathrm{i}}}\right)^2}} \tag{4.7}$$

Filler Mean Difference (FMD) utilizes the fact that the filler items have a mean rating similar to the overall system average in the random attack model. If the mean ratings are similar, then the user profile could potentially be a random attack profile [12].

$$FMD = \frac{1}{U_u} \sum_{\mathrm{i}=1}^{|U|} |r_{u,\mathrm{i}} - \overline{r_{\mathrm{i}}}| \tag{4.8}$$

4.5 Detection Algorithms

The detection algorithms can be broadly classified into two: Supervised detection methods and Unsupervised detection methods. The supervised techniques require the data to be labeled during the training process, whereas the unsupervised approaches do not. The availability of labeled ground truth is minimal in the recommender system datasets. This

downside has led to unsupervised approaches being adopted more than supervised in recent times.

4.5.1 Supervised Approaches

The shilling attack problem was treated as a classification problem by Chirita *et al.* [10], used the RDMA and DegSim as the feature metrics for detecting malicious profiles. The method was developed to detect random and bandwagon attacks. Later on, two more generic metrics, namely WDMA and WDA, were added by Burke *et al.* [11] to improve the classifier's performance. SVM, kNN, and C4.5 were the most commonly used classifiers for the detection of fake injected profiles. The problem with using the generic attributes was that many authentic users who had extreme behaviors were misclassified as shilling profiles. To overcome this problem, as well as to improve the accuracy of the classifications, attack specific attributes were formulated by [11],[12]. Different attack specific attributes were formed for average, random, segment, and bandwagon attacks.

Williams *et al.* [96] utilized three strategies to increase the accuracy of detection in the supervised approaches: similarity to reverse-engineered attacks, target concentration, and rating anomaly detection. This detection technique is effective because of the added robustness to the system, but it is highly reliant on the classifier's choice. Their study shows that combining various attributes improves the classifier's performance, especially the support vector machine, and significantly reduces the impact of the most potent attack models. The attributes used in their method are RDMA, WDMA, DegSim, LengthVar, MeanVar, FMD, FAC, and FMTD.

The use of meta-learning was introduced by [97] to improve the precision of the detection. This algorithm can be considered a two-step process where the base-level training is done on attack profiles and available ratings. The second step is to combine the base-level output with the meta-level input for final attack detection. This algorithm had higher precision than previous methods. The diversity of the classifiers reduces the correlation of misclassification, positively impacting the meta-level prediction. They tested their approach against single

Table 4.3. Supervised Classification based Detection Algorithms

Detection Algorithms	Assumptions	Targeted Trait	Effective Against	Weak Against	Downside
Chirita *et al.* [10]	Attack profiles have high similarity with its neighbors	Rating Behavior	Random, Segmented	Average, Obfuscated	Authentic user misclassification
Burke *et al.* [11]	Items with fewer ratings have higher importance	Rating Behavior	Random, Segmented	Average, Obfuscated	Authentic user misclassification
Mobasher *et al.* [12]	Attack type is known	Filler rating variance, Profile length	Bandwagon, Random	Obfuscated	Being attack specific misses other attacks
Williams *et al.* [96]	Attack profiles have rating anomaly	Rating Behavior, Profile Similarity	Random, Bandwagon, Segmented	Obfuscated	Highly reliant on choice of classifier
Zhang *et al.* [97]	Multi-level classifiers improve efficiency	Profile Similarity and length	Random, Bandwagon, Average	Hybrid, Obfuscated	Two level classification leads to longer training time
Zhou *et al.* [98]	There are equal number of authentic and attack profiles	Target Analysis, Rating Behavior	Random, Average, Bandwagon	Hybrid, Obfuscated	Attack and authentic profiles need to be balanced to get good results
Yang *et al.* [14]	Weighted observations simulate balanced dataset	Filler Rating Variance and Length	PIA, PUA, Bandwagon	Probe	The process needs to be executed repetitively

SVM and voting SVM and experimentally proved to be more effective. The attributes used in their method are WDMA, RDMA, WDA, LengthVar, DegSim, MeanVar, FMD, and FAC.

SVM-TIA [98] had supervised, unsupervised, and semi-supervised detection approaches. The pitfall with using the supervised approach was that it needs a balanced data; it means that there should be an equal number of authentic profiles and attack profiles. The accuracy of the supervised approach was lower than their unsupervised approach which involved clustering and statistical methods. It is a two-phase process where rough detecting results are obtained in the first phase by alleviating class imbalance. In the second phase, the potential attack profiles are analyzed to discover the target profiles. Model-specific attributes like FMTD, MeanVar, FAC, and FMD are used in this method.

As mentioned earlier, the imbalance in the data available skewed the outcome of the supervised learning classifiers. AdaBoost was incorporated in [14] to diminish the perturbation caused by the imbalance. The authors first ease the hard classification task by using well designed features for the user profiles. It was achieved by applying weights to the various observations to accentuate the poorly modeled samples. This process was done repetitively to strengthen the correction of misclassification. The attributes used are RDMA, WDMA, WDA, LengthVar, MeanVar, FMTD, and FAC. In addition, they also use attributes that detect filler size with unpopular items.

Hao *et al.* [99] employed an ensemble detection method on features extracted from ratings, item popularity, and user-user graph. The feature extraction is performed by using Stacked Denoising AutoEncoders and PCA. It automatically extracts user features with different corruption rates. It used a three-stage process involving data preprocessing, feature extraction, and detection using weak classifiers. The novelty of items- the degree of difference between various items- was also used as a feature.

Table. 4.3 explains the different traits used for detection in some of the algorithms. It also discusses the various assumptions based on which the algorithms are built.

Table 4.4. Unsupervised Clustering based Detection Algorithms

Detection Algorithms	Assumptions	Targeted Trait	Effective Against	Weak Against	Downside
Mehta *et al.* [100]	Attack profiles are highly correlated	Rating Behavior, Profile Similarity	Random, Average	Segmented, Obfuscated	Sparse matrix reduces reliability
Bryan *et al.* [101]	Attack profiles exhibit anomalous structure	Rating Behavior	Bandwagon, Segmented	AoP, PIA, PUA	Misclassification of authentic users is higher in obfuscated attacks
Bhaumik *et al.* [92]	Attack profiles are smaller in number and exhibit high similarity with each other	Attack Size, Profile Similarity	Average, Segmented	Obfuscated	Effectiveness relies on attack size
Chung *et al.* [102]	Lower rating values have minimal contribution to the system	Rating Behavior	Random, Obfuscated	Probe, Average	All the downsides of using Beta probability apply to this method
Bilge *et al.* [103]	Attack profiles are similar to each other	Profile Similarity	Average, Segmented, Bandwagon	Obfuscated	Dependent on profile similarity, making it weaker against obfuscation
Yang *et al.* [104]	Most attack profiles exhibit detection features	Profile Similarity, Rating Behavior	Random, Bandwagon	Obfuscated	If attack profiles evade initial clustering, adaptive structure learning ineffective
Zhang *et al.* [105]	Authentic users do not exhibit extreme behavior	Rating Behavior	Random, Bandwagon, Segmented	Obfuscated	Authentic users with extreme behavior get misclassified.

4.5.2 Unsupervised Approach

The initial unsupervised approach introduced by Mehta *et al.* [100] applied Principle Component Analysis to the profile detection problem. Four factors led to this problem being suitable for PCA: spam users are highly correlated, low deviation from mean rating value, a high similarity with a large number of users, and the assumption that spam users work together. All the user profiles in the recommender system were projected onto a hyperplane formed from the user-item matrix. The user profiles which were clustered closer to the origin of the hyperplane were the attack profiles. The sparsity of the user-item matrix makes it harder for these predictions to be reliable. RDMA and WDMA are also used as detection attributes.

Bryan *et al.* [101] formulated a generic attribute aiding in the detection of attack profiles in an unsupervised manner. Their approach treats the attack profiles detection problem as an anomalous structure detection problem. The metric used is a variation of the Hv-score metric which was initially used in gene data analysis to aid in locating biclusters. This algorithm, called the UnRAP, seems useful in detecting both standard and obfuscated attacks. Their approach has better chances of catching future novel attack strategies that may escape supervised methods.

Based on the assumption that attack profiles are lesser in number and exhibit high similarity, [92] applied an attribute-based k-means clustering technique. The users were divided into two clusters, and the smaller cluster was identified as attack profiles. This method showcased a higher accuracy and lesser misclassification of genuine users. Irrespective of the attack strategy used, this work claims to have fewer authentic user misclassifications than previous methods. The attributes used include RDMA, WDMA, WDA, and LengthVar, along with the Hv-score metric used in [101].

Chung *et al.* [102] applied the Beta distribution algorithm to detect attacks. This method detected as many attacks as possible without penalizing the authentic users. Most of the problems associated with this method were inherited from Beta probability distribution itself. The upsides of using this method are its low alarm rate and high detection rate. This method

claims to work with sparse data and an unbalanced attack-normal profile ratio. This approach exhibits high performance even with a small attack size and has a low false alarm rate.

Another clustering approach relying on the attack profile similarities was [103], which used k-means clustering to move the fake profiles to the leaf nodes of a binary tree. With the user-item matrix and an optimal number of neighbors N, it recursively uses k-means clustering to cluster the users into two distinct groups. The indexed-cluster centers and intra-cluster correlation of the binary tree are used for attack profile detection. This approach's success rate is particularly high in the average, segment, and bandwagon attack models.

Yang *et al.* [104] developed an algorithm that focused on analyzing target users and items. It was a two-phase method. First, a density-based clustering method is applied to the dataset based on some selection features to identify malicious users. DBSCAN is used to determine the suspected users based on user features. Second, it spots suspicious items based on adaptive structure learning on selected features and further uses it to capture the attackers. The second phase helps in further scrutinizing the users from the first phase.

Zhang *et al.* [105] built a clustering approach based on the hidden Markov model (HMM) and hierarchical clustering. The users' rating behaviors are modeled using HMM. Based on the users' preference sequence and modeled rating behavior, each user's suspicion degree is calculated. Then, a hierarchical clustering method is used to group these users based on their suspicion degree into genuine and attack user clusters. They also apply their method on sampled Amazon review dataset to show its effectiveness.

