Project Report on

Shilling Attack Detection In Recommender Systems

Submitted by

Anushka Kadam (Roll No. 60)
Madhura Kamat (Roll No. 61)
Supriya Joshi (Roll No. 62)
Mark Leitao (Roll No. 63)

in partial fulfillment for the award of the degree

BACHELOR OF ENGINEERING

in
Electronics & Telecommunication Engineering

Under the Guidance of

Dr. Santosh Chapaneri

St. Francis Institute of Technology, Mumbai
University of Mumbai
2022-2023

ABSTRACT

Massive amounts of useful data are constantly being generated on the internet, making it difficult for users to find information that is relevant to them. A recommendation system is a subclass of Information filtering Systems that seeks to predict the rating or the preference a user might give to an item. In simple words, it is an algorithm that suggests relevant items to users. One of the most widely used recommendation systems is collaborative filtering. However, collaborative filtering-based recommender systems open architecture makes it susceptible to shilling attacks. Shilling attacks are a type of attack in which a malicious user profile is inserted into an existing collaborative filtering dataset to change the outcome of the recommender system. Their are two types of shilling attacks i.e. Push Attack (the attacker will give target items the highest rating) and Nuke Attack (the attacker will give target items the lowest rating). Attackers utilise user-generated data, such as user ratings and reviews, to influence recommendation ranks. Due to the widespread use of recommender systems, more attackers are disrupting the system in an effort to profit from the altered recommendation results. So, it's becoming more and more important to learn how to recognise shilling attacks. To maintain their neutrality and long-term survival, recommender systems must be able to recognise shilling attacks. Issues with the current research include poor algorithm universality, difficulty choosing user profile attributes, and a dearth of an optimization approach. The majority of detection techniques in use today identify attackers by statistical methods. However, their detection techniques were less effective since they were unable to detect the delicate interactions between people and objects. In this project, we have used CoDetector, a collaborative shilling detection model that simultaneously decomposes the user-item interaction matrix and the user-user co-occurrence matrix with

shared user latent variables.Then, the network embedding information contained in the learned user latent factors is used as a feature to identify attackers. CoDetector outperforms state-of-the-art techniques, according to tests done on both simulated and real-world datasets. CoDetector also has a good performance and generalization capacity.

CERTIFICATE

This is to certify that Anushka Kadam, Madhura Kamat, Supriya Joshi and Mark Leitao are the bonafide students of St. Francis Institute of Technology, Mumbai. They have successfully carried out the project titled "SHILLING ATTACK DETECTION IN RECOMMENDER SYSTEMS" in partial fulfilment of the requirement of B. E. Degree in Electronics and Telecommunication Engineering of Mumbai University during the academic year 2022-2023. The work has not been presented elsewhere for the award of any other degree or diploma prior to this.

(Dr. Santosh Chapaneri)
Internal Guide

Internal Examiner

External Examiner

(Dr. Kevin Noronha)
EXTC HOD

(Dr. Sincy George)
Principal

Project Report Approval for B.E.

This project entitled ***'Shilling Attack Detection In Recommender Systems Approach'*** by **Anushka Kadam, Madhura Kamat, Supriya Joshi, Mark Leitao** is approved for the degree of Bachelor of Engineering in Electronics and Telecommunication from University of Mumbai.

Examiners

1. - - - - - - - - - - - - - - - - - -

2. - - - - - - - - - - - - - - - - - -

Date:

Place:

ACKNOWLEDGEMENT

We are thankful to a number of individuals who have contributed towards our final year project and without their help; it would not have been possible. Firstly, we offer our sincere thanks to our project guide, Dr. Santosh Chapaneri for his constant and timely help and guidance throughout our preparation.

We are grateful to all project co-ordinators for their valuable inputs to our project. We are also grateful to the college authorities and the entire faculty for their support in providing us with the facilities required throughout this semester.

We are also highly grateful to Dr. Kevin Noronha, Head of Department (EXTC), Principal, Dr. Sincy George, and Director Bro. Shantilal Kujur for providing the facilities, a conducive environment and encouragement.

Signatures of all the students in the group

(Anushka Kadam)

(Madhura Kamat)

(Supriya Joshi)

(Mark Leitao)

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included; we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in this submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Signatures of all the students in the group

(Anushka Kadam)

(Madhura Kamat)

(Supriya Joshi)

(Mark Leitao)

Contents

List of Figures

List of Abbreviations

RS	Recommender Systems
CBFS	Content Based Filtering Systems
CFS	Collaborative Filtering Systems
SVD	Singular Value Decomposition
MF	Matrix Factorization
MDP	Markov decision process
DegSim	Degree of Similarity with Top Neighbors
RDMA	Rating Deviation from Mean Agreement
UB-CF	User-Based Collaborative Filtering
IB-CF	Item-Based Collaborative Filtering
TRS	Tourism Recommendation Systems
AI	Artificial Intelligence
I_T	Target Items
I_S	Selected Items
I_F	Filler Items
I_N	Unrated Items
LFM	Latent Factor Model
LSI	Latent Semantic Index
PCA	Principal component analysis
RMSE	Root Mean Square Error
SGNS	Skip-gram model with negative sampling
PMI	Pointwise mutual information
SPPMI	Shifted positive pointwise mutual information

Chapter 1

Introduction

The growth of e-commerce has made information overload a serious problem. Because of this issue, online users find it difficult to find the relevant content for them. To address this issue, use of recommender system is done. A recommender system, is a subclass of information filtering system that suggests the most relevant items to a particular user. Simply put, it is an algorithm that suggests relevant articles to users. For instance, choosing which Netflix movie to watch, which e-commerce item to purchase, which Kindle book to read, etc. There are a number of different types of recommender systems. The main two approaches are collaborative filter recommendation systems and content based recommender systems. The first approach is collaborative based recommendation system. A collaborative filter recommendation system is a type of machine learning algorithm that predicts what a user might want to buy or watch based on the past behavior of other users. Based on what other users with similar tastes have purchased or highly rated, the algorithm chooses items for the new user. The second approach is content based recommendation system. In order to make recommendations, a content based recommender system compares items that are similar to one another. Content-based filtering methods are built on the description of a product and a profile of the user's preferred options. A user profile is created in this recommendation system to express the types of products the user is interested in. Products are described using keywords. Its ability to recommend a variety of items, such as movies, without requiring a deep understanding of the content of the item itself is one of the main benefits of recommender systems that use a collaborative filtering approach. Contrarily, in case of content based filtering strategy, it might require extra details like the actors genre and roles. Comparing a user's collected data to similar and dissimilar data collected from other users, collaborative filtering based recommender

systems determine a user's list of suggested products. However, because collaborative filtering recommender systems are open, they are susceptible to attacks from malicious users who introduce skewed ratings into their user profiles. To manipulate the user's choice and the recommendation ranking, some users insert fake user profiles with biased ratings. Attacks on the behavior of the recommender system are referred to as "shilling" or "profile injection" attacks. An attack known as a "shilling attack" involves inserting a false user profile into an already existing collaborative filtering dataset in order to change how the recommender system performs. The injected profiles expressly assign ratings to items that either elevate or defame the target item. This lowers a recommender system's ability to make accurate and reliable recommendations. As a result, one of the major difficulties in recommender system research is shilling attack detection.

Generally, shilling attacks can be seen as a binary classification problem, [2] that is, for each user profile, the classified result can only be a normal user or an attacker. Therefore, the key point for this problem is to appropriately design the user features [3]. Most existing detection models recognize attackers in statistics-based manners [4, 5]. However, they failed in revealing the fine-grained interactions between users and items, [2] leading to a degradation in detection accuracy when attackers are disguised elaborately.

Collaborative filtering recommendation systems frequently employ the matrix factorization technique. Its goal is to factorize a user-item matrix into two low-ranking matrices, the user-factor matrix and the item-factor matrix, which can forecast new items that users might be interested in. The matrix factorization (MF) algorithm is one of the potent model-based collaborative filtering algorithms that has proven successful for examining the sparseness of data problem of recommender systems. The purpose of matrix factorization is to create a low-rank matrix approximation of the user rating matrix, which may have an impact on the recommendations made to users. Latent factors derived from MF capture the implicit features [2] underlying the interactions between users and items. In this project, we are using CoDetector as detection model which is based on MF. However, to further include more information, we bridge basic MF and the word embedding model [6] which can uncover the context structure of users. Inspired by [7] , the model jointly factorizes the rating matrix and the user-user co-occurrence matrix with shared user latent factors. For each pair of users, the user-user co-occurrence matrix encodes the number of items they both consumed, which is similar to the word co-occurrence matrix

in word-embedding models. The main idea of CoDetector model is that attackers tend to promote the target item in group [8] so as to enhance the attack effect. Therefore, factorizing these two matrices can fuse rating preferences and structural information in the user-item bipartite network into the user latent factors, [2] which are the input of the classifier. The experimental results show that CoDetector has a good performance and generalization capacity and significantly outperforms state-of-the-art methods in real scenarios.

1.1 Motivation

The main motivation for developing this current project started by observing the intruder who are injecting fake profiles in the system by various ways. An attacker can insert fake profiles into the system either manually or by using an automated tool to insert them into the system. If the system is very small i.e. number of users is very less and so the reviews or ratings they provide manually by insertion technique is also efficient. Although inserting profiles through manual approach into the system take lots of effort and time. In small system even small number of fake profiles can promote and demote a particular item. But if our system is large containing thousands of users and billions ratings, in this condition small number of fake users do not cause much effect in the system. And if attacker tries to insert large number of profiles manually then it will not be an optimal solution as well as its time consuming. In this scenario, the attacker must adopt the automation tools to insert the biased profiles in the system resulting in recommendations that favors or disfavors a given item. A collaborative recommendation system must be open to user inputs, so it is very difficult to design a collaborative recommender system that cannot be attacked. So researchers are mainly focused on the defending against such shilling attacks.

1.2 Recommender Systems (RS)

A recommender system, or a recommendation system, is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. In simple words, it is an algorithm that suggests relevant items to users. They are primarily used in commercial applications. Recommendation systems use specialized algorithms and machine learning solutions. The recommendation system is powered by automated configuration, coordination, and management of machine learning predictive analytics algorithms, allowing it to make informed decisions about which filters to use depending on the circumstances surrounding a given user. It facilitates marketers to maximize conversions and average order value. As powerful personalization tools, recommendation systems leverage machine learning algorithms and techniques to give the most relevant suggestions to particular users by learning data, and predicting current interests and preferences. Data required for recommender systems stems from explicit user ratings after watching a movie or listening to a song, from implicit search engine queries and purchase histories, or from other knowledge about the users/items themselves. Sites like Spotify, YouTube or Netflix use that data in order to suggest playlists, so-called daily mixes, or to make video recommendations, respectively. Recommender systems work by using users data to generate suggestions. The following image shows a simple recommender system.