Zhang *et al.* [106] proposed a method to improve the PCA approach in shilling profile detection. PCA is initially used to separate the profiles into two classes, positive labels for the detected and negative labels for all other users. Then they use the detection features - RDMA, WDMA, WDA, and LenVar - as data complexity features to calculate the CCMeasure of the dataset. CCMeasure is the classification complexity, a quantitative estimate on how difficult it is to classify the dataset. If the measure is high, it indicates that a significant number of authentic users are mislabeled, and the labels are flipped to reduce the data complexity.

Table. 4.4 shows the assumptions, traits, and the downsides of using some of these algorithms.

Having discussed detection techniques, other privacy risks that come with attack detection methods are also studied. Luo et al. [107] discuss the impact of an insider attack on shilling attack detection for recommendation systems. They consider a possible scenario where an attacker poses as an examiner who is kept from individual rating profiles by secure computations. Their attack model can infer the target rating profile with little prior knowledge and the output of the secure computations. Such an insider attack would pose a serious threat to the privacy of users.

4.5.3 Defense against Shilling Attacks

Parallel to the works focusing on shilling attack detection, there is a line of research intended to create robust algorithms that are immune to shilling attacks. These algorithms do not have a mechanism to find and remove the shilling profiles but can reduce the attack's effectiveness. We briefly discuss some of the recent robust algorithms in this subsection.

Yang *et al.* [108] combined the soft co-clustering algorithm with the user propensity similarity method to enhance the robustness of the recommender system and detect shilling attacks. It uses Bayesian co-clustering, a soft co-clustering algorithm that allows mixed membership of row and column, highly suitable for real data. This model combines RDMA with soft co-clustering to reduce the influence of shilling attacks. All the attack profiles are clustered into the same cluster, limiting the shilling influence amongst the attack profiles.

Turk *et al.* [109] developed a robust multi-criteria collaborative filtering algorithm. A multi-criteria CF has multiple categories in which the user can rate each item. MCCF helps in better understanding the likes and dislikes of a customer. The robustness in their method is achieved by eliminating suspicious ratings based on the degree of uncertainty. The users are also categorized into different groups based on preference similarities to restrict authentic users from mixing with attack profiles.

Deng *et al.* [110] integrated entropy scaling into the collaborative filtering process to reduce the impact of over positive and negative users. They also used a minimum threshold to invert the entropy further assisting in the prevention of random attacks.

Alonso *et al.* [111] calculated a reliability value for each prediction of a user to an item. When an unusual change is observed in the item prediction's reliability value, it indicates a possible shilling attack. They use the Matrix Factorization method to neutralize the impact of a shilling attack. Promoting such shilling predictions can be avoided to reduce the extent of the attack and neutralize the presence of shilling profiles. This method's performance drops with a decrease in the size of the attack, but it is claimed that such a small attack size has a negligible impact.

5. A PRACTICAL SHILLING ATTACK ON RECOMMENDER SYSTEMS- A MULTI-ARMED-BANDIT-BASED REINFORCEMENT LEARNING APPROACH

A version of this chapter and some preliminary experimental data have been published in IEEE MASS 2020 [112].

5.1 Introduction

Though the State-of-the-art shilling attacks can theoretically push/nuke items effectively, they come with their own pitfalls. The downside of most existing attacks is that their performance varies drastically with the type of CF algorithm used. These attacks work well in offline evaluations but cannot be used to execute an effective real-world attack. Moreover, these are single-time attacks; all the fake profiles and all their ratings are injected at once. Most attack schemes do not get feedback from the recommender system to assert the efficacy of the attack.

To overcome these drawbacks, we propose an online shilling attack scheme with high-efficiency under different CF algorithms. For an attack to work under multiple algorithms, the items selected by the attack should suit the particular algorithm under consideration, and cannot be the same for all systems. But, there is a high degree of uncertainty associated with choosing the most optimal items without knowing the algorithm. To tackle this problem, we develop a Multi-Armed-Bandit-based item selection process that uses the recommender system's feedback. We inject observer profiles, exclusively to understand and categorize the recommendations made by the system. We use these recommendations to reduce uncertainty categorically while simultaneously extending the attack reach.

Our work proposes an online attack method that aims to be efficient with different types of collaborative filtering methods used. The contributions of our work are the following:

• We design an online attack scheme that treats the recommender system as a black-box, knowing what the system is capable of doing but not the algorithm behind it.

• We employ a multi-armed bandit-based approach for selecting the most optimal items to

enhance the attack performance.

• We inject observer profiles to get the recommendations made by the CF and use these recommendations to extend attack reach.

5.2 Motivation

This section discusses the need for a Reinforcement Learning-based shilling attack. We discuss the newer developments and inferences that act as motivators for this attack.

5.2.1 Developments in Collaborative Filtering

In recent years, with the need to improve the recommenders' performance, online platforms have been including newer features into the CF process [113]. These features are modified according to the platform's needs. Some of those features are:

• The similarity of items depends on the probabilistic association between items. In other words, the number of items purchased by a customer is also taken into account while calculating the similarity.

• The similarity between items also depends on the period of the purchase/rating of the items. The items bought months apart from each other will have far less similarity score than items which are purchased on the same day.

• Sometimes, the recommender system recommends a variety of moderately related items than a narrowly targeted list. For instance, if a user only purchases books, sometimes non-books are also recommended to the user.

Intuitively, not all online platforms need these added features to be better, but there is no guarantee of whether such elements are present in a CF system. But these changes do affect the way items are recommended; each recommendation includes a fraction of the CF's hallmark. We want to build an attack scheme which works both when these developments are present, and when they are absent.

5.2.2 Detection Traits

Most of the detection algorithms work by targeting a particular trait observed in the shilling attacks. Though obfuscation manages to evade detection to some extent, some innate qualities need to be present in an attack, to be effective. Such qualities are usually targeted by the detection algorithms, both in the supervised classification and the unsupervised clustering methods. We briefly discuss what those qualities are in this section.

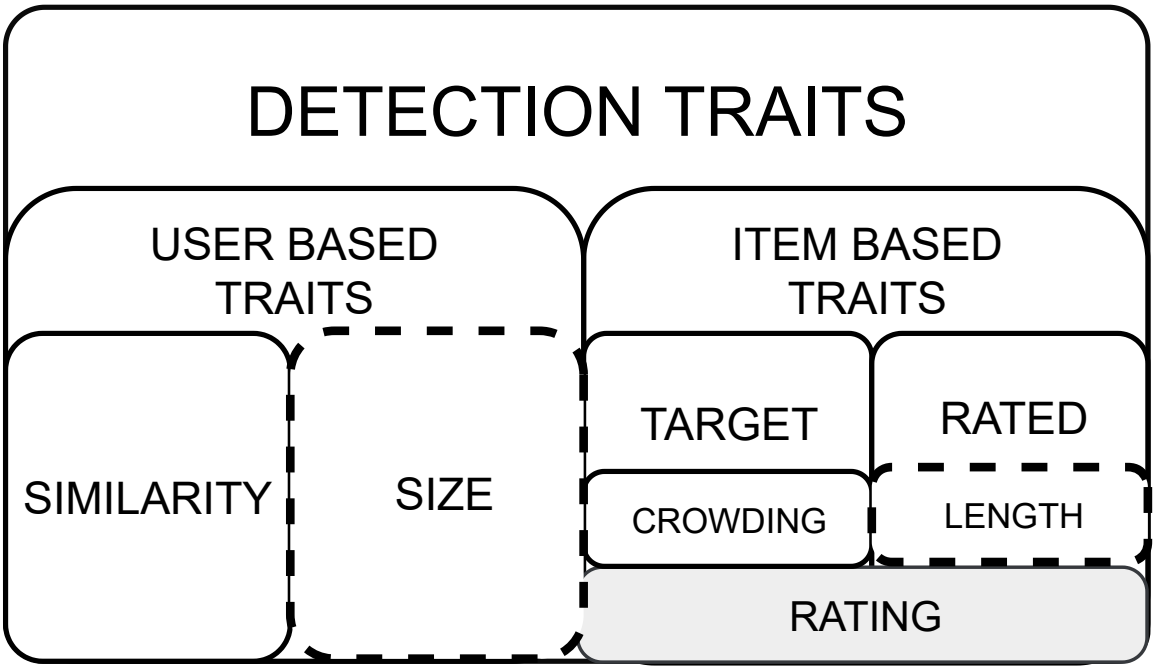


Figure 5.1. Characteristic traits of an attack which are exploited during the detection process

User-based Traits

The basic division of such detection traits comes from whether the detection algorithm is focusing on finding the attack user profiles or the items. In the user-based trait, the user's behavior is checked for anomalies, which can imply that the profile is fake.

Similarity. The similarity of a user profile to a large number of its neighbors is exhibited by most attack profiles.

Size. The size of an attack, the number of attack profiles injected, is relatively much smaller than the entire user set. This size difference, combined with the high similarity among them, prove to be useful resources in detection.

Item-based Traits

Most of the detection methods rely on the set of items rated by each profile to check if it is a fake profile or not. From a detection point of view, we can categorize the items in an attack profile into 2.

Rated Items. Rated items are the items that are used for supporting the push/nuke of the target profile. Both the selected and filler items fall into this category from a detection front.

• Length: The length of an attack profile, the number of items rated by an attack profile, is usually much higher than an ordinary profile. An attacker usually tries to increase the similarity between the attack profile and many other profiles by rating several filler items.

• Rating: The rating given to an item is maintained closer to the average rating of the item to ensure maximum similarity. Detection algorithms usually target such anomalous rating behaviors.

Target Item. The target item is the item that is promoted or demoted in an attack.

• Crowding: The concentration of users rating a target item will be abnormally high when an attack is executed. Such abnormalities have a sizeable effect on the overall rating of the item.

• Rating: The primary reason behind an attack is to modify the opinion about the target item among users. The opinion cannot be altered without giving the target item a high rating in the case of a push attack and the least possible rating in the case of a nuke attack. Usually, such ratings widely deviate from the authentic ratings given to the item.

Fig. 5.1 showcases the different types of traits used in detection. Here, both the attack size and filler length indicate the numerical differences in their behaviors and are detected using similar techniques. Likewise, the rating behavior is one of the most important features used in identifying an attack. Detection algorithms use the rating behavior differences in

one form or the other in their algorithms. We want to create an approach which can evade conventional detection algorithms built based on these traits.

5.2.3 Factors Influencing the Effectiveness of a Shilling Attack

The attackers have two significant factors that can be modified: the number of profiles injected and items rated by these profiles.

Attack Size

Attack size is the number of fake profiles injected by the attacker. The larger the attack size, the better the reach of the attack but the cost involved will also exponentially increase.

Choice of Rated Items (Selected Items)

Items to be rated by the injected profiles are selected in such a way that-

- The attack profiles have maximum similarity with other authentic users in a User-User based CF system.
- The target items have a high degree of co-occurrence with many of the other items in an Item-Item based CF system.

The attacker does not know the type of CF used in most of the cases. The existing attack schemes use offline strategies to choose these selected items. Moreover, other factors, like the developments we discussed earlier, also impact the attack but are unknown to the attacker. We want our approach to have maximum similarity with as many benign users as possible, irrespective of the CF type.

5.3 Methodology

In our method, we focus on creating an online attack, utilizing the CF's recommendation feedback while concurrently attaining maximum efficiency for a given attack size. We show that our method is system-agnostic by treating the CF as a black box; the internal parameters and the CF algorithm used are unknown to the attacker. The attacker can only rate/review

the items and view the recommendations made to him by the system. To make the attack online, we deploy a continuous attack strategy, where the selected items are added over a more extended period.

Owing to the different CF algorithms and the newer developments in CF, it is evident that each recommendation made by the system has imbibed the essence of the entire CF system. The best way to subsume the recommendations into the attack is by adding these recommended items to the fake profile's selected items list. To get the recommendations, we exclusively create multiple fake profiles (observer profiles) and fill them with some items appropriate for the target item.

Not all recommendations made by the system adds value to the attack, but only the ones relevant to the target items. This criterion leaves an *uncertainty* associated with choosing the most optimal recommended items from all the profiles created. Only a limited number of items can be added to the selected items for a given cost. Such a limitation leads to making two modifications to reduce this uncertainty.

5.3.1 First, instead of treating each profile individually, grouping them into categories, makes it easier to reduce the uncertainty as a group: Categorizing Injected Profiles.

In most of the existing works, only the attack profiles are injected into the recommender system, but in our work, we inject two types of profiles into the system.

Observer Profile

These profiles are injected to learn from the recommender system. The observer profiles use the recommendations made by the system to populate the attacker profiles.

• Selected Items Subset: These are the items that are a randomly chosen subset of the attacker profile's initial selected items.

• Random Items: These items are randomly chosen from the entire website's list of items.

• Recommended Items: These are the items that are recommended to the observer profiles after the subset of selected items and random items have been rated.

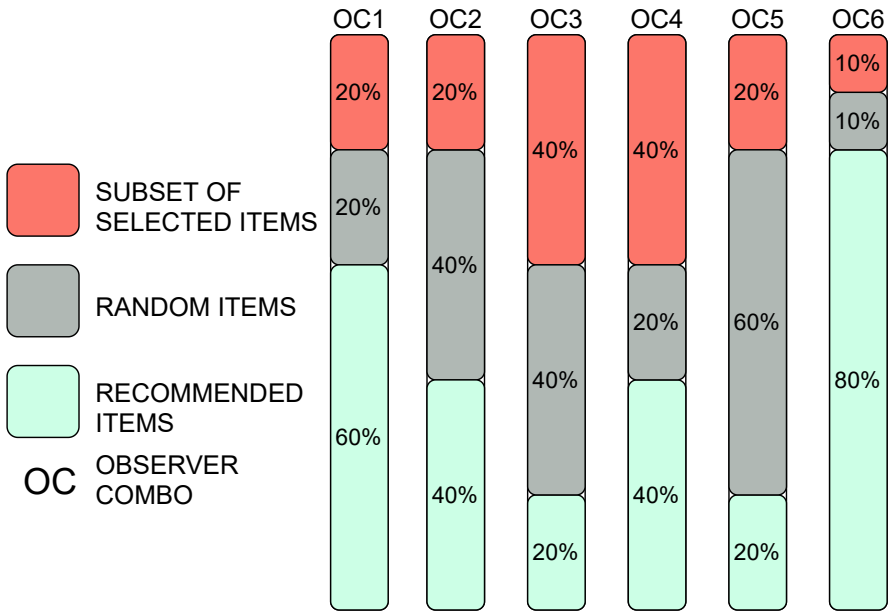


Figure 5.2. A toy example of possible observer profile combos.

The observer profiles are divided into multiple combinations, with each combination having an equal number of profiles. The combinations differ from each other based on the ratio of selected subset, random, and recommended items. Fig. 5.2 illustrates six of the possible observer profile combos. This figure is only an example of the possible combinations.

Attacker Profile

The attacker profile is the major part of the injected profiles used for promoting the target item to as many authentic users as possible. The profile has three types of rated items.

- Target Item: The item which needs to be promoted or demoted.
- Selected Items: The set of items that are rated to increase the reach of an attack.
- Filler Items: The items which are rated to camouflage the presence of an attack profile among authentic profiles to avoid detection.

5.3.2 Second, to ensure high efficiency, the uncertainty reduction process needs to happen simultaneously with the item selection proces: Multi-Armed-Bandit-based Uncertainty Reduction.

In MAB, a fixed number of resources need to be apportioned among multiple opposing choices in a way such that the overall gain is maximized [114]. The individual reward associated with each of the options is not known at the beginning. This problem models the exploration vs. exploitation dilemma. The name comes from a gambler having to choose the right one-armed bandit, or slot machine, to play from a row of bandits with varying rewards. The gambler has to select the number of plays in each bandit as well as the sequence of play. In scientific research, MABs are used in pharmaceutical trials, information retrieval, and even recommender systems. There are many optimal solutions for the MAB problem, but we are only using *Thompson Sampling* [115] approach in our method. Epsilon Greedy bandit algorithm undertakes a random exploration strategy which is not suitable for our work. Upper Confidence Bound acts under the optimistic assumption that the selection made has the highest possible reward and expends on exploring other options to decrease uncertainty. Thompson Sampling, on the other hand, is solely bayesian and is more suitable for our work.

Thompson Sampling uses the concept of probability and depends on the Beta distribution to make each selection.

$$\beta(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1} \tag{5.1}$$

Here, α value is increased by 1 for a win, and β value is increased by 1 for a loss after each iteration.

In the shilling attack process, MAB can be used to balance the exploration vs. exploitation with uncertainty in the item selection process. Each of the observer profile combos act as individual arms in the MAB. With each trial of the MAB, the uncertainty associated will the selected observer combo gets reduced. The probability of success replaces this uncertainty in each combo. If the recommended item is relevant to the target item, it is considered a win. If the recommended item is either irrelevant or is already part of the attacker profile, then it is a loss. Through multiple iterations, the MAB eventually starts exploiting the observer combo that makes the most useful recommendations.

Algorithm 2: Iterative and Termination Phase.

Input : Attacker and Observer profiles formed from Setup Phase.
Assume $\alpha = 1$ and $\beta = 1$ for all combos(bandits)

Output : Efficient Attacker and Observer profiles

1 **Iterative Phase:**
2 **MAB Selection Process:**
3 •Use eqn.5.1 to get beta distribution.
4 •Randomly sample a value from the probability density function beta distribution of all the combos;
5 •Select the combo with maximum sampled value;
6 Check recommendations for first observer profile in selected observer combo;
7 **if** *Recommended item is related to Target item* **then**
8 MAB Combo Reward = 1;
9 $\alpha = \alpha + 1$
10 Add item to all Attacker Profiles;
11 **else**
12 MAB Combo Reward = 0;
13 $\beta = \beta + 1$
14 **end**
15 Add item to current observer profile;
16 Move current observer profile to the end of queue in observer combo;
17 **Termination Phase**
18 **if** *New Items Count <= Batch Threshold* **then**
19 goto 1;
20 **else**
21 **if** *Attacker Profile Length < Filler Size* **then**
22 Terminate one batch of attack profiles;
23 Reset New Items Count;
24 Add Target item to terminated batch;
25 goto 1;
26 **else**
27 Attack Size Reached;
28 Terminate Attack;
29 **end**
30 **end**
31 **return** *Fully-populated Attacker and Observer profiles*

5.3.3 Rating Scheme

For each new item added to the attacker profile or the observer profile, a rating needs to be included. For the simplicity of discussion, we pick the two most common rating scales as examples. If the system uses the 0 - 1 rating scale, then the ratings given to all the items should be 1. If the 1 - 5 rating scale is used in the system, then the rating provided by the attacker profile for an item should be equivalent to the average rating of the item in that platform. The rating given by the observer profile should be similar to the system mean rating. These details are readily available in most of the online platforms.

5.4 MAB-based Shilling Attack Scheme

In this section, we explain the scheme in which the online attack is executed. Our attack scheme is categorized into three phases for ease of understanding.

5.4.1 Setup Phase

The selection of the items for the initial attacker profile and observer profiles constitute the setup phase. The attacker profile mostly consists of the essential selected items, which are chosen as per the target item. The most popular items which are similar to the target item are chosen to be the selected items. For example, if the target item is a sci-fi novel, then the selected items should be set of the most famous novels.

We want the initial items in the attack profile to be 70% filled with the selected items at the setup phase. The rest of the items are filler items chosen at random from the entire item set. This mix of items ensures that the attacker profile is very similar to the target item while also evading detection because of filler items. The target item is not introduced during the setup phase.

The observer profile has fewer items than the attacker profile. The number of combinations that can be used in an attack is dependent on the size of the attack. If many profiles are injected, and the number of items that can be rated by each profile is higher, then more observer combinations can be explored.

Having multiple observer combos means getting a better understanding of how the recommender system treats the different types of users. The system treats the user who has only purchased books different from the user who has consumed an array of items. So, the recommendations made to these users differ drastically. Having different observer combos helps in mimicking various types of authentic users.