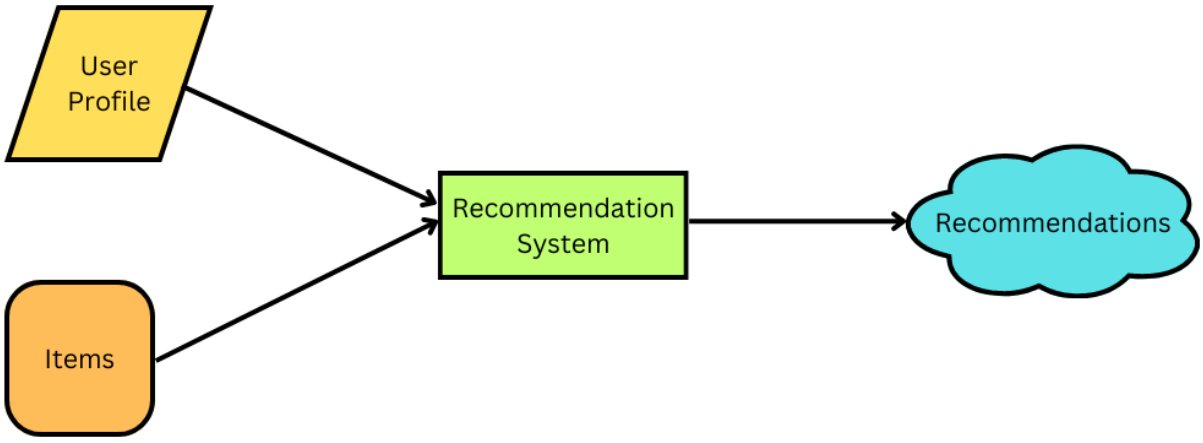

Figure 1.1: Recommender System (RS)

1.3 Need for Recommender System

Companies using recommender systems focus on increasing sales as a result of very personalized offers and an enhanced customer experience. Recommendations typically speed up searches and make it easier for users to access content they are interested in, and surprise them with offers they would have never searched for. What is more, companies are able to gain and retain customers by sending out emails with links to new offers that meet the recipient's interests, or suggestions of films and TV shows that suit their profiles. The user starts to feel known and understood and is more likely to buy additional products or consume more content. By knowing what a user wants, the company gains competitive advantage and the threat of losing a customer to a competitor decreases. Providing that added value to users by including recommendations in systems and products is appealing. Furthermore, it allows companies to position ahead of their competitors and eventually increase their earnings.

1.4 Issues with Recommender Systems

While the extreme high growth shows that businesses all over the world are exploring what recommendation engines can do for them, effectively using this technology comes with its fair share of challenges. Some of the most common issues with recommender systems are as follows :

- **Cold Start Problem :** When new customers or new products enter e-commerce platforms, a cold start occurs. Traditional recommender systems, such as collaborative filtering, make the assumption that each user or item has some ratings, allowing us to extrapolate ratings of comparable users or items even in the absence of those ratings. However, this becomes challenging for new users/items because it wont have any browsing, clicking, or purchasing information for them. Because of this, it cannot fill the blank using standard matrix factorization methods.

- **Sparsity :** Generally, majority of the users do not rate most of the items and consequently the ratings matrix becomes very sparse. Due to this, the data sparsity problem arises that declines the chances of finding a set of users with similar ratings.

This is the most eminent drawback of the recommender system. This concern can be alleviated by using some additional domain information.

- **Scalibility :** A crucial and significant issue for recommender systems today is the scalability of algorithms with large real-world datasets. The management of large and dynamic data sets produced by user interactions with products, such as preferences, ratings, and reviews, is getting more and more challenging. When applied to very large amounts of data, some recommendation algorithms may behave unfavorably or ineffectively. Some recommendation algorithms may perform best when applied to small amounts of data. Therefore, to address this issue, some cutting-edge large-scale assessment methodologies are needed.

- **Privacy issue :** To generate personalized recommendations that are of the highest caliber, recommender systems must gather as much user data as they can and make the most of it. The user might, however, have a negative perception of their privacy because the system already knows so much about them. Therefore, it is important to create techniques that can use user data intelligently, thoroughly, and carefully while preventing malicious users from having free access to data on the users' actual preferences.

- **Robustness :** Another major challenge in recommender systems is its robustness to attacks. Robustness is a performance measure of recommender systems. To gain certain profits, an attacker may generate some fake user profiles based on some attack models, such as Push/Nuke Attacks to make some target items more/less popular respectively. Such attacks are collectively called shilling attacks or profile injection attacks.

1.5 Types of Recommender Systems

The types of Recommender Systems are :

- Content based filtering systems
- Collaborative filtering systems

1.5.1 Content Based Filtering Systems (CBFS)

Content-based filtering is one popular technique of recommendation or recommender systems. The content or attributes of the things user like are referred to as "content". In order to predict and make suggestions to the user for new but related items, content-based filtering recommender systems use machine learning algorithms. Product recommendations based on characteristics are only possible with a clear list of the product's features and a list of the user's options. The recommender system stores previous user data like clicks, ratings, and likes to create a user profile. The more a customer engages, the more accurate future recommendations are. Here, the system uses user's features and likes in order to recommend the user with things that user like. It uses the information provided by user over the internet and the ones they are able to gather and then they curate recommendations according to that. The goal behind content-based filtering is to classify products with specific keywords, learn what the customer likes, look up those terms in the database, and then recommend similar things. This type of recommender system is hugely dependent on the inputs provided by users, some common examples included Google, Wikipedia, etc.

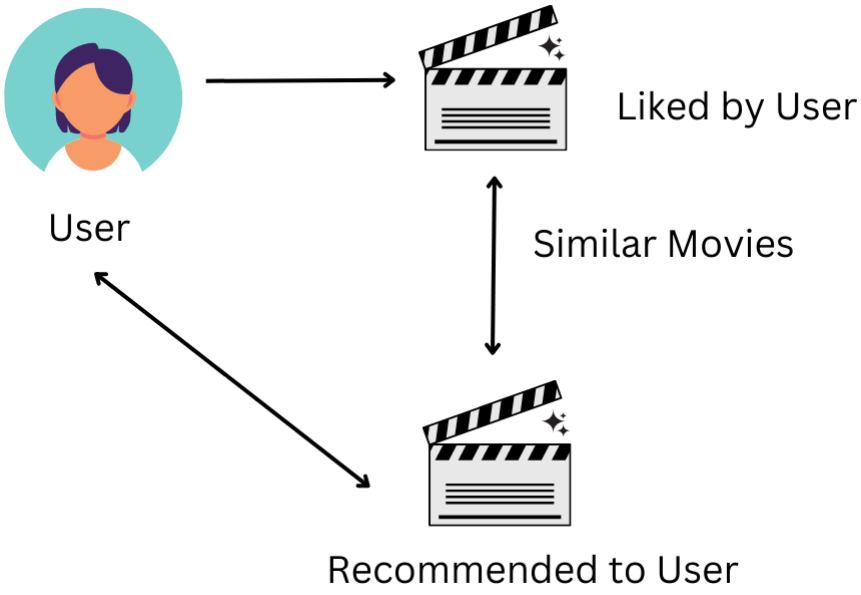

Figure 1.2: Content Based Filtering System (CBFS)

1.5.2 Collaborative Filtering Systems (CFS)

Collaborative filtering is currently one of the most frequently used approaches and usually provides better results than content-based recommendations. Some examples of this are found in the recommendation systems of Youtube, Netflix, and Spotify. Collaborative Filtering Based Recommender Systems solely rely on past interactions between users and items in order to suggest new products. The features of every individual item are not considered.In collaborative filtering, the historical data of the user interacting with the items is recorded and stored. This is usually represented by a matrix known as user-item interaction matrix, where rows represent users and columns represent the items. Similar users are grouped and all their interactions are considered when making recommendations to the target user. Collaborative filtering filters information by using the interactions and data collected by the system from other users. It's based on the idea that people who agreed in their evaluation of certain items are likely to agree again in the future. The concept is simple: when anyone wants to find a new movie to watch they often ask their friends for recommendations. Naturally, anyone will have greater trust in the recommendations from friends who share tastes similar to their own. Most collaborative filtering systems apply the so-called similarity index-based technique. In the neighborhood-based approach, a number of users are selected based on their similarity to the active user.

Inference for the active user is made by calculating a weighted average of the ratings of the selected users. Collaborative filtering systems focus on the relationship between users and items. The similarity of items is determined by the similarity of the ratings of those items by the users who have rated both items.

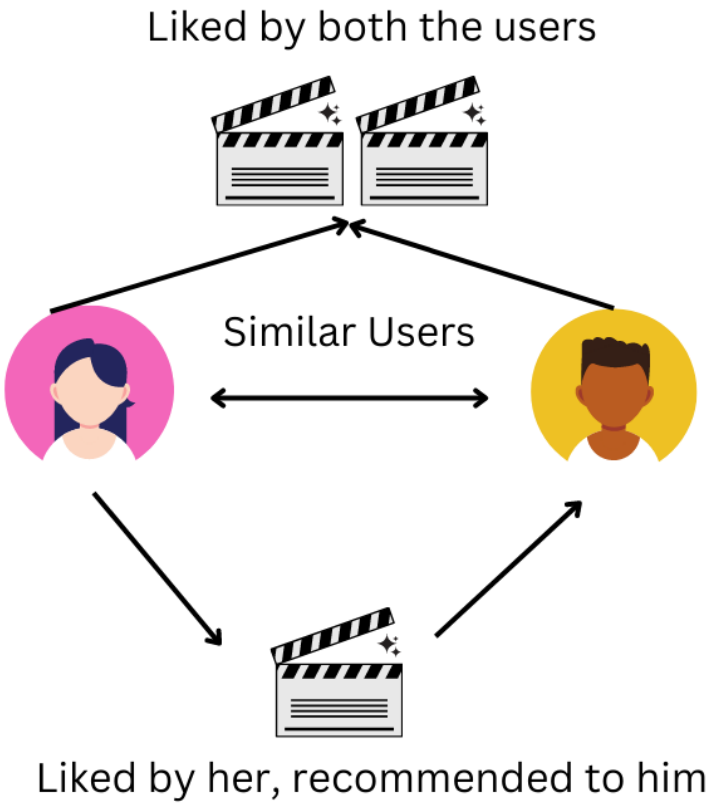

Figure 1.3: Colaborative Filtering System(CLFS)

There are two types of the collaborative filtering process:

- Memory Based Collaborative Filtering
- Model Based Collaborative Filtering

Memory Based Collaborative Filtering :

The Memory Based Collaborative Filtering method relies on the assumption that predictions can be made on the pure "memory" of past data. It uses the past ratings to predict the preferences of one user by searching for similar users, or the "neighbors"

that share the same preferences. That is why the method is sometimes referred to as neighborhood-based.

Memory-based collaborative filtering computes similarities between users or items and predicts a new rating for an item by taking the weighted average of ratings from the similar group. There are two types in this approach: user-based filtering and item-based filtering. Well known for its relative effectiveness and easy implementation, Memory Based Collaborative Filtering is limited in terms of recommendation variety. It's unlikely to provide real-time recommendations or those not so popular among users.

There are two ways to achieve Memory Based Collaborative Filtering through:

- **User Based Collaborative Filtering (UB-CF) :** User Based Collaborative Filtering makes recommendations based on the user's preferences that are similar to other users.

- **Item Based Collaborative Filtering (IB-CF) :** Item Based Collaborative Filtering suggests items similar to other items the active user liked.

Model Based Collaborative Filtering :

Model-based collaborative filtering is not required to remember the based matrix. Instead, the machine models are used to forecast and calculate how a customer gives a rating to each product. These system algorithms are based on machine learning to predict unrated products by customer ratings. Model-Based procedure facilitates machine learning techniques such as Singular Value Decomposition (SVD) and Matrix Factorization (MF) models to predict the end user's rating on unrated items.

The main drawback of Memory Based technique is the requirement of loading a large amount of in-line memory. The problem is serious when rating matrix becomes so huge in situation that there are extremely many persons using system. Computational resource is consumed much and system performance goes down; so system can't respond user request immediately. Model-based approach intends to solve such problems. There are four common approaches for model-based CF such as clustering, classification, latent model, Markov decision process (MDP), and matrix factorization.

1.6 Applications of Recommender Systems

Now that we've covered some of the basic benefits and terminology, we'll explore applications of recommendation engines across large and well-known companies and sectors.

- **Amazon.com**

  For most of their internet pages and email campaigns, Amazon.com uses item-to-item collaborative filtering recommendations. McKinsey estimates that recommendation algorithms play a role in 35% of Amazon transactions.

- **Netflix**

  Netflix is another data-driven company that leverages recommendation systems to boost customer satisfaction. Mckinsey study highlights that 75% of Netflix viewing is driven by recommendations. In fact, Netflix is so obsessed with providing the best results for users that they held data science competitions called Netflix Prize where one with the most accurate movie recommendation algorithm wins a prize worth $1,000,000.

- **Spotify**

  Every week, Spotify generates a new customized playlist for each subscriber called "Discover Weekly" which is a personalized list of 30 songs based on users' unique music tastes. Their acquisition of Echo Nest, a music intelligence and data-analytics startup, enable them to create a music recommendation engine that uses three different types of recommendation models: Collaborative filtering , Natural language processing , Audio file analysis.

- **LinkedIn**

  Online professional social networks such as LinkedIn play a key role in helping job seekers find right career opportunities and job providers reach out to potential candidates. LinkedIn's job ecosystem has been designed to serve as a marketplace for efficient matching between potential candidates and job postings, and to provide tools to connect job seekers and job providers. LinkedIn's job recommendations product is a crucial mechanism to help achieve these goals, wherein personalized sets

of recommended job postings are presented for members based on the structured, context data present in their profiles.

- **Banking**

  A mass-market product that is consumed digitally by millions. Banking for the masses and SMEs are prime for recommendations. Knowing a customer's detailed financial situation, along with their past preferences, coupled with data of thousands of similar users, is quite powerful.

- **Tourism** Tourism Recommendation Systems (TRS) provide suggestions to the tourists to identify the most suited transport (flight, train, etc.), accommodations, museums, special interest places and other items which are required for the trip. Several techniques are used and a thorough study of various techniques of traditional RS and TRS techniques have been done which are specially designed for tourism domain. Various Artificial Intelligence (AI) techniques have been highlighted which are used to solve the tourist recommendation problem.

Chapter 2

Related Work

Collaborative filtering is a popular recommendation algorithm [9] which is very susceptible to shilling attacks. The attacker falsifies the user profile and make the fake users become close neighbors of genuine users as more as possible. Since collaborative filtering is recommended based on the interests of neighbours, the attacker can influence the system's recommendation results and increase or decrease the recommended frequency of the target object. While there is a lot of work in the field of developing collaborative filtering algorithms, only recently some papers have concentrated on developing shilling attack models [32, 33]and on benchmarking the robustness of recommender systems against shilling attacks [34, 35]. The word "shilling" was first coined by Lam.SK and J.Riedl [10] which is nothing but injecting some fake profiles inside the server records and try to post fake reviews on the products which are not at all purchased by any one. 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. Shilling attack contains two intents [11] : (1) increase the recommendation frequency of target items, namely push attack; (2) reduce the recommendation frequency of target items, namely nuke attack. In [41], Mehta et al. focus exclusively on robust collaborative filtering techniques and not on detection techniques or attack strategies. In [42], the types of attacks and the detection techniques discussed are limited. In 2014, [43] produced one of the most comprehensive surveys on the topic, but it presents details on the attacks only until 2011. The survey in [44] focuses only on the statistical measures used in the detection and the basic shilling attack methods. Kaur and Goel [19] perform experimental evaluation comparing

the most commonly used shilling attack methods. In [45] and [46], the discussions do not consider the different detection attributes used in supervised and unsupervised detection methods. Both [47] and [48] briefly discuss the various attack and detection methods. There is no discussion on robust algorithms, and the detection methods are not categorized. Recently, research into recognizing and minimizing the effects of profile injection attacks has increased. The majority of this shilling attack consists of a number of attack profiles, each of which links fabricated user identities to biased rating information. It is challenging to distinguish between "shilling" and genuine profiles despite the fact that they both exhibit similar behaviors. It has been examined that the attacks can be mounted successfully even with the little knowledge of the system. These attack profiles are mainly depend on either random or average attack models which were introduced originally in [10] and used in [12]. For classification of shilling attacks, users profiles in the system are represented in a feature space. There are several features proposed in the literature, such as degree of similarity with top neighbors (DegSim) [13], rating deviation from mean agreement (RDMA) [13], weighted deviation from mean agreement(WDMA) [58] and weighted degree of agreement(WDA) [58]. These features are called as generic attributes which capture characteristics of attack profiles and make attack profiles to be easily identified from the rating profile database. In addition, model-based attributes are proposed under the assumption that the attack models which generate the attack profiles are known. For example, filler mean variance and filler mean target [59] are used for the average attack. For the bandwagon and segment attack, filler mean target difference (FMTD) [59] is used as features for the classification. In addition, the attack profile injection can not only affect the result of recommendation systems but also can breach privacy of personal sensitive data. It is shown that [60] the individual's rating history can be successfully revealed by the inference attack on KNN-based collaborative filtering. In this type of attack, attackers can create fake neighbors of the active user based on some auxiliary information.

The creation of attack profiles using ratings that were arbitrarily assigned to the profile filler items is required for both of these attack models. Lam and Riedl [34] introduce the Random Bot and the Average Bot types of shilling attacks and evaluates their effectiveness in promoting the target items by computing the prediction shift and expected Top-N occupancy for these items in both user-based and item-based collaborative filtering

environments. A Random Bot attacker rates all the items in the system with the mean 3.6 out of 5 and a 1.1 deviation. The intuition behind this is that making random ratings [13] within a certain average interval will allow the attacker to have a high influence in making predictions for other users. Depending on the objective of the attack, the items in the target set are rated with the minimum rating (for nuke attack) or maximum rating (for push attack). An Average Bot attacker is more effective but requires knowledge of the average rating for each item in the system. Each Average Bot attacker rates the items outside the target set randomly, following a normal distribution with a mean equal to the average rating for that item, thus becoming more similar to the real users than the Random Bot. [19] In this, automated CF was used for generating the recommendations. They proved that item based algorithm offers advantage over user based algorithm because in item based algorithm, attacks are not much successful in altering the system's result.Amazon.com uses item-item CF approach [49]. For finding the similar customers, cluster model is used. Its main motive is to assign target user to the cluster having most similar users. Performance and scalability of cluster model is better than traditional CF algorithms. Mobasher experimentally proved that item based approach also suffers from profile injection attacks [50]. [19] Attacks consist of attack profiles that contain biased data associated with malicious users. It has been examined that the attacks can be mounted successfully even with the little knowledge of the system [51, 52, 35]. Segment based attacks against CFRSs has been introduced which ensures that the item pushed by the attacker would be recommended to the target users [53]. Zhang [54] examined that topic level recommendation algorithm based on trust is more secure. It incorporates topic oriented trust model into CF algorithms under average attack and he concluded that this CF algorithm has more stability than standard kNN approach under average attack. Zhang [55] proposed an average hybrid and bandwagon attack model, analyzed their effectiveness against trust-based recommendation algorithm and showed that the proposed hybrid attack model has more impact on recommendations generated by system than other attack models. Donovan and Smyth introduced trust based models in CF with the aim to improve the accuracy [56]. They concluded that even trust based models are more vulnerable to shilling attack. In a later research, they modified the trust building process and solved this problem, which reduces the prediction shift by 75 percent as compared to classic CF algorithm. Calculating the similarity between the users is difficult because of

sparse data in the matrix of ratings, so a trust metric is required to solve this problem. Avesani and Massa introduced a robust CF algorithm based on "web of trust" metric [57]. In [33] several other attack models are developed, under the assumption that the attacker has some knowledge about the ratings of the other users. We think that such knowledge is hard to obtain, if not impossible in a real world system. Another disadvantage of this approach is that the ratings introduced by an attacker are algorithm dependent. Finally, the detection of the attackers is not addressed in the paper. In fact, the only work that partially tackles this challenge is [36]. There, a spreading similarity algorithm is developed in order to detect groups of very similar shilling attackers. While this is indeed a first step, it only applies to a simplified attack scenario, whereas our algorithm applies to more general and powerful attack. We think that zero-knowledge attacks such as the Random Bot are particularly interesting, since for the other attacks, recommender systems administrators could increase the privacy of user profiles using cryptographic meansvc[37, 38, 39], thus falling back to the zero-knowledge ones. In general, the more insight a recommender system offers about its ratings, the more susceptible to attack it is, allowing powerful low cost (in terms of number of fake profiles) attacks to be mounted on the system.

A number of researchers have proposed detection algorithms to be against on recommended attacks, which can mainly be generalized as statistical, classification, clustering, and data dimensionality reduction methods.

Based on statistical methods: Hurley et al. [61] utilized the Neyman-Person statistical principle presenting a user profile detection algorithm. Li and Luo [62] proposed a probabilistic Bayesian network model to test that a new user profile is normal or malicious.

Supervised classification methods: Chirita et al. [51] presented two attributes of user profile, the rating deviation from mean agreement (RDMA) and the degree of similarity with top neighbors, and utilized a classification approach for detecting malicious users. However, when the user profile was very sparse, it didn't have an obvious distinction between the attributes of user profile, therefore it couldn't detect malicious users effectively. Subsequently, Burke et al. [58] showed three more efficient user profile characteristic attributes on the basis of RDMA, which include the length variance (LengthVar), the weighted degree of agreement and the weighted deviation from mean agreement, and their algorithm took into account these additional attributes of user profile, construct-

ing a classifier through training set to potential attack users. Although it included more characteristics of the user, the classification-based detection algorithm always had low precision and false positives.

Unsupervised clustering methods: Mehta and Nejdl [18] introduced a PLSA-based clustering method to divide users and determine them to be utilized in the recommending generation process instead of using the traditional nearest neighbor method. Bhaumik et al. [64] adopted the k-means clustering method to detect sham user groups.

Data dimensionality reduction methods: Mehta et al. [63]applied PCA-based variable selection by computing the covariance between pairs of users to locate malicious users in the system. Although it can effectively detect malicious users, this algorithm demanded to know the number of attack user profiles injected into the recommender system, which is difficult to be estimated in real applications.

This section focuses on research into collaborative filtering recommender systems' ability to detect attacks. Attack detection algorithms fall into three categories: supervised, unsupervised, and semi-supervised.

Supervised detection model: A number of studies have employed supervised method to detect shilling profiles. In order to train a model, [1] labelled data is used in these supervised methods, and the quality of labelled data influence the detecting result directly. Classifiers are trained through samples and labels. Moreover, these methods are only applicable for detecting known types of attacks [65]. Features of attack profiles are extracted and supervised detecting method is built on these features. These methods only consider individual user's features but ignore the relationship between attack profiles. Moreover, these supervised methods do not perform well in blurring attack profile detection. So, this type of detection methods usually recognize attackers [2] by elaborately designing user features. [13] computes rating indicators like DegSim (Degree of Similarity with Top Neighbors) and RDMA (Rating Deviation from Mean Agreement) for all users. [5] extracts popularity patterns based on items analysis.