5.4.2 Iterative Phase

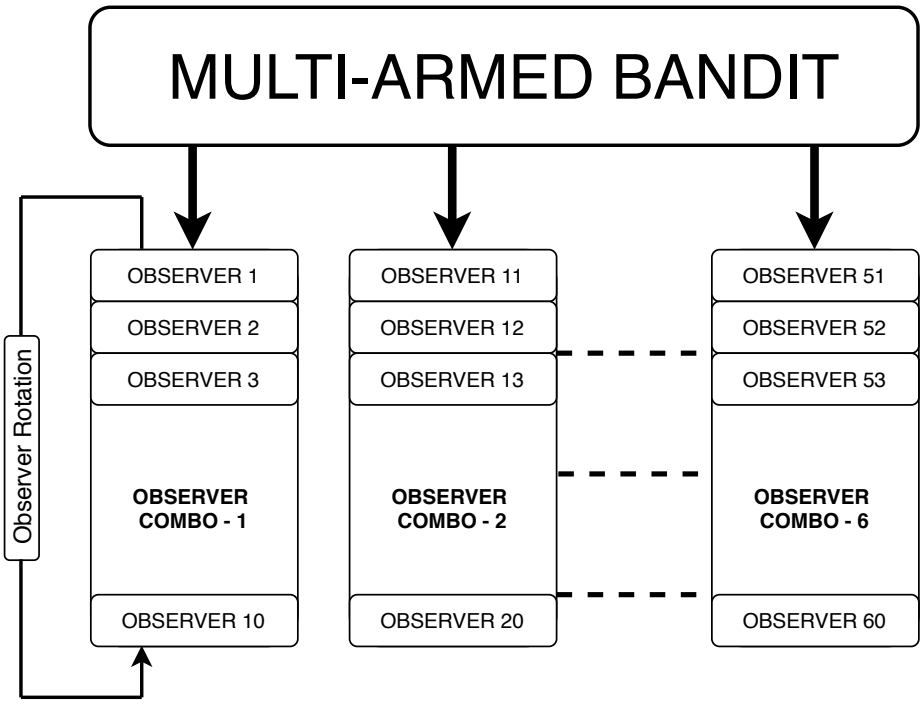


Figure 5.3. Using MAB for observer combo selection.

In the iterative phase, the recommendations made to the observer profiles are added to the attacker profile. In this phase, the multi-armed bandit algorithm is used to select the observer combo. Each observer combo has a specific number of profiles arranged in a queue; the recommendation is chosen from the first profile in the line. The observer profiles are presented as given in Fig. 5.3. If the recommended item is related to the target item, then it is added to the attacker profile as well as the observer profile. If not, then it is only added to the observer profile. Adding the recommended item to the observer profile ensures

that the item is not recommended again. After each iteration, the profile which made the recommendation is moved to the back of the queue.

Alg.2 explains the algorithm involved in the iterative and termination phases. Initially, all the observer combos are likely to be selected equally. But, as the number of iterations increases, the observer combos, which have a higher chance of giving a reward, are selected with a higher probability by the MAB.

5.4.3 Termination Phase

As the name suggests, the termination phase terminates the addition of new items to the attacker profile in batches and the termination of the attack process. Two main activities take place in the termination phase. These two actions help in tackling some of the newer developments in the CF.

First, after a threshold number of new items are added to all the attacker profiles, one batch of the attacker profile is retired after adding the target item as the final item to the batch. The size of the batch depends on the number of iteration cycles we want to continue and the attack size. Such batched termination creates attack profiles of different lengths.

As mentioned earlier, some of the more recent CF models take into consideration the number of items rated by an account for similarity calculation. By creating attacker profiles of different lengths, we are ensuring that there is some attack profile batch to match each kind of authentic users: from people who have rated only a few items to people who have rated many items.

At the termination of each batch, adding the target item creates a higher similarity between the newly added selected items and the target item. If the period in which the items were rated is taken into consideration in the CF, then the target would still have a high similarity with all the selected items.

Once the final batch reaches the predetermined maximum number of items allowed per attack profile, the attack process gets terminated. This termination keeps the estimated cost of the attack in check. After the final batch termination, the target item is added to all the observer profiles.

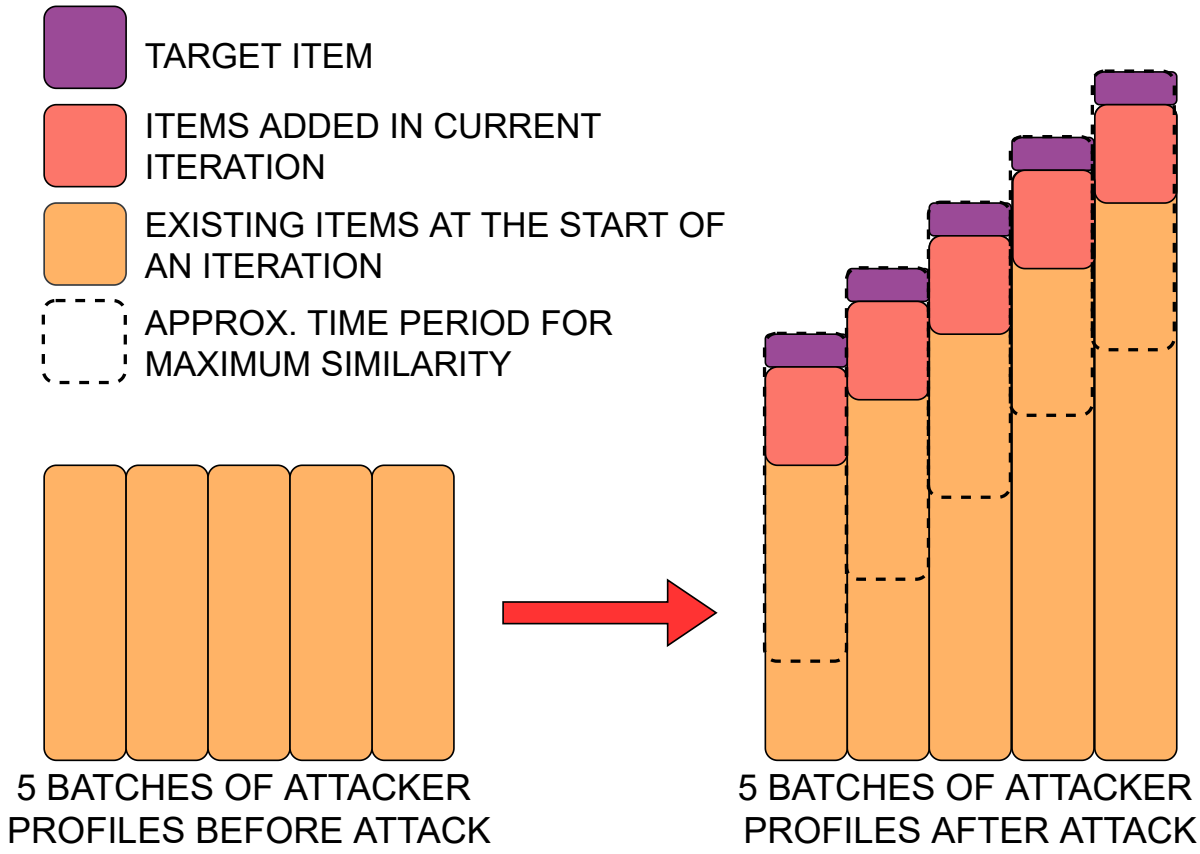


Figure 5.4. Attacker profiles at the end of the attack.

The attacker profiles at the end of the attack are illustrated in fig. 5.4. Initially, all the attacker profiles are of the same length. After the attack, the orange portion shows the items that were part of the batch during the previous iterative cycle. The blue-colored portion shows the newly added items after the termination of the previous batch. The green-colored part is the target item, the last item added before batch termination. The dotted region is a representation of the items which will have high similarity with the target if the time-period of rating is taken into consideration by the CF algorithm. By adding the target item as the last item of a batch, we are ensuring that all the newly added items get closely related to the target. It is important to note here that the different profile length or adding the target as the last item does not affect the attack even if the system doesn't use any additional features.

5.4.4 Toy Example of MAB Shilling Attack

To better understand the attack scheme, let us consider the toy example in fig. 5.5. In the example, we are injecting 200 profiles into the system, out of which 140 are attacker profiles, and 60 are observer profiles. The attacker profiles are divided into ten batches, and the observer profiles into six observer combos, with ten profiles each. The initial number of rated items in the attacker profile is 30, and that of the observer profile is 15. For this example, we are going to consider a filler size of 80 items. This size implies that no injected profile should have more than 80 rated items. Given that the filler size, initial rated items, and the number of batches in the attacker profile, we can estimate that one batch should be terminated after every 5 new items added. This way, the first terminated batch will have 35 rated items, and the last terminated batch will have 80 rated items. The batch threshold value for this attack is 5.

In the iterative phase, the MAB algorithm is used to select one of the observer combos. The recommendation made to the first profile of the chosen observer combo is examined. If the recommended item is related to the target item, then it is a win for the MAB, and the item is added to both the chosen observer and all the attacker profiles. If the recommended item is not related, then it is a loss, and the item is only added to the chosen observer. After each new item added, the control is passed to the termination phase. This process is repeated until the attacker profile length is the same as the filler size.

The first step in the termination phase is to check if the new items added after the previous batch termination are less than the batch threshold, which is 5 in the example. If so, then the control is passed back to the iterative phase. If not, the length of the current attacker profiles is inspected. If the profile length is the same or greater than filler size, then the attack is terminated. If the profile length is less than filler size, then one batch of attacker profile is terminated after adding the target item as the last new item to the batch. The control is then passed back to the iterative phase, and the process continues until the last batch has a length of 80.