Semi-supervised detection model: The data in recommendation systems consists of large amount of unlabeled data and a small amount of labeled data. It will be a great lost of information if we discard those unlabeled data [67]. Unsupervised based methods address shilling attack issues by training unlabelled datasets. Some assumptions and prior knowledge are needed [1] before perform these methods. These methods involve

much less computation than supervised methods. The benefit of doing this is that these methods can be used in online detection. Some techniques use clustering, association rule methods, and statistical methods. In the real situation, there exist few labeled data [2] and much unlabeled data. Thus, semi-supervised detection models make use of both unlabeled and labeled user profiles for shilling attacks. [14] proposed a model which firstly use naive bayes to train an initial classifier and then improve it with unlabeled data. Wu et al. [68]proposed HySAD detection model which collects many detection metrics for selecting features via a wrapper called MC-Relief and the semi-supervised Naive Bayes for classification. In [16] , a model based on PU-Learning [15] which relies on a few positive labels and much unlabeled data to construct a classifier iteratively was introduced.

Unsuppervised detection model: Unsupervised learning model does not rely on labels. Semi-supervised based methods [1] use both unlabelled and labelled profiles. Semi-supervised learning is a learning technique between supervised learning and unsupervised learning, which learns both tagged and untagged data. It is hard to get enough labelled data, semi-supervised based methods perform well with less labelled data than supervise based methods.[2] Compared with supervised detection algorithms, unsupervised ones are more applicable to real scenarios because of less labeled data set. It usually clusters by calculating the similarities between users to recognize spammers. For example, the unsupervised learning detector PCA VarSelect which was proposed by Mehta et al.[18]. It uses PCA to calculate the covariance between users and distinguishes the category of them by results. Yang [66] proposed an unsupervised method that combines both saving computation time and improving detection effect. It consists of three phases. Firstly, an undirected user graph is constructed from the original user history rating information. The graph mining method is used to estimate the similarities between the vertices. Then, the similarities are utilized to distinguish the differences between vertices, which is used to exclude real users. Finally, the method further filters out the remaining real users by analyzing target items. [17] proposed a graph-based detection approach which finds most associated sub-matrices in a user-user similarity matrix for shilling attacks. In [18] the authors exploited the similarity structure in shilling user profiles to separate them from normal user profiles using principal components analysis. Shilling attack models will evolve with changes [1] in shilling detection methods. Once attackers are aware of shilling detection mechanisms, they will react quickly reduce the effectiveness of the detection

method. Thus, there is a downside to fixed shilling attack methods.

Chapter 3

Shilling Attacks in Collaborative Filtering

To skew the recommendation ranking and influence the user's choice, some users insert fake user profiles with biased ratings. They use several fake attack profiles to increase their attack effectiveness, either to encourage their own desired products or downgrade others. There are several attributes for every attack profile, including attack size, content, and type. Attack size may represent the number of injected fake profiles or ratings in each of them. Shilling attacks or "profile injection attacks" target the behavior of recommender systems. Attackers and shillers are terms used to describe users who engage in shilling attacks. Shilling attacks are further classified as push attack and nuke attack. In push attack, the attacker will give target items the highest rating i.e. the product will be promoted. In nuke attack, the attacker will give target items the lowest rating, i.e. the product will be demoted. Different attack models and profiles have been created. To stop such attacks, numerous detection methods and algorithms have simultaneously been developed. 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.

An attack model is an approach to constructing attack profiles based on knowledge of the recommender system, its rating database, its products, and/or its users. There are various attack profiles that can be used to attack a collaborative filtering recommender system. An attack profile consists of an m-dimensional vector of ratings, where m is the total number of items in the system. An attack consists of a set of target items,

biased data, and attack profiles that skew the recommender system's results in favor of the attacker. Many attack models have been discovered, each based on a different premise regarding the knowledge and intent of the attacker. In recommender systems, the four most common attack models are random attack, average attack, bandwagon attack, and love-hate attack. An attack profile's ratings can be broken down into four sets of items. Selected items (I_S), Target item(s) (I_T), Filler items (I_F) and Null items (I_N). Selected items is a set of selected items that represents a small group of items that have been selected because of their association with the target item. For some of attacks this set may be empty. These are generally chosen randomly. Tagret item is a attribute clearly identify that, always there will be only one target item in the items set. Filler items is the set of filler items represents a group of randomly selected items in the database which are assigned ratings according to the proportion of the attack. Size of the selected item set is small, so size of each profile (total number of ratings) is determined by the size of filler item set. Null items is the set of items that are not rated by attack profile. The knowledge already in existence gleaned from the recommender system determines the quality of the filler items. A generated attack becomes more sophisticated as more knowledge is gained. The main distinction between attack models is how filler item ratings are calculated. The variance rating distribution in filler items differentiates attack models.

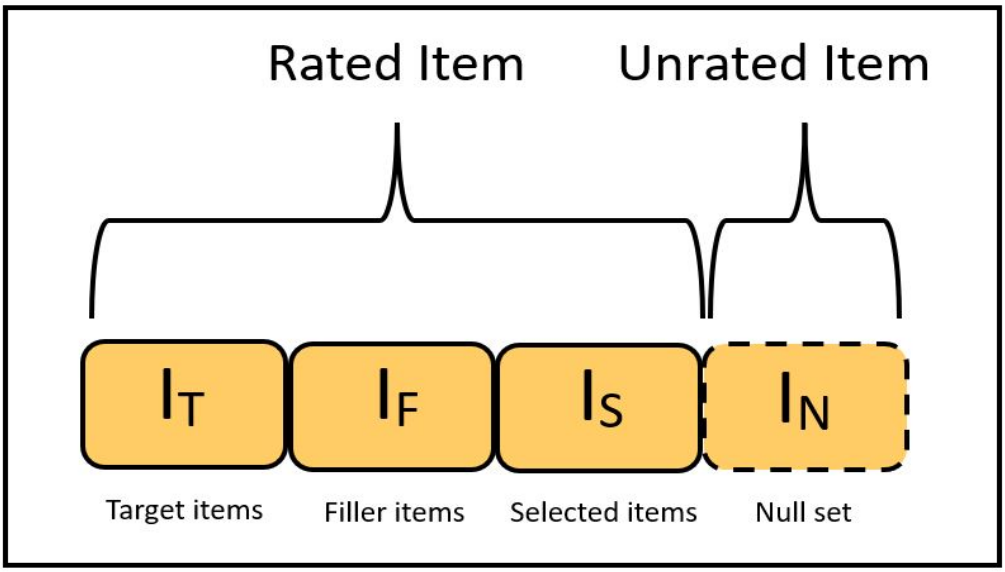

Figure 3.1: Attack Profile

3.1 Attack Dimensions

Profile injection attacks can be categorized based on the knowledge required by the attacker to mount the attack, the intent of a particular attack, and the size of the attack. From the perspective of the attacker, the best attack against a system is one that yields the biggest impact for the least amount of effort. While the knowledge and effort required for an attack is an important aspect to consider, from a detection perspective, the main focus is on how these factors combine to define the dimensions of an attack. From this perspective the main concern is for the dimensions of:

- **Attack Model:** The attack model specifies the rating characteristics of the attack profile. The model associated with an attack details the items that should be included in the attack profiles, and the strategy that should be used in assigning ratings to these items.

- **Attack Intent:** The intent of an attack describes the intent of the attacker. Two simple intents are "push" and "nuke". An attacker may insert profiles to make a product more likely ("push") or less likely ("nuke") to be recommended. Another possible aim of an attacker might be simple vandalism to make the entire system function poorly. Here it assumes a more focused economic motivation on the part of the attacker, namely that there is something to be gained by promoting or demoting a particular product.

- **Profile Size:** The number of ratings assigned in a given attack profile is the profile size. The addition of ratings is relatively lower in cost for the attacker compared to the creating of additional profiles. However, there is the additional factor of risk at work when profiles include ratings for a large percentage of the rateable items. Real users rarely rate more than a small fraction of the rateable items in a large recommendation space. No one can read every book that is published or view every movie. So, attack profiles with many, many ratings are easy to distinguish from those of genuine users and are a reasonably certain indicator of an attack.

- **Attack Size:** The attack size is the number of profiles inserted related to an attack. Here it assumes that a sophisticated attacker will be able to automate the profile injection process.Therefore, the number of profiles is a crucial variable because it is

possible to build on-line registration schemes requiring human intervention, and by this means, the site owner can impose a cost on the creation of new profiles.

3.2 Types of Attacks

Standard attack models are those 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. Prior research on the stability of recommender systems mainly focused on four different attack models:

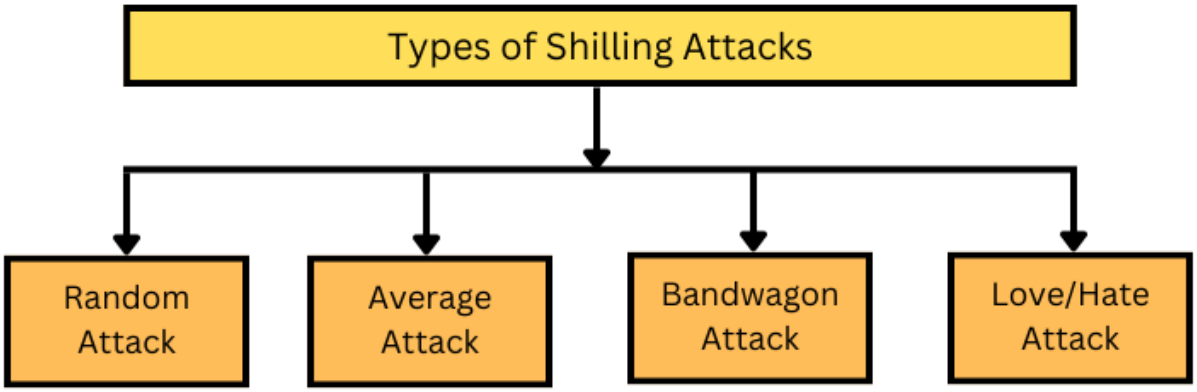

Figure 3.2: Types of Shilling Attacks

- **Random Attack**

  Random attack also known as RandomBot attack [21], is the simplest form of shilling attack. Random attack model is a low knowledge attack [19], in which filler items (I_F) are selected in a random manner and rate them by using normal distribution with standard deviation and mean rating of the system. In this model, the selected item set is empty i.e. $(I_S) = \phi(\text{null})$. The set of targeted items are rated with minimum or maximum depending on the type of attack i.e. nuke or push. For e.g., rating is in between 1 and 5, where 5 means liked item and 1 means disliked item, therefore, in push attack $r_{target} = 5$ and in nuke attack, $r_{target} = 1$. Some attacks are intended to disrupt the trustworthiness of a recommender system, known as random vandalism [21]. 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 [21] 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**

  Average attack model is a more sophisticated attack model [20] than random attack model and requires knowledge of the average rating of each item in the recommender system. But it is impractical to implement because it requires knowledge about the system [19], as it uses individual average ratings for each item instead of global mean of the system. Attackers rate items in the filler set randomly using a normal distribution with mean set to the average rating of the filler item being rated and the standard deviation. By introducing the average attack model, attackers disguise themselves and are harder to differentiate when compared to genuine users, thus, have a large effect on recommendations. As with the random attack model, the ratings of target items are set to either the maximum or minimum allowable rating based on the intention of the attack. This attack model is considered as high knowledge attack as it requires the average rating of individual item. However, this attack is not much effective in case of item based algorithm

- **Bandwagon Attack**

  In Bandwagon attack model, attacker takes advantage of Zipf's law distribution of popularity [19] and generates the biased profiles that contain most popular items. Popular items are those items that are rated by lots of users. Therefore, there is a high possibility that attackers become similar to the actual users. Bandwagon attack model is the type of attack where the profiles generated by attackers [21] 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. This attack model is considered as low knowledge attack because in order to determine the popular products in any product space, knowledge required about

the system is less.

- **Love/Hate Attack**

  The love/hate attack is a very simple attack [69], with no knowledge requirements. Love/Hate attack model is a highly effective nuke attack. Here, the attacker randomly chooses filler items [21] 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.

The attack models' features are summarized in the table below.

Table 3.1: Attack Models

Attack Models	I_S		I_F		I_T
	Selection	Rating	Selection	Rating	Rating
Random	$\emptyset$		Random	System mean	max/min
Average	$\emptyset$		Random	Item mean	max / min
Bandwagon	Popular	max / min	Random	System mean	max /min
Love/Hate	$\emptyset$		Random	max	$-$/ min

3.3 Matrix Factorization (MF)

Matrix factorization, also referred as the Latent Factor Model (LFM) [22], basically extracts the user's historical rating from the whole database and integrates them into a rating matrix. The specific algorithm separates the matrix into the production of several matrices. The recommendation system competition hosted by Nettlix in 2009 showed the high-level matrix factorization methods that many competitors used is of great help to enhance the prediction accuracy. The MF recommendation algorithms covers the advantages owned by model-based recommendation algorithms, which include simple computation and high prediction accuracy, but it lacks good interpretability.

Some of the most successful realizations of latent factor models are based on matrix factorization. In its basic form, matrix factorization characterizes both items and users by vectors of factors inferred from item rating patterns. High correspondence between item and user factors leads to a recommendation. These methods have become popular in recent years by combining good scalability with predictive accuracy. In addition, they offer much flexibility for modeling various real-life situations.

Recommender systems rely on different types of input data, which are often placed in a matrix with one dimension representing users and the other dimension representing items of interest. The most convenient data is high-quality explicit feedback, which includes explicit input by users regarding their interest in products. For example, Netflix collects star ratings for movies, and TiVo users indicate their preferences for TV shows by pressing thumbs-up and thumbs-down buttons. We refer to explicit user feedback as ratings. Usually, explicit feedback comprises a sparse matrix, since any single user is likely to have rated only a small percentage of possible items. One strength of matrix factorization is that it allows incorporation of additional information. When explicit feedback is not available, recommender systems can infer user preferences using implicit feedback, which indirectly reflects opinion [23] by observing user behavior including purchase history, browsing history, search patterns, or even mouse movements. Implicit feedback usually evidences the presence or absence of an event, so it is typically represented by a densely filled matrix.These methods have become popular in recent years by combining good scalability with predictive accuracy. In addition, they offer much adaptability for modeling various real-life situations.

Most of the MF models are based on the latent factor model [23]. Matrix Factorization approach is found to be most accurate approach to reduce the problem from high levels of sparsity in RS database, certain studies have used dimensionality reduction techniques. In the model-based technique Latent Semantic Index (LSI) and the dimensionality reduction method Singular Value Decomposition (SVD) are typically combined [23, 24]. SVD and Principal component analysis (PCA) are well-established technique for identifying latent factors in the field of Information Retrieval to deal with CFS challenges. These methods have become popular recently by combining good scalability [25] with predictive accuracy. They offers much flexibility for modeling various real-life applications.

Firstly, let U be the set of users, I be the set of items. Let R be the matrix of size $|U| \times |I|$ that contains all the ratings that the users have assigned to the items. Now the latent features [25] would be discovered. Our task then, is to find two matrices, P ($|U| \times |K|$) and Q ($|I| \times |K|$) such that their product approximately equals to R is given by:

$$\mathrm{R} \approx \mathrm{P} \times \mathrm{Q}^{\mathrm{T}} = \hat{\mathrm{R}} \tag{3.1}$$

In this way, the Matrix factorization models map both users and items to a joint latent factor space of dimensionality *f*, user-item interactions are modeled as inner products in that space [23]. Accordingly, each item i is associated with a vector [25] $\mathbf{q}_i \in \mathbb{R}^f$, and each user u is associated with a vector $\mathbf{p}_u \in \mathbb{R}^f$. For a given item i, the elements of $\mathbf{q}_i$ measure the extent to which the item possesses those factors positive or negative. The resulting dot product $\mathbf{q}_i^T \mathbf{p}_u$captures the interaction between user u and item i, the users overall interest in the item characteristics. This approximates user u's rating of item i which is denoted by r_{ui} leading to estimate [23] :

$$\hat{r}_{ui} = \mathbf{q}_i^T \mathbf{p}_u \tag{3.2}$$

The major challenge is computing the mapping of each item and user to factor vectors $\mathbf{q}_i, \mathbf{p}_u \in \mathbb{R}^f$. After the recommender system completes this mapping, it can easily estimate the rating a user will give to any item by using equation 3.2.

Such a model is closely related to singular value decomposition (SVD), a well-established technique for identifying latent semantic factors in information retrieval. Applying SVD in the collaborative filtering domain requires factoring the user-item rating matrix. This often raises difficulties due to the high portion of missing values caused by sparseness

in the user-item ratings matrix. Conventional SVD is undefined when knowledge about the matrix is incomplete. Moreover, carelessly addressing only the relatively few known entries is highly prone to overfitting.

Earlier systems relied on imputation to fill in missing ratings and make the rating matrix dense. However, imputation can be very expensive as it significantly increases the amount of data. In addition, inaccurate imputation might distort the data considerably. Hence, more recent works suggested modeling directly the observed ratings only, while avoiding overfitting through a regularized model. To learn the factor vectors $(\mathbf{q}_i, \mathbf{p}_u)$, the system minimizes the regularized squared error on the set of known ratings as [23] :

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} \left(r_{ui} - q_i^T p_u\right)^2 + \lambda \left(\|q_i\|^2 + \|p_u\|^2\right) \tag{3.3}$$

Here, K is the set of the *(u, i)* pairs for which r_{ui} is known the training set. The constant λ controls the extent of regularization and is usually determined by cross-validation.

3.4 Matrix Factorization (MF) Models

3.4.1 Singular Value Decomposition (SVD)

Singular Value Decomposition, abbreviated as SVD, is one of the factorization algorithms [40] for collaborative filtering [Zhang et al., 2005]. This type of algorithm finds the features of users and objects, and makes predictions based on these factors. Some factorization algorithms have additional restrictions on each single feature value, or between the feature vectors of multiple users, but Singular Value Decomposition does not impose restrictions and is easier to implement.

Singular Value Decomposition[26] technique was first used in 2000 for Recommendation systems. SVD reduces the dimensionality of a user-item rating matrix [27] and generate low rank matrix approximations, which represents the latent features (hidden features) of users and items inherent in rating matrix. For example, latent features can be price or brand for product, genres for movies and so on. The low rank approximations are further used for recommendation tasks for a new and unknown user-item rating. The main challenge in SVD lies in finding a lower dimensional rank matrix. The Singular Value Decomposition (SVD) is a powerful technique of dimentionality reduction. It is

particular realization of the MF approach.

SVD is a well-known method for establishing latent factors in the scope of recommendation systems to work on problems faced by collaborating filtering technique. These techniques have become popular due to good scalability and better predictive accuracy. We use SVD in recommender systems to perform two different tasks: First, we use SVD to capture latent relationships between customers and products that allow us to compute the predicted likeliness of a certain product by a customer. Second, we use SVD to produce a low-dimensional representation of the original customer-product space and then compute neighborhood in the reduced space. We then used that to generate a list of top-N product recommendations for customers.

SVD of an $m \times n$ matrix $\mathbf{A}$ is of the form [28, 29]:

$$SVD(\mathbf{A}) = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{3.4}$$

Where,

$\mathbf{U}$ and $\mathbf{V}$ are $m \times m$ *and* $n \times n$ are orthogonal matrices respectively.

An $m \times m$ matrix $\mathbf{U}$ is called orthogonal if $\mathbf{U^T U}$ equals to an $m \times m$ identity matrix. The diagonal elements in $\mathbf{\Sigma}$ $(\sigma_1, \sigma_2, \sigma_3, \ldots\ldots \sigma_n)$ are called singular values of matrix $\mathbf{A}$. Usually the singular values are placed in descending order in $\mathbf{\Sigma}$. The column vectors of $\mathbf{U}$ and $\mathbf{V}$ are called left singular vectors and the right singular vectors respectively [29].

SVD has many desirable properties and is used in many important applications. One of them is low rank approximation of matrix $\mathbf{A}$. The truncated SVD of rank k is defined as :

$$\text{SVD}\,(\mathbf{A_k}) = \mathbf{U_k \Sigma_k V_k^T} \tag{3.5}$$

Where,

$\mathbf{U_k and V_k}$ are m $\times$ k and n $\times$ k matrices composed by the first k columns of matrix $\mathbf{U}$ and the first k columns of matrix V respectively. Matrix $\mathbf{\Sigma_k}$ is $k \times k$ principle diagonal sub-matrix of $\mathbf{\Sigma_k}$.

$\mathbf{A_k}$ represents the closest linear approximation of the original matrix $\mathbf{A}$ with reduced rank k.

Once the transformation is completed, user and items can be thought off as points in the $k-$dimentional space.

Thus, applying SVD performs dimensionality reduction. The prediction and recommendation task is carried out by first imputing missing values [27] in rating matrix and then performing matrix factorization. The SVD provided better performance than memory based approach, on e-commerce and movie data. However, the imputation causes distortions in data and makes the recommendation model less accurate. Also, this imputation incurs data overfitting problem and are significantly expensive, as they increase the volume of data.

3.5 Word Embedding

Word embedding represents a set of successful models in natural language processing. Using these methods, each word in a sequence of words can be embedded [2] into a continuous vector space. Skip-gram model with negative sampling (SGNS) in word2vec is a neural model which trained with the negative-sampling procedure. It also proves that Skip-gram model with negative sampling (SGNS) is equivalent to factorizing a word-context matrix, whose cells are the pointwise mutual information(PMI) of the respective word and context pairs. PMI between a word w and a context c is an information-theoretic measure, can be empirically estimated as:

$$PMI(i,j) = \log \frac{\#(i,j) \cdot |D|}{\#(i) \cdot \#(j)} \tag{3.6}$$

where (i, j) is the number of times word j appears in the context of word i , $\#(i) = \sum_j \#(i,j)$ and $\#(j) = \sum_i \#(i,j)$ $\|D\|$ is the total number of words-context pairs in the corpora.Then [2] proposed SPPMI (Shifted Positive PMI) based on PMI with different negative samples count k to improve the resulting embedding.

$$SPPMI(i,j) = \max\{PMI(i,j) - \log k, 0\} \tag{3.7}$$

3.6 CoDetector

Attackers manipulate recommendation results by injecting biased user profiles in a large scale [2], causing abnormalities not only in the given ratings but also in the local clusters in the user-item bipartite graph. Therefore, to further capture the fine-grained character-

istics of attackers, both rating and structural information are supposed to be fused into the user features. In CoDetector, we adopt user embedding to discover the anomalies.