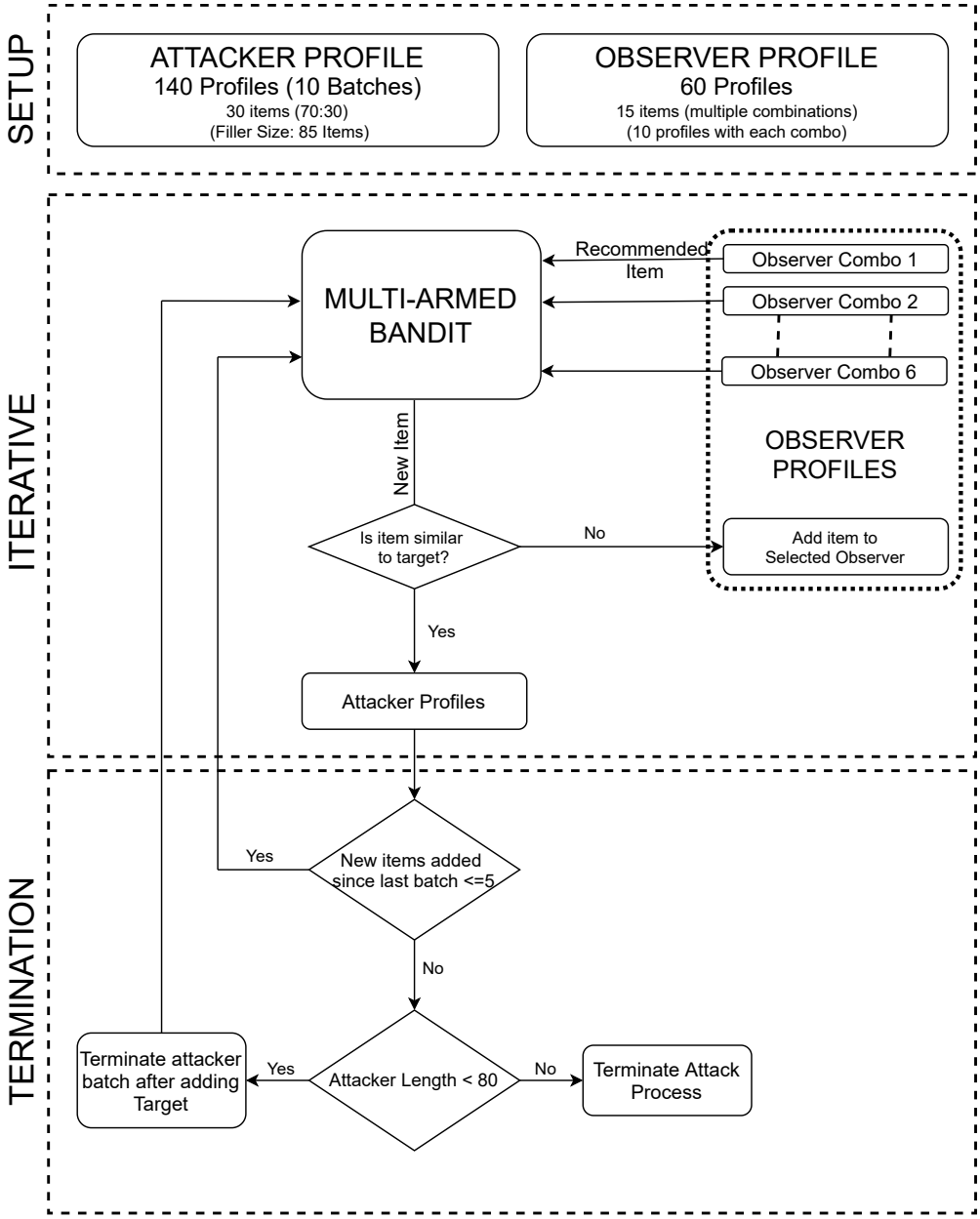


Figure 5.5. Toy example of a shilling attack with 200 injected profiles.

5.5 Experiments

In this section, we discuss the experimental evaluation of our method.

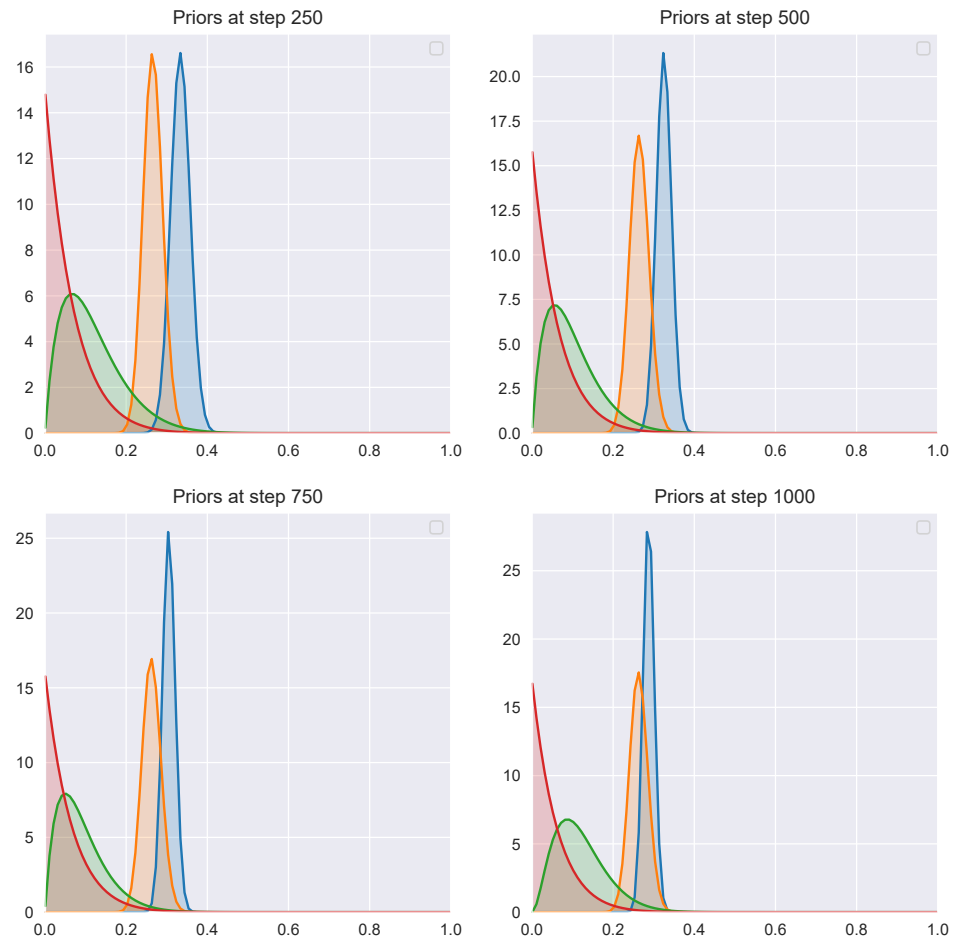


Figure 5.6. Probability density of the beta distribution for four observer combos at different steps.

5.5.1 Dataset

We used the MovieLens 1M dataset for the evaluation of our method. This dataset was collected as part of the GroupLens Research Project for their work in [116]. It consists of 1,000,209 ratings from 6040 users on 3,900 items (including movies, series, and documentaries), with the genre included. Each user has rated a minimum of 20 items. The users rate the items on a scale of 1 to 5, with 1 being the least and 5 being the maximum possible rating.

5.5.2 Our Approach

We incorporate our MAB-based attack scheme by utilizing the genre of the items. 70% of the injected profiles are attacker profiles, and the remaining 30% are observer profiles. We

choose the selected items related to the target by comparing their genre. We use four different observer combos for this attack, with 0%, 33%, 66%, and 100% selected items each, and the rest are filler items. We do not use recommended items in the observer combo formations, owing to the small number of total items. While using the MAB for item selection, if the genre of the item matches the target, then it is a win; otherwise, it is a loss.

Fig. 5.6 shows the probability density function of the beta distribution for the four observer combos at different steps while using kNN basic algorithm at 3% filler size, and 5% attack size. As the number of trials increases, the uncertainty associated with each combo decreases. Here, the color blue represents the 66% observer combo, which is the most explored arm and gives the most wins. One possible reason for this combo to be selected more often than the 100% combo could be that the 100% combo only recommends the items which are very similar to the target item. Most of the profiles in the 100% combo might have the same list of recommendations. Once an item is selected, the same item cannot be selected again, causing the trial to fail. On the other hand, the 66% combo would have a more diverse list of recommended items, leading to a higher win rate. It is important to note that the number of trials shown in the MAB, 1000, is depicted to show how the attack works. Generally, the attack terminates once the filler size is reached.

5.5.3 Baseline Attack Algorithms

We briefly reiterate the baseline algorithms against which we compare our method.

Bandwagon Attack

The Bandwagon attack operates by choosing all the rated items for the attack profiles from the most popular items in the dataset and give them high ratings. These items have a high rating from a large number of users [32].

Average Over Popular Attack

The AoP attack is a variation of the average attack, used to obfuscate the attack signature. In AoP, the rating scheme is similar to the average attack, but the items are not chosen at

random. The most popular items in the dataset are selected to be the filler items for this attack [38].

Random Attack

It is the primary form of shilling attack, where the items rated by the attack profile are chosen at random. The system overall mean is used for rating each of these items, with a standard deviation of 1.1. The target item is given the maximum rating [31].

Average Attack

This attack is a slightly more sophisticated form of the random attack. Here too, the rated items are chosen at random, but the rating is the average rating of the item. The target item is given the maximum rating [31].

5.5.4 Baseline CF Algorithms

Different websites use different algorithms to fulfill their recommendation system needs. The attacker does not know of the algorithm used, making it necessary for the attack to be successful under different algorithms.

CoClustering Algorithm

CoClustering algorithm simultaneously clusters users and items. This algorithm is designed to be a practical real-time CF approach owing to its low computational requirement. It works by generating user and item co-clusters and obtaining the average rating predictions based on these co-clusters [117].

kNN Basic

k-Nearest Neighbor algorithm works by finding the closest neighbors of a user and estimating the ratings for this user from the rating behavior of the neighboring users. The

similarity measure between two users needs to be positive for them to be considered a neighbor. This algorithm is the vanilla implementation of the kNN approach [118].

kNN with Means

This algorithm additionally takes into account the mean ratings of users in the similarity calculation process of the kNN algorithm. By including the mean rating values, the overly positive and overly negative users will not be part of an average user's neighborhood [118].