3.6.1 User Embedding

The principle of user embedding is the same as word embedding. We set the target user as the center user, and set other users who rate the same items as the target user does as the context of the target user. The attackers handle the recommendation result by injecting a large range of false rating records, which not only causes an abnormity in the given rating, but also causes an abnormity of the local cluster in the user project bipartite graph. Therefore, to further capture the fine-grained features of the attacker, both the user's rating preference information and the structural information should be integrated into the user's features. In the user-item rating matrix, once we confirm the center user, we can regard other users who have similar rated items with center user as his context. Then, the user-user SPPMI (Shifted Positive Point Mutual Information) matrix is built based on calculating the number of common user rating items. By factorizing SPPMI matrix is able to further describe the latent interactions among users than just by factorizing basic matrix. In word2vec, given a center word, the sequence of words surrounding it are defined as the context of the center word. Likewise, in the user-item bipartite graph of the recommender system [2], we can define the context of a user u as other users who consumed or rated same items. For example, both u_1, u_2 consumed i_1, i_2 , therefore, u_1, u_2 are context of each other. Then, the user-user co-occurrence SPPMI matrix $M \in \mathbb{R}^{m \times m}$ is constructed by computing $\#(i, j)$ that denotes the number of items which both user i and user j consumed. After that, we can obtain user embedding by factorizing M. As attackers tend to promote/demote target items in group, factorizing Shifted positive pointwise mutual (SPPMI) matrix can reveal the implicit interactions among attackers in the user-item bipartite graph and embed structural information into user latent factors. In addition, by tuning the value of negative samples, noises in the bipartite can be neglected whereas available connections are preserved.

3.6.2 Training Procedures

To fuse both rating and structural information into user latent factors, CoDetector jointly decomposes the rating matrix R and the SPPMI matrix M [2] with shared user latent factors. The overall process is shown in below Algorithm. The objective function of CoDetector is stated as:

$$\begin{aligned} L = & \sum_{u,i} \left(y_{ui} - p_u^T q_i\right)^2 + \sum_{u,j} \left(m_{uj} - p_u^T g_j - w_u - c_j\right)^2 \\ & + \lambda \left(\sum_u \|p_u\|^2 + \sum_i \|q_i\|^2 + \sum_j \|g_j\|^2 \right) \end{aligned} \tag{3.8}$$

Algorithm 1. The process of CoDetector

Input: User labels U; user-item ratings matrix R, which include attack profiles.

Output: Labels of users to be recognized.

1: Constructing SPPMI matrix M
2: **for** user i in U do
3: for user j in U do
4: Count the number of items both user i and user j consumed.
5: Compute the shifted positive point-wise mutual information.
6: **end for**
7: **end for**
8: **while** notConverged **do**
9: Jointly decompose R and M with shared user latent factors P;
10: update latent vectors.
11: **end while**
12: Use P to predict user labels.

where $\mathbf{p_u}$ is the shared user latent factors which embeds rating and structure information, $\mathbf{m_uj}$ denotes the shifted positive point-wise mutual information [2] between user u and user j, $\mathbf{g_j}$ is the context of user u, and w_u and c_j are the biases of the user and context. The model parameters are updated by using stochastic gradient descent method. The update rules are as follows:

$$\frac{\partial L}{\partial \mathbf{p}_u} = \lambda \mathbf{p}_u - \left(y_{ui} - \mathbf{p}_u^T \mathbf{q}_i\right) \mathbf{q}_i - \left(m_{uj} - \mathbf{p}_u^T \mathbf{g}_j - w_u - c_j\right) \mathbf{g}_j \tag{3.9}$$

$$\frac{\partial L}{\partial \mathbf{q}_i} = \lambda \mathbf{q}_i - \left(y_{ui} - \mathbf{p}_u^T \mathbf{q}_i\right) \mathbf{p}_u \tag{3.10}$$

$$\frac{\partial L}{\partial \mathbf{g}_j} = \lambda \mathbf{g}_j - \left(m_{uj} - \mathbf{p}_u^T \mathbf{g}_j\right) \mathbf{p}_u \tag{3.11}$$

$$\frac{\partial L}{\partial w_u} = m_{uj} - \mathbf{p}_u^T \mathbf{g}_j - w_u - c_j \tag{3.12}$$

$$\frac{\partial L}{\partial c_j} = m_{uj} - \mathbf{p}_u^T \mathbf{g}_j - w_u - c_j \tag{3.13}$$

The constructing SPPMI matrix is time-consuming due to its $O(n^2)$ complexity. This part should be computed off-line. Fortunately, updates for elements in this matrix [2] is not computationally expensive. For each update, only the cells related to this consumption or click need to be modified.

3.7 Evaluation Metrics

To measure the performance of the detection algorithm, we use root mean square error, prediction shift, precision, recall and F1 score.

3.7.1 Root Mean Square Error (RMSE)

Root mean square error computes the mean value of all the differences squared between the true and the predicted ratings and then proceeds to calculate the square root out of the result [30]. As a consequence, large errors may dramatically affect the RMSE rating, rendering the RMSE metric most valuable when significantly large errors are unwanted. The root mean square error between the true ratings and predicted ratings is given by [31]:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(d_i - \hat{d}_i\right)^2} \tag{3.14}$$

d_i is the actual rating.

$\hat{d}_i$ is the predicted rating.

n is the amount of ratings.

3.7.2 Prediction Shift

The predicted shift is a measure of the changes of rating scores in the predicted value before and after the implementation of an attack. Prediction shift is usually used to measure the impact of an attack [1] to recommender systems. $p_{u,i}$ is prediction score of User u on Item i before an attack. $p_{u,i}{}'$ is the prediction score of User u on Item i after attack profiles are injected into the rating matrix.

$\Delta_{u,i}$ is the difference between $p_{u,i}$ and $p_{u,i}{}'$.

$$\Delta_{u,i} = \left|p'_{u,i} - p_{u,i}\right| \tag{3.15}$$

The prediction shift of User u on Item i can be calculated by the following equation:

$$\Delta_i = \sum_{u \in U_T} \frac{\Delta_{u,i}}{|U_T|} \tag{3.16}$$

The prediction shift of all items in the rating matrix after an attack is performed can be calculated by the following equation:

$$\bar{\Delta} = \sum_{u \in I_T} \frac{\Delta_i}{I_T} \tag{3.17}$$

3.7.3 Precision

$$Precision = \frac{TP}{TP + FP} \tag{3.18}$$

Precision reflects the proportion of a category of targets detected by the detector [9] that really belong to that category. Where TP is the number of attack profiles detected correctly, FP is the number of misclassified genuine profiles into attack profiles.

3.7.4 Recall

$$Recall = \frac{TP}{TP + FN} \tag{3.19}$$

Recall reflects the true number of categories [9] in a category of targets detected by the detector. FN is the number of attack profiles misclassified into genuine profiles.

3.7.5 F1 Score

$$F1 = 2.\frac{\text{Precision } \times \text{ Recall}}{\text{Precision } + \text{ Recall}} \tag{3.20}$$

The F1 score is a metric that conveys the balance between precision and recall. It is the harmonic mean of the precision and recall. An F1 score of 1 implies perfect precision and recall, whereas a score of 0 implies precision and recall are not possible:

3.8 Flowchart

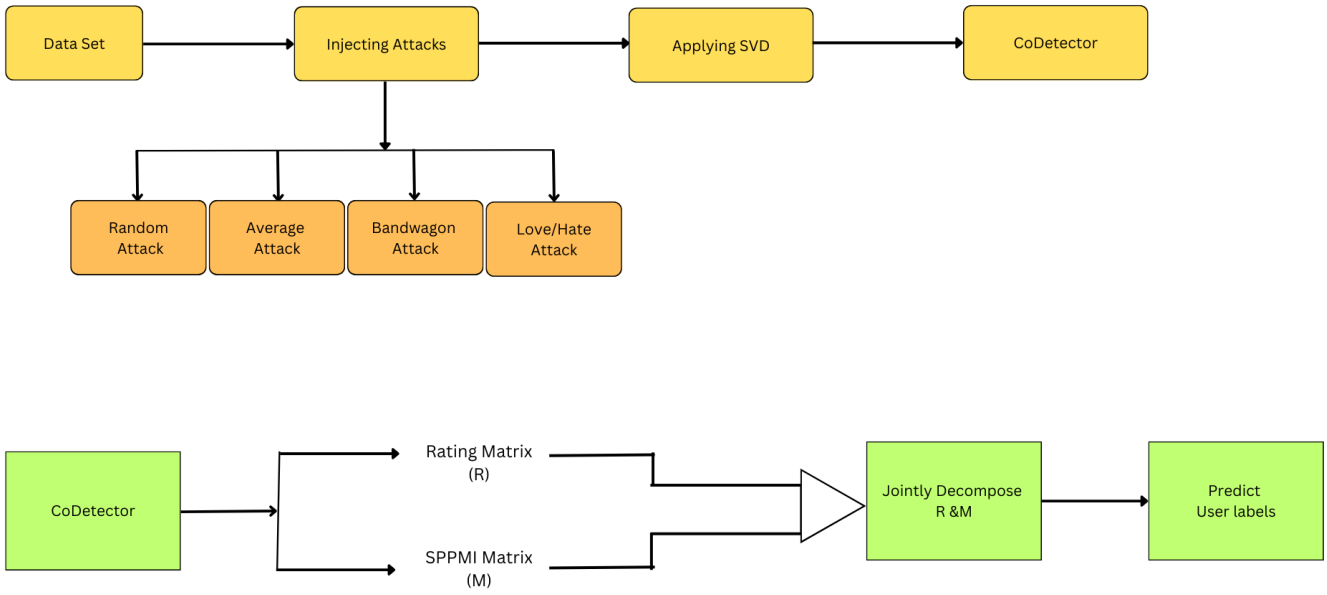

Figure 3.3: Flowchart of implementation of project

This project address the issue of shilling attacks on collaborative filtering recommender systems. To address this issue, we use a collaborative shilling detection model called CoDetector. CoDetector jointly decomposes the user-item interaction matrix and the user-user co-occurrence matrix with shared user latent factors. The user latent factors contain network embedding information, which is then used as features to detect attackers. In this project we present an approach to detecting shilling attacks on collaborative filtering recommender systems, which could improve the accuracy and usefulness of these systems for end-users.

- **Data Set** In order to evaluate the effectiveness of the CoDetector algorithm for detecting shilling attacks on collaborative filtering recommender systems, we conducted experiments using the MovieLens dataset. This dataset contains 100,836 ratings provided by 610 users on 9742 movies.

  We assume dataset from MovieLens without any attacks or fake profiles/users. We then performed different types of shilling attacks on this dataset to see their impact on the recommendation system. After the attacks, we compared the original dataset to the attacked dataset using the root mean square error (RMSE) metric to determine the effectiveness of the attacks. The RMSE measures the differences between

the predicted ratings and the actual ratings.The attacked dataset had errors due to the shilling attacks.A higher RMSE value indicates a larger prediction error, which suggests that the attack has had a more significant impact on the recommendation system's performance.To avoid shilling attacks and improve the error in the dataset, we used a detection method called CoDetector.

In this project, precision, recall, and F1-score were used as performance metrics to evaluate the effectiveness of the different shilling attack detection methods. We found that the CoDetector algorithm achieved the best results in terms of precision, recall, and F1-score, outperforming the other methods tested.

- **Attack Injection** In collaborative filtering (CF) recommender systems, users with similar preferences are likely to choose similar items. Attackers can exploit this open nature of recommender systems by injecting biased profiles to manipulate recommendations. To avoid detection, malicious users use attack models to generate attacker profiles based on their knowledge of the recommender systems.

  The general profile of an attacker can be divided into four segments: target item set (I_T), selected items based on specific needs of the spam user (I_S), filler item set (I_F) used to disguise attackers, and unrated item set (I_N) which forms the majority of the profile and is always empty. The target item set represents the items that attackers want to promote (push attack) or demote (nuke attack) in the recommendations.

  The selected items in the attacker's profile are chosen based on the attacker's specific needs and preferences. The filler item set is used to hide the attacker's malicious behavior by including items that are not related to the target item set or the attacker's preferences. The unrated item set is empty because attackers do not rate items in this set, as they do not want to provide information that could reveal their malicious behavior.

  Overall, attackers use these profile segments to manipulate the recommender system and promote or demote certain items in the recommendations while avoiding detection. Detecting and preventing these shilling attacks is an important area of research in the field of recommender systems.

  Random attack : Filler items are assigned random values.

Average attack : Filler items are assigned the corresponding average ratings of items.

Bandwagon attack : Selected items are the frequently rated items and assigned the maximum rating.

Love/Hate Attack : The attacker selects filler objects at random and assigns the highest rating to those items while assigning the lowest rating to the largest item.

Then we find RMSE and Prediction shift , A higher RMSE value indicates a larger prediction error, which suggests that the attack has had a more significant impact on the recommendation system's performance.Prediction Shift is used to evaluate the stability of the algorithm under a shilling attack. It denotes the difference in the system's predicted rating for an item before and after the attack.

- **Detection Algorithm** Collaborative shilling detection model, CoDetector, which combines matrix factorization (MF) and user embedding techniques to capture the implicit interactions between users and items. Matrix factorization is a popular technique used in recommender systems to decompose the user-item interaction matrix into low-rank matrices and extract latent features. User embedding, on the other hand, represents users as low-dimensional vectors in a continuous space, which capture the user's preferences and behavior. By jointly decomposing the user-item interaction matrix and the user-user co-occurrence matrix with shared user latent factors, CoDetector can capture the fine-grained interactions between users and items and detect shilling attacks effectively.

  Matrix Factorization Matrix Factorization (MF) is a collaborative filtering method that aims to uncover the underlying latent features that influence the interactions between users and items. It does this by mapping both users and items into a low-dimensional latent-factor space. The objective function of MF includes a term that measures the difference between the predicted and observed ratings, and a regularization term that controls the magnitudes of the latent factors.

  Word Embedding Word embedding, on the other hand, is a set of models used in natural language processing to represent words in a continuous vector space. SGNS, or skip-gram neural embedding model, is a popular neural model for training word embeddings using the negative-sampling procedure. SGNS is equivalent to

factorizing a word-context matrix whose cells are the pointwise mutual information (PMI) of the respective word and context pairs. PMI is an information-theoretic measure that can be estimated empirically.

User Embedding User embedding is a technique used to represent users in a high-dimensional latent space. In the context of CoDetector, the user embedding is constructed by factorizing the user-user co-occurrence SPPMI matrix, which is computed by counting the number of items that both user i and user **j** consumed. The SPPMI matrix is used to capture the structural information of the user-item bipartite graph, where the context of a user is defined as other users who consumed or rated the same items as the user.

By factorizing the SPPMI matrix, CoDetector is able to embed the structural information into user latent factors, which can be used to identify anomalies caused by attackers who manipulate recommendation results by injecting biased user profiles in a large scale. Attackers tend to promote or demote target items in groups, and factorizing the SPPMI matrix can reveal the implicit interactions among attackers in the user-item bipartite graph.

In addition, by tuning the value of negative samples, CoDetector can neglect noises in the bipartite while preserving available connections. This allows the model to focus on the relevant information and improve the accuracy of anomaly detection.

Training Procedure Training procedure involves jointly decomposing the rating matrix R and the SPPMI matrix M with shared user latent factors, where p_u is the shared user latent factors that embeds rating and structure information, muj denotes the shifted positive poin-twise mutual information between user u and user j, g_j is the context of user u, and w_j and c_j are the biases of the user and context. The model parameters are updated using stochastic gradient descent method.

To reduce the time complexity of constructing the SPPMI matrix, it is computed offline. During updates, only the cells related to the consumption or click need to be modified, which is not computationally expensive. This approach allows for the fusion of both rating and structural information into user latent factors, which helps detect anomalies introduced by attackers who manipulate recommendation results by injecting biased user profiles.

- **Performance measure** To evaluate the performance of the proposed algorithm, we conduct project on MovieLens datasets. Here we show the results of CoDetector in comparison with four state-of-the-art shilling attack detectors and analyze the effect of model parameters.

  In this project , precision, recall, and F1-score were used as performance metrics to evaluate the effectiveness of the different shilling attack detection methods.

  RMSE RMSE is a commonly used metric to evaluate the accuracy of recommender systems. In the context of shilling attacks, it can be used to compare the prediction accuracy of the original (unattacked) dataset with the attacked dataset. A higher RMSE value indicates a larger difference in prediction accuracy between the two datasets, which suggests that the attack has had a greater impact on the recommendation system's performance.

  Precision precision is used to measures the accuracy of positive predictions. reflects the proportion of a category of targets detected by the detector that really belong to that category.

  Recall Recall is evaluation matrix which is used to measures the completeness of positive predictions.

  F1 score F1 score measures a model's accuracy. It combines the precision and recall scores of a model. The accuracy metric computes how many times a model made a correct prediction across the entire dataset.

  Prediction shift Prediction Shift is used to evaluate the stability of the algorithm under a shilling attack. It denotes the difference in the system's predicted rating for an item before and after the attack.

Chapter 4

Implementation

4.1 Dataset Description

The dataset used in this project is Movielens. This dataset contains a set of movie ratings from the MovieLens website, a movie recommendation service. This dataset was collected and maintained by GroupLens, a research group at the University of Minnesota.

This dataset is ml-latest-small describes 5-star rating and free-text tagging activity from MovieLens, a movie recommendation service. It contains 100836 ratings and 3683 tag applications across 9742 movies. These data were created by 610 users between March 29, 1996 and September 24, 2018. This dataset was generated on September 26, 2018.

Users were selected at random for inclusion. All selected users had rated at least 20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided. The data are contained in the files links.csv, movies.csv, ratings.csv and tags.csv.

4.2 Software Requirements

- **Google Collaboratory:** Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook.

Features of Google Collaboratory :

- Write and execute code in Python
- Document the code which supports the mathematical equations
- Create new notebooks
- Upload the existing notebooks
- Share the notebooks with the google link
- Import data from Google Drive
- Save notebooks from/to Google Drive
- Import/Publish notebooks from GitHub
- Import external datasets e.g. from Movielens
- Integrate PyTorch, TensorFlow, Keras, OpenCV
- Free Cloud service with free GPU and TPU.
- **Jupyter Notebook** The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter. Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

Features of Jupyter Notebook:

- Run Commands: To run commands in cells, we can simply prefix an exclamation mark before the command.
- Check Active Variables: When we work with lots of data, we may have created many intermediate variables in our workspace. To get to know the current pool of these variables, we can use the magic method %whoor %whos, as shown below, with the latter showing more details of the variables.

- API Lookups: We don't always remember the functions or attributes that we want to use. However, we do have some impressions about them, after all, we may have used them from time to time before. In this case, we can list all related methods that may remind us.

- Change the Default Output Mode: By default, each cell only prints the last expression after executing the entire cell. However, chances are that we want to display more items, such as all the expressions in the cell. In this case, we can change the default setting that allows the output of multiple items.

4.3 Experimental Results

To evaluate the performance of the proposed algorithm, we conduct experiments on MovieLens dataset. Here we show the results of CoDetector and analyse the model parameters. Table 4.1 and 4.2 displays the findings of CoDetector on the MovieLens dataset using the evaluation metrics precision, recall and F1. We firstly evaluate the performance of CoDetector on four different attack models: Random attack, Average attack, and Bandwagon attack and Love/Hate attack. In MovieLens, we assume that the original users are normal user, and inject simulated attackers manually according to the definition of attack models. So according to how well these strategies perform, we record the best parameters. With attack size 20% and filler size 10% we achieve the best performance on the bandwagon attack, with precision 0.9986, recall 0.996 and F1 0.992.

Table 4.1: Detection results on four typical shilling attacks on MovieLens

Attack Models	Attack Size	Filter Size	10%	20%	30%	40%
Random Attack	5%	Precision	0.9406	0.9328	0.9375	0.9375
		Recall	0.6333	0.6206	0.6083	0.4762
		F1	0.5727	0.6003	0.5552	0.4918
	10%	Precision	0.8689	0.8733	0.8733	0.8793
		Recall	0.5987	0.6069	0.615	0.6033
		F1	0.588	0.5967	0.6212	0.5725
	15%	Precision	0.8545	0.8217	0.8145	0.8174
		Recall	0.6696	0.5893	0.5758	0.5817
		F1	0.6541	0.5849	0.5708	0.5771
	20%	Precision	0.7855	0.7786	0.7555	0.8074
		Recall	0.6082	0.5894	0.5832	0.6499
		F1	0.6049	0.585	0.5913	0.6486
Average Attack	5%	Precision	0.9313	0.9203	0.9141	0.9297
		Recall	0.5573	0.5814	0.5411	0.6148
		F1	0.5519	0.5937	0.5429	0.6145
	10%	Precision	0.8793	0.8719	0.8823	0.8749
		Recall	0.6356	0.6312	0.6546	0.6406
		F1	0.6232	0.615	0.6787	0.6527
	15%	Precision	0.883	0.8359	0.846	0.8245
		Recall	0.7338	0.6361	0.6594	0.5981
		F1	0.7173	0.6343	0.6551	0.6094
	20%	Precision	0.832	0.8348	0.8292	0.8142
		Recall	0.704	0.7035	0.7028	0.6634
		F1	0.7286	0.7048	0.7305	0.6694

Table 4.2: Detection results on four typical shilling attacks on MovieLens

Attack Models	Attack Size	Filter Size	10%	20%	30%	40%
Bandwagon Attack	5%	Precision	0.9969	0.9984	0.9953	0.9984
		Recall	0.9714	0.9857	0.9841	0.90857
		F1	0.9984	0.9992	0.9659	0.9992
	10%	Precision	0.994	0.997	0.9955	0.9955
		Recall	0.983	0.9915	0.9838	0.9831
		F1	0.9817	0.9908	0.9907	0.99
	15%	Precision	0.99720	0.9972	0.9957	0.9957
		Recall	0.9895	0.9939	0.9884	0.9931
		F1	0.9984	0.9939	0.9928	0.9884
	20%	Precision	0.9986	0.9986	0.998	0.9973
		Recall	0.996	0.996	0.9896	0.9952
		F1	0.9992	0.9992	0.9992	0.995
Love Hate Attack	5%	Precision	0.9735	0.9672	0.9657	0.9656
		Recall	0.8565	0.8226	0.8142	0.8256
		F1	0.891	0.8401	0.8552	0.8235
	10%	Precision	0.9717	0.9732	0.9746	0.9657
		Recall	0.9191	0.9133	0.9358	0.8977
		F1	0.9106	0.9346	0.9122	0.8991
	15%	Precision	0.97	0.9715	0.9743	0.9743
		Recall	0.9387	0.9474	0.9415	0.9413
		F1	0.936	0.9276	0.9476	0.9474
	20%	Precision	0.9795	0.9713	0.9741	0.9699
		Recall	0.9644	0.9539	0.9521	0.9476
		F1	0.9649	0.9431	0.9585	0.9458

4.4 Graphs

To test the impact of shilling attacks on recommender system, we calculated RMSE values with different types of shilling attacks injected into the rating matrix. In this, the MovieLens ML100K dataset was used to calculate the RMSE values when different attack profiles are injected while the attack sizes and filler sizes vary. For this we inject some attack profiles (the same number of profiles of Random Attack, Average Attack,Bandwagon Attack and Love/Hate Attack) into the data set.