Non-negative Matrix Factorization (NMF)

NMF-based CF model is a single-element-based approach, such that none of the matrices involved in the factorization have negative elements. This algorithm works by examining the non-negative update process depending on each involved feature rather than on the entire feature matrices [119].

5.5.5 Evaluation Metrics

We conduct experiments to test the ability of the attack to push a target product. Initially, we select a target product that is not a part of the top-N recommended items of any of the 6,040 authentic users under various CF algorithms. Then we inject the same number of profiles into the system for each of the attacks. At the end of the attack, we evaluate the success of the attack by checking the percentage of authentic users with the target item as part of their top-N recommended items list. We repeat the evaluations by varying the attack size, filler size, and CF algorithms for all the attacks.

5.5.6 Result Analysis

We will discuss how the different attacks perform in the various algorithms and the impact of filler size. Fig. 5.7 and fig. 5.8 show the extent of attack in the different algorithms. The x-axis shows the attack size, and the y-axis shows the percentage of authentic users who have the target item in their top-20 recommendations after the attack.

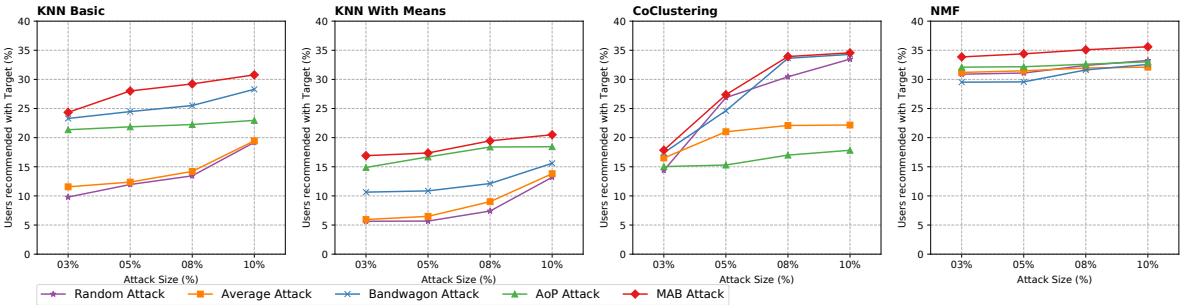


Figure 5.7. Percentage of users with target item in their top-20 recommendation list when filler size is 3%.

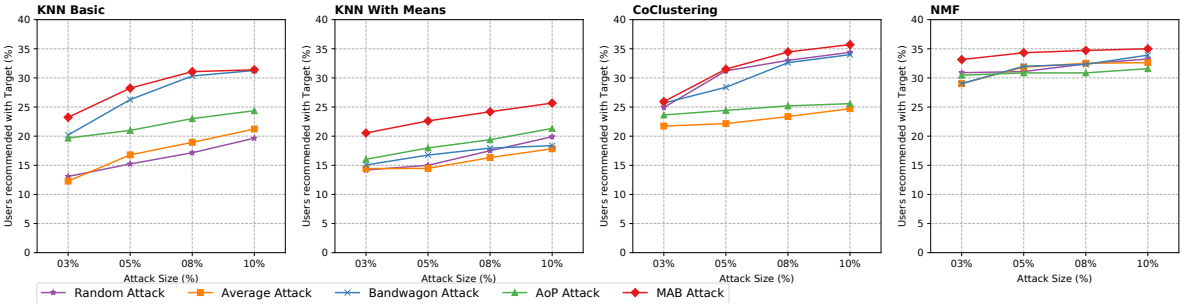


Figure 5.8. Percentage of users with target item in their top-20 recommendation list when filler size is 5%.

Changes with filler size

Fig. 5.7 shows the impact of the different attacks in different algorithms when the filler size is 3%. Similarly, fig. 5.8 shows filler size of 5%. Increasing the filler size does not hugely enhance the attack efficiency in either of the algorithms. In some instances, the similarity between attack and authentic profiles is lost because of the attack profile length. Moreover, profiles with too many rated items can easily be detected using simple detection techniques. By using attack profiles of different lengths, our approach has better reach and lesser detectible features.

Changes with CF Algorithms

One of the key advantages of using our MAB-based approach is the ability to adapt to the CF system used. By using the recommendations made by the system, MAB manages to choose the optimal items to attack the system. The MAB performance with different CFs alters with how the CF handles the recommendation process and user information.

kNN Basic: In both the 3% and 5% filler sizes, the MAB based attack seems to be outperforming the other attacks. We can notice that the average and random attacks, being the simplest, seem to be performing significantly lower than others.

kNN with Means: By including the mean rating of a user, the CF alters the outcome of the attack. In the kNN basic method, the bandwagon attack performed better than the average attack. But, giving high ratings to most of its items reduces the similarity between bandwagon's attack profiles and the authentic users in kNN with means. This aspect affects its performance.

CoClustering: When it comes to the CoClusteing approach, both average and AoP attacks have low efficiency. This outcome could be because of the rating schemes used in these approaches. Using the system mean for rating the observer items and varying filler length of attack profiles, gives MAB method an edge over the other attacks.

NMF: By observing the attack reach in this method, we can notice that there is not a significant difference between the baseline attacks. The NMF algorithm does not seem to take the attacks' rating scheme into account during the recommendation process. The item selection method used by MAB appears to be giving it an edge over the other methods.

5.6 Conclusion

In this chapter, we have explored the possibility of applying an online shilling attack, which utilizes the feedback from the recommendation system to increase the reach of the attack. The framework treats the Collaborative Filtering RS as a BlackBox and functions well with both user-user-based and item-item-based methods. We use a Multi-Armed Bandit based approach to reduce the uncertainty associated with the item selection process. Our results are encouraging and show that our online shilling attack approach has a better reach than the existing baseline methods. More research on shilling attacks is imperative as more and more businesses are using Collaborative Filtering systems. We are currently working on obfuscating our attack without affecting efficiency. In the future, we are planning to extend our approach to work on a Graph-based Recommender System as well.

6. SUMMARY

In this work, we have applied Deep Learning and Reinforcement Learning-based defense and attack, respectively, on Recommender Systems. We have shown that the Deep Dynamic Clustering approach can effectively remove the influence of spam reviewers, especially in the crowdsourced manipulators' scenario. When it comes to the shilling attack, we have proposed a practical attack method; a MAB-based reinforcement learning strategy is used to select the items to increase the attack reach effectively. On top of expanding the attack reach, our approach intuitively evades most of the standard detection techniques. It is important to note that this work incorporates sophisticated strategies while strongly emphasizing practical applicability.

REFERENCES

[1] N. Jindal and B. Liu, "Opinion spam and analysis," in *Proceedings of the 2008 international conference on web search and data mining*, ACM, 2008, pp. 219–230.

[2] J. Ye and L. Akoglu, "Discovering opinion spammer groups by network footprints," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2015, pp. 267–282.

[3] S. K. Lam and J. Riedl, "Shilling recommender systems for fun and profit," in *Proceedings of the 13th international conference on World Wide Web*, 2004, pp. 393–402.

[4] M. P. O'Mahony, N. J. Hurley, and G. C. Silvestre, "Recommender systems: Attack types and strategies," in *AAAI*, 2005, pp. 334–339.

[5] B. Mobasher, R. Burke, R. Bhaumik, and J. J. Sandvig, "Attacks and remedies in collaborative recommendation," *IEEE Intelligent Systems*, vol. 22, no. 3, pp. 56–63, 2007.

[6] Z. Yang, Z. Cai, and X. Guan, "Estimating user behavior toward detecting anomalous ratings in rating systems," *Knowledge-Based Systems*, vol. 111, pp. 144–158, 2016.

[7] V. W. Anelli, Y. Deldjoo, T. Di Noia, E. Di Sciascio, and F. A. Merra, "Sasha: Semantic-aware shilling attacks on recommender systems exploiting knowledge graphs," in *European Semantic Web Conference*, Springer, 2020, pp. 307–323.

[8] 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*, vol. 10, no. 7, pp. 1833–1845, 2019.

[9] C. E. Seminario and D. C. Wilson, "Nuking item-based collaborative recommenders with power items and multiple targets," in *The Twenty-Ninth International Flairs Conference*, 2016.

[10] P.-A. Chirita, W. Nejdl, and C. Zamfir, "Preventing shilling attacks in online recommender systems," in *Proceedings of the 7th annual ACM international workshop on Web information and data management*, 2005, pp. 67–74.

[11] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik, "Classification features for attack detection in collaborative recommender systems," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 542–547.

[12] B. Mobasher, C. Williams, R. Bhaumik, and R. Burke, "Detecting profile injection attacks in collaborative recommender systems," in *The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06)*, IEEE, 2006, pp. 23–23.

[13] Y. Hao, P. Zhang, and F. Zhang, "Multiview ensemble method for detecting shilling attacks in collaborative recommender systems," *Security and Communication Networks*, vol. 2018, 2018.

[14] Z. Yang, L. Xu, Z. Cai, and Z. Xu, "Re-scale adaboost for attack detection in collaborative filtering recommender systems," *Knowledge-Based Systems*, vol. 100, pp. 74–88, 2016.

[15] A. M. Turk and A. Bilge, "Robustness analysis of multi-criteria collaborative filtering algorithms against shilling attacks," *Expert Systems with Applications*, vol. 115, pp. 386–402, 2019.

[16] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw, "Detecting product review spammers using rating behaviors," in *Proceedings of the 19th ACM international conference on Information and knowledge management*, ACM, 2010, pp. 939–948.

[17] G. Fei, A. Mukherjee, B. Liu, M. Hsu, M. Castellanos, and R. Ghosh, "Exploiting burstiness in reviews for review spammer detection," in *Seventh international AAAI conference on weblogs and social media*, 2013.

[18] A. Mukherjee, A. Kumar, B. Liu, J. Wang, M. Hsu, M. Castellanos, and R. Ghosh, "Spotting opinion spammers using behavioral footprints," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2013, pp. 632–640.

[19] A. Mukherjee, B. Liu, J. Wang, N. Glance, and N. Jindal, "Detecting group review spam," in *Proceedings of the 20th international conference companion on World wide web*, ACM, 2011, pp. 93–94.

[20] A. Mukherjee, B. Liu, and N. Glance, "Spotting fake reviewer groups in consumer reviews," in *Proceedings of the 21st international conference on World Wide Web*, ACM, 2012, pp. 191–200.

[21] Y. Ren, Y. Zhang, M. Zhang, and D. Ji, "Context-sensitive twitter sentiment classification using neural network," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[22] Y. Ren, R. Wang, and D. Ji, "A topic-enhanced word embedding for twitter sentiment classification," *Information Sciences*, vol. 369, pp. 188–198, 2016.

[23] Y. Ren and Y. Zhang, "Deceptive opinion spam detection using neural network," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 140–150.

[24] X. Wang, K. Liu, S. He, and J. Zhao, "Learning to represent review with tensor decomposition for spam detection," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 866–875.

[25] W. Zhang, Y. Du, T. Yoshida, and Q. Wang, "Dri-rcnn: An approach to deceptive review identification using recurrent convolutional neural network," *Information Processing & Management*, vol. 54, no. 4, pp. 576–592, 2018.

[26] G. Wang, S. Xie, B. Liu, and S. Y. Philip, "Review graph based online store review spammer detection," in *2011 IEEE 11th International Conference on Data Mining*, IEEE, 2011, pp. 1242–1247.

[27] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos, "Fraudar: Bounding graph fraud in the face of camouflage," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2016, pp. 895–904.

[28] M. Zheng, C. Zhou, J. Wu, S. Pan, J. Shi, and L. Guo, "Fraudne: A joint embedding approach for fraud detection," in *2018 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2018, pp. 1–8.

[29] P. Kaghazgaran, J. Caverlee, and A. Squicciarini, "Combating crowdsourced review manipulators: A neighborhood-based approach," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, ACM, 2018, pp. 306–314.

[30] C. M. Yilmaz and A. O. Durahim, "Spr2ep: A semi-supervised spam review detection framework," in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, IEEE, 2018, pp. 306–313.

[31] S. K. Lam and J. Riedl, "Shilling recommender systems for fun and profit," in *Proceedings of the 13th international conference on World Wide Web*, 2004, pp. 393–402.

[32] M. P. O'Mahony, N. J. Hurley, and G. C. Silvestre, "Recommender systems: Attack types and strategies," in *AAAI*, 2005, pp. 334–339.

[33] B. Mobasher, R. Burke, R. Bhaumik, and J. J. Sandvig, "Attacks and remedies in collaborative recommendation," *IEEE Intelligent Systems*, vol. 22, no. 3, pp. 56–63, 2007.

[34] R. Burke, B. Mobasher, and R. Bhaumik, "Limited knowledge shilling attacks in collaborative filtering systems," in *Proceedings of 3rd International Workshop on Intelligent Techniques for Web Personalization (ITWP 2005), 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, 2005, pp. 17–24.

[35] R. Burke, B. Mobasher, R. Zabicki, and R. Bhaumik, "Identifying attack models for secure recommendation," *Beyond Personalization*, vol. 2005, 2005.

[36] C. Williams, B. Mobasher, R. Burke, J. Sandvig, and R. Bhaumik, "Detection of obfuscated attacks in collaborative recommender systems," in *Proceedings of the ECAI'06 Workshop on Recommender Systems*, vol. 94, 2006.

[37] R. Bhaumik, B. Mobasher, and R. Burke, "A clustering approach to unsupervised attack detection in collaborative recommender systems," in *Proceedings of the International Conference on Data Mining (DMIN)*, Citeseer, 2011, p. 1.

[38] N. Hurley, Z. Cheng, and M. Zhang, "Statistical attack detection," in *Proceedings of the third ACM conference on Recommender systems*, 2009, pp. 149–156.

[39] P. Adamopoulos and A. Tuzhilin, "On unexpectedness in recommender systems: Or how to better expect the unexpected," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 4, pp. 1–32, 2014.

[40] A. Durand, C. Achilleos, D. Iacovides, K. Strati, G. D. Mitsis, and J. Pineau, "Contextual bandits for adapting treatment in a mouse model of de novo carcinogenesis," in *Machine Learning for Healthcare Conference*, 2018, pp. 67–82.

[41] W. Shen, J. Wang, Y.-G. Jiang, and H. Zha, "Portfolio choices with orthogonal bandit learning," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[42] K. Misra, E. M. Schwartz, and J. Abernethy, "Dynamic online pricing with incomplete information using multiarmed bandit experiments," *Marketing Science*, vol. 38, no. 2, pp. 226–252, 2019.

[43] Q. Zhou, X. Zhang, J. Xu, and B. Liang, "Large-scale bandit approaches for recommender systems," in *International Conference on Neural Information Processing*, Springer, 2017, pp. 811–821.

[44] S. Vaswani, B. Kveton, Z. Wen, M. Ghavamzadeh, L. V. Lakshmanan, and M. Schmidt, "Model-independent online learning for influence maximization," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 3530–3539.

[45] B. Liu, T. Yu, I. Lane, and O. J. Mengshoel, "Customized nonlinear bandits for online response selection in neural conversation models," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[46] K. Ding, J. Li, and H. Liu, "Interactive anomaly detection on attributed networks," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019, pp. 357–365.

[47] S. Boldrini, L. De Nardis, G. Caso, M. T. Le, J. Fiorina, and M.-G. Di Benedetto, "Mumab: A multi-armed bandit model for wireless network selection," *Algorithms*, vol. 11, no. 2, p. 13, 2018.

[48] A. P. Sundar, F. Lilt, X. Zou, and T. Gao, "Deepdynamic clustering of spam reviewers using behavior-anomaly-based graph embedding," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*, IEEE, 2020, pp. 01–06.

[49] P. Kaghazgaran, J. Caverlee, and M. Alfifi, "Behavioral analysis of review fraud: Linking malicious crowdsourcing to amazon and beyond," in *Eleventh International AAAI Conference on Web and Social Media*, 2017.

[50] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2014, pp. 701–710.

[51] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2016, pp. 1225–1234.

[52] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*, International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[53] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, 1996, pp. 226–231.

[54] L. Akoglu, R. Chandy, and C. Faloutsos, "Opinion fraud detection in online reviews by network effects," in *Seventh international AAAI conference on weblogs and social media*, 2013.

[55] A. P. Sundar, F. Li, X. Zou, T. Gao, and E. D. Russomanno, "Understanding shilling attacks and their detection traits: A comprehensive survey," *IEEE Access*, vol. 8, pp. 171 703–171 715, 2020.

[56] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, 1994, pp. 175–186.

[57] T. Osadchiy, I. Poliakov, P. Olivier, M. Rowland, and E. Foster, "Recommender system based on pairwise association rules," *Expert Systems with Applications*, vol. 115, pp. 535–542, 2019.

[58] M. Chen, A. Beutel, P. Covington, S. Jain, F. Belletti, and E. H. Chi, "Top-k off-policy correction for a reinforce recommender system," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019, pp. 456–464.

[59] F. Amato, V. Moscato, A. Picariello, and F. Piccialli, "Sos: A multimedia recommender system for online social networks," *Future generation computer systems*, vol. 93, pp. 914–923, 2019.

[60] Y. Qian, Y. Zhang, X. Ma, H. Yu, and L. Peng, "Ears: Emotion-aware recommender system based on hybrid information fusion," *Information Fusion*, vol. 46, pp. 141–146, 2019.

[61] M. Nilashi, O. Ibrahim, and K. Bagherifard, "A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques," *Expert Systems with Applications*, vol. 92, pp. 507–520, 2018.

[62] H. Ying, F. Zhuang, F. Zhang, Y. Liu, G. Xu, X. Xie, H. Xiong, and J. Wu, "Sequential recommender system based on hierarchical attention network," in *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

[63] M. Cerna, "Modified recommender system model for the utilized elearning platform," *Journal of Computers in Education*, vol. 7, no. 1, pp. 105–129, 2020.

[64] R. Van Meteren and M. Van Someren, "Using content-based filtering for recommendation," in *Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop*, vol. 30, 2000, pp. 47–56.

[65] D. Wang, Y. Liang, D. Xu, X. Feng, and R. Guan, "A content-based recommender system for computer science publications," *Knowledge-Based Systems*, vol. 157, pp. 1–9, 2018.

[66] N. A. Albatayneh, K. I. Ghauth, and F.-F. Chua, "Utilizing learners' negative ratings in semantic content-based recommender system for e-learning forum," *Journal of Educational Technology & Society*, vol. 21, no. 1, pp. 112–125, 2018.

[67] B. R. Cami, H. Hassanpour, and H. Mashayekhi, "User preferences modeling using dirichlet process mixture model for a content-based recommender system," *Knowledge-Based Systems*, vol. 163, pp. 644–655, 2019.

[68] M. M. Islam, M. J. Hossain, and Z. B. Zahir, "Content-based health recommender system for icu patient," in *Multi-disciplinary Trends in Artificial Intelligence: 13th International Conference, MIWAI 2019, Kuala Lumpur, Malaysia, November 17–19, 2019, Proceedings*, Springer, vol. 11909, 2019, p. 229.

[69] D. Mittal, S. Shandilya, D. Khirwar, and A. Bhise, "Smart billing using content-based recommender systems based on fingerprint," in *ICT Analysis and Applications*, Springer, 2020, pp. 85–93.

[70] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[71] R. Logesh, V. Subramaniyaswamy, D. Malathi, N. Sivaramakrishnan, and V. Vijayakumar, "Enhancing recommendation stability of collaborative filtering recommender system through bio-inspired clustering ensemble method," *Neural Computing and Applications*, vol. 32, no. 7, pp. 2141–2164, 2020.

[72] S. Natarajan, S. Vairavasundaram, S. Natarajan, and A. H. Gandomi, "Resolving data sparsity and cold start problem in collaborative filtering recommender system using linked open data," *Expert Systems with Applications*, vol. 149, p. 113 248, 2020.

[73] T. Mohammadpour, A. M. Bidgoli, R. Enayatifar, and H. H. S. Javadi, "Efficient clustering in collaborative filtering recommender system: Hybrid method based on genetic algorithm and gravitational emulation local search algorithm," *Genomics*, vol. 111, no. 6, pp. 1902–1912, 2019.

[74] M. Azizi and H. Do, "A collaborative filtering recommender system for test case prioritization in web applications," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018, pp. 1560–1567.

[75] Q. Han, Í. M. d. R. de Troya, M. Ji, M. Gaur, and L. Zejnilovic, "A collaborative filtering recommender system in primary care: Towards a trusting patient-doctor relationship," in *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, IEEE, 2018, pp. 377–379.

[76] M. Si and Q. Li, "Shilling attacks against collaborative recommender systems: A review," *Artificial Intelligence Review*, vol. 53, no. 1, pp. 291–319, 2020.

[77] K. Patel, A. Thakkar, C. Shah, and K. Makvana, "A state of art survey on shilling attack in collaborative filtering based recommendation system," in *Proceedings of First International Conference on Information and Communication Technology for Intelligent Systems: Volume 1*, Springer, 2016, pp. 377–385.

[78] R. A. Zayed, L. F. Ibrahim, H. A. Hefny, and H. A. Salman, "Shilling attacks detection in collaborative recommender system: Challenges and promise," in *Workshops of the International Conference on Advanced Information Networking and Applications*, Springer, 2020, pp. 429–439.

[79] M. E. K. Badran, W. Jurdi, and J. B. Abdo, "Survey on shilling attacks and their detection algorithms in recommender systems," in *Proceedings of the International Conference on Security and Management (SAM)*, The Steering Committee of The World Congress in Computer Science, Computer …, 2019, pp. 141–146.

[80] B. Mehta and T. Hofmann, "A survey of attack-resistant collaborative filtering algorithms.," *IEEE Data Eng. Bull.*, vol. 31, no. 2, pp. 14–22, 2008.

[81] F. Zhang, "A survey of shilling attacks in collaborative filtering recommender systems," in *2009 International Conference on Computational Intelligence and Software Engineering*, IEEE, 2009, pp. 1–4.

[82] I. Gunes, C. Kaleli, A. Bilge, and H. Polat, "Shilling attacks against recommender systems: A comprehensive survey," *Artificial Intelligence Review*, vol. 42, no. 4, pp. 767–799, 2014.

[83] P. Kaur and S. Goel, "Shilling attack models in recommender system," in *2016 International Conference on Inventive Computation Technologies (ICICT)*, IEEE, vol. 2, 2016, pp. 1–5.

[84] R. Burke, M. P. O'Mahony, and N. J. Hurley, "Robust collaborative recommendation," in *Recommender systems handbook*, Springer, 2015, pp. 961–995.

[85] M. Balabanović and Y. Shoham, "Fab: Content-based, collaborative recommendation," *Communications of the ACM*, vol. 40, no. 3, pp. 66–72, 1997.

[86] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 285–295.

[87] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre, "Collaborative recommendation: A robustness analysis," *ACM Transactions on Internet Technology (TOIT)*, vol. 4, no. 4, pp. 344–377, 2004.

[88] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[89] R. Burke, B. Mobasher, R. Zabicki, and R. Bhaumik, "Identifying attack models for secure recommendation," *Beyond Personalization*, vol. 2005, 2005.

[90] R. Burke, B. Mobasher, and R. Bhaumik, "Limited knowledge shilling attacks in collaborative filtering systems," in *Proceedings of 3rd International Workshop on Intelligent Techniques for Web Personalization (ITWP 2005), 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, 2005, pp. 17–24.

[91] N. Hurley, Z. Cheng, and M. Zhang, "Statistical attack detection," in *Proceedings of the third ACM conference on Recommender systems*, 2009, pp. 149–156.

[92] R. Bhaumik, B. Mobasher, and R. Burke, "A clustering approach to unsupervised attack detection in collaborative recommender systems," in *Proceedings of the International Conference on Data Mining (DMIN)*, Citeseer, 2011, p. 1.

[93] C. Williams, B. Mobasher, R. Burke, J. Sandvig, and R. Bhaumik, "Detection of obfuscated attacks in collaborative recommender systems," in *Proceedings of the ECAI'06 Workshop on Recommender Systems*, vol. 94, 2006.

[94] V. W. Anelli, Y. Deldjoo, T. Di Noia, and F. Antonio, "Knowledge-enhanced shilling attacks for recommendation ,"

[95] P. Adamopoulos and A. Tuzhilin, "On unexpectedness in recommender systems: Or how to better expect the unexpected," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 4, pp. 1–32, 2014.

[96] C. A. Williams, B. Mobasher, and R. Burke, "Defending recommender systems: Detection of profile injection attacks," *Service Oriented Computing and Applications*, vol. 1, no. 3, pp. 157–170, 2007.

[97] F. Zhang and Q. Zhou, "A meta-learning-based approach for detecting profile injection attacks in collaborative recommender systems," *J. Comput*, vol. 7, no. 1, pp. 226–234, 2012.

[98] W. Zhou, J. Wen, Q. Xiong, M. Gao, and J. Zeng, "Svm-tia a shilling attack detection method based on svm and target item analysis in recommender systems," *Neurocomputing*, vol. 210, pp. 197–205, 2016.

[99] Y. Hao, F. Zhang, J. Wang, Q. Zhao, and J. Cao, "Detecting shilling attacks with automatic features from multiple views," *Security and Communication Networks*, vol. 2019, 2019.

[100] B. Mehta, T. Hofmann, and P. Fankhauser, "Lies and propaganda: Detecting spam users in collaborative filtering," in *Proceedings of the 12th international conference on Intelligent user interfaces*, 2007, pp. 14–21.

[101] K. Bryan, M. O'Mahony, and P. Cunningham, "Unsupervised retrieval of attack profiles in collaborative recommender systems," in *Proceedings of the 2008 ACM conference on Recommender systems*, 2008, pp. 155–162.

[102] C.-Y. Chung, P.-Y. Hsu, and S.-H. Huang, "βp: A novel approach to filter out malicious rating profiles from recommender systems," *Decision Support Systems*, vol. 55, no. 1, pp. 314–325, 2013.

[103] A. Bilge, Z. Ozdemir, and H. Polat, "A novel shilling attack detection method," *Procedia Computer Science*, vol. 31, pp. 165–174, 2014.

[104] Z. Yang, Z. Cai, and Y. Yang, "Spotting anomalous ratings for rating systems by analyzing target users and items," *Neurocomputing*, vol. 240, pp. 25–46, 2017.

[105] F. Zhang, Z. Zhang, P. Zhang, and S. Wang, "Ud-hmm: An unsupervised method for shilling attack detection based on hidden markov model and hierarchical clustering," *Knowledge-Based Systems*, vol. 148, pp. 146–166, 2018.

[106] F. Zhang, Z.-J. Deng, Z.-M. He, X.-C. Lin, and L.-L. Sun, "Detection of shilling attack in collaborative filtering recommender system by pca and data complexity," in *2018 International Conference on Machine Learning and Cybernetics (ICMLC)*, IEEE, vol. 2, 2018, pp. 673–678.

[107] Z. Luo and C. Liang, "An insider attack on shilling attack detection for recommendation systems," in *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, IEEE, 2016, pp. 277–280.

[108] L. Yang, W. Huang, and X. Niu, "Defending shilling attacks in recommender systems using soft co-clustering," *IET Information Security*, vol. 11, no. 6, pp. 319–325, 2017.

[109] A. M. Turk and A. Bilge, "A robust multi-criteria collaborative filtering algorithm," in *2018 Innovations in Intelligent Systems and Applications (INISTA)*, IEEE, 2018, pp. 1–6.

[110] D. Deng, J. J. Mai, C. K. Leung, and A. Cuzzocrea, "Cognitive-based hybrid collaborative filtering with rating scaling on entropy to defend shilling influence," in *Proceedings of the 2019 8th International Conference on Networks, Communication and Computing*, 2019, pp. 176–185.

[111] S. Alonso, J. Bobadilla, F. Ortega, and R. Moya, "Robust model-based reliability approach to tackle shilling attacks in collaborative filtering recommender systems," *IEEE Access*, vol. 7, pp. 41 782–41 798, 2019.

[112] A. P. Sundar, F. Li, X. Zou, Q. Hu, and T. Gao, "Multi-armed-bandit-based shilling attack on collaborative filtering recommender systems," in *2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, IEEE, 2020, pp. 347–355.

[113] B. Smith and G. Linden, "Two decades of recommender systems at amazon. com," *Ieee internet computing*, vol. 21, no. 3, pp. 12–18, 2017.

[114] H. Robbins, "Some aspects of the sequential design of experiments," *Bulletin of the American Mathematical Society*, vol. 58, no. 5, pp. 527–535, 1952.

[115] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.

[116] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[117] T. George and S. Merugu, "A scalable collaborative filtering framework based on co-clustering," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*, IEEE, 2005, 4–pp.

[118] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[119] 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*, vol. 10, no. 2, pp. 1273–1284, 2014.

PUBLICATIONS

In this section, a complete list of my publications at the time of writing is given below.

1) Palanisamy Sundar, Agnideven, Feng Li, Xukai Zou, and Tianchong Gao. "DeepDynamic Clustering of Spam Reviewers using Behavior-Anomaly-based Graph Embedding." In GLOBECOM 2020-2020 IEEE Global Communications Conference, pp. 01-06. IEEE, 2020.

2) Palanisamy Sundar, Agnideven, Feng Li, Xukai Zou, Tianchong Gao, and Evan D. Russomanno. "Understanding shilling attacks and their detection traits: a comprehensive survey." IEEE Access 8 (2020): 171703-171715.

3) Palanisamy Sundar, Agnideven, Feng Li, Xukai Zou, Qin Hu, and Tianchong Gao. "Multi-Armed-Bandit-based Shilling Attack on Collaborative Filtering Recommender Systems." In 2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), pp. 347-355. IEEE, 2020.