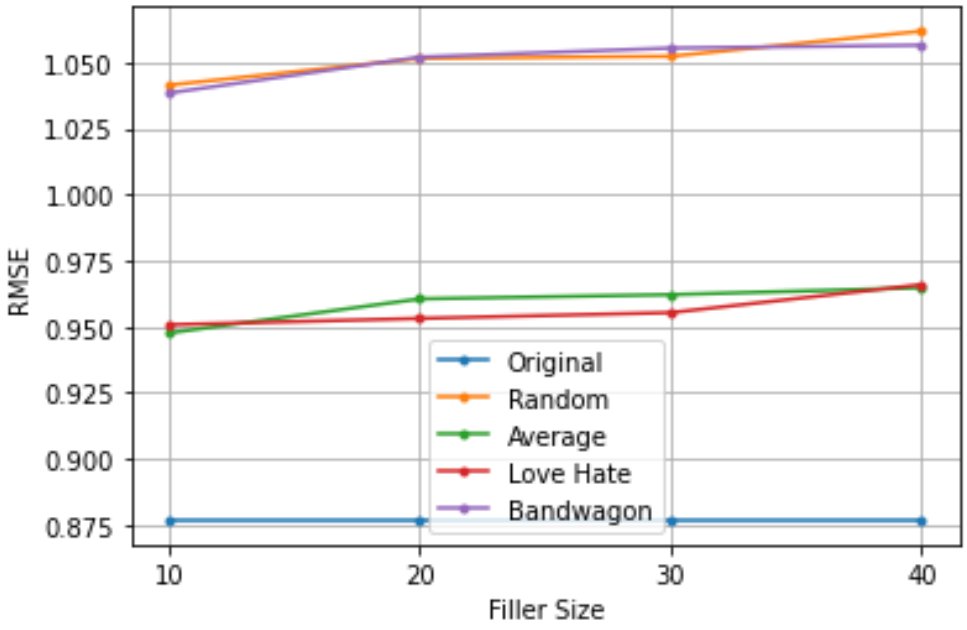

Figure 4.1: RMSE vs Attack size

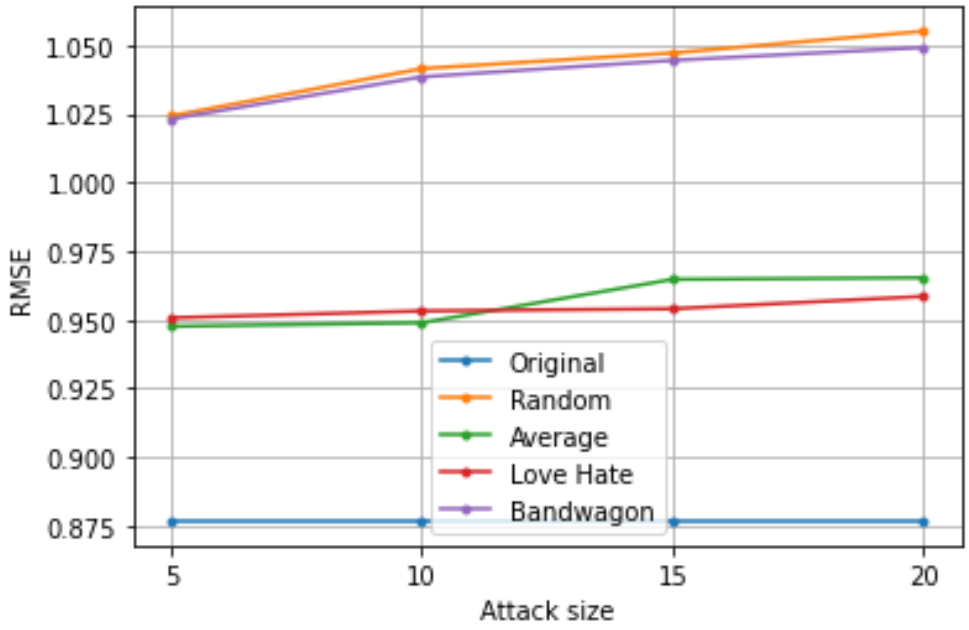

Figure 4.2: RMSE vs Attack size

In 4.1, Y-axis represents RMSE values and X-axis represents filler sizes.The filer size is taken as 10% , 20%, 30% and 40% and the target count is kept at 10. The original RMSE value was found out to be 0.875. After injecting attack profiles it was found that as the filler size increases, RMSE value increases. RMSE value for Random attack, Average attack, Bandwagon attack and Love/Hate attack was found out to be 1.055, 0.960, 1.052 and 0.960 respectively. Thus Random attack and Bandwagon attack performs in a more effective as well as in a impactful way in the system.

In 4.2, Y-axis represents RMSE values and X-axis represents attack sizes. Attack size is taken as 5%, 10%, 15% and 20% and the target count is kept at 5. The orginal RMSE vale before injecting attack profiles was found out to be 0.875. After injecting attack profiles it was found that as the attack size increases, RMSE value increases. RMSE value for Random attack, Average attack, Bandwagon attack and Love/Hate attack was found out to be 1.052, 0.960, 1.050 and 0.955 respectively. Thus Random attack and Bandwagon attack performs in a more effective as well as in a impactful way in the system.

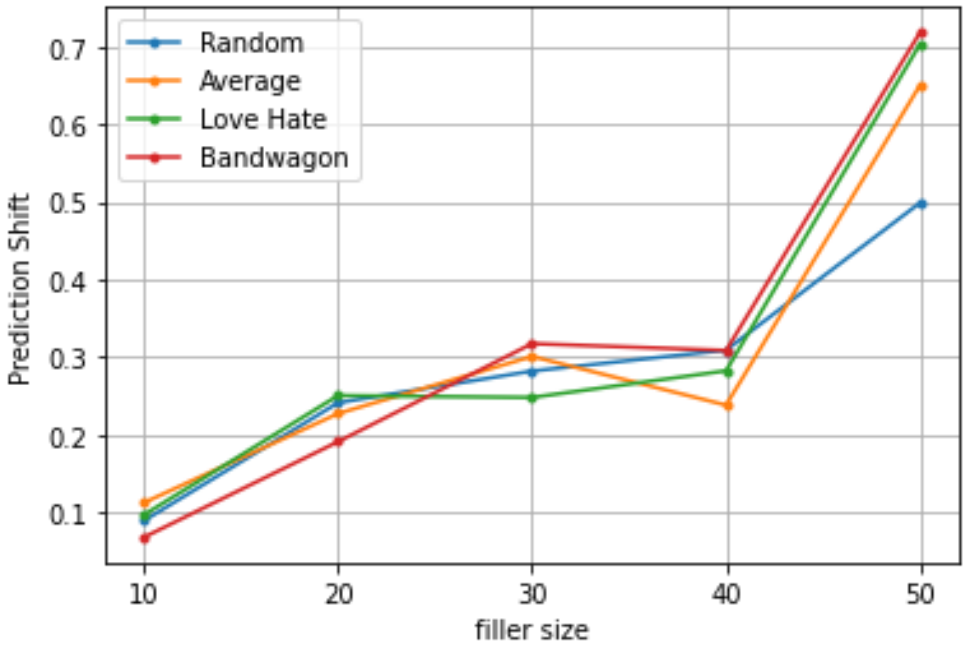

Figure 4.3: Prediction shift vs Filler size

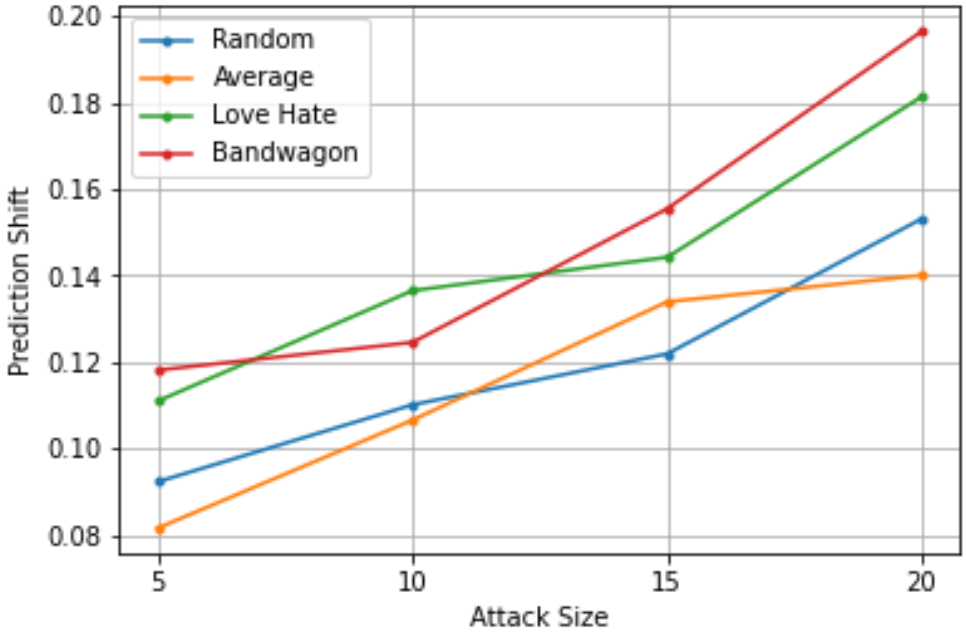

Figure 4.4: Prediction shift vs attack size

To test the impact of shilling attacks on recommender system, we also calculated prediction shift values when different types of shilling attacks are injected into the rating matrix. In this, the MovieLens ML100K dataset was used to calculate the prediction shift values when different attack profiles are injected while the attack sizes and filler sizes vary. For this we inject some attack profiles (the same number of profiles of Random Attack, Average Attack,Bandwagon Attack and Love/Hate Attack) into the data set.

From 4.3 and 4.4, we can see that the value of prediction shift increases when attack size increases. When the filler size increases, the value of prediction shift also increases. We can infer from this that the value of prediction shift increases with filler size and attack size increase.

Chapter 5

Conclusion

In this project we first built a recommender system, then we implemented different types of shilling attack namely Random attack, Average attack, Bandwagon attack and Love/Hate attack. We then defined the various detection attributes which are widely used in detection techniques. We also briefly address the impact of shilling attacks on implicit rating systems. We used a collaborative shilling detection model called CoDetector bridging factorization and user embedding. It integrates rating and structure information into shared user latent factors to recognize shilling attackers in recommender systems. Our results show that CoDetector model significantly outperforms state-of-the-art methods.

Bibliography

[1] Wei Zhou, Junhao Wen, Qiang Qu, Jun Zeng and Tian Cheng, "Shilling attack detection for recommender systems based on credibility of group users and rating time series", *PloS one, vol.13(5)*, Public Library of Science San Francisco, CA USA, pp.e0196533, 2018.

[2] Tong Dou, Junliang Yu, Qingyu Xiong, Min Gao, Yuqi Song, and Qianqi Fang, "Collaborative shilling detection bridging factorization and user embedding", *Collaborative Computing : Networking, Applications and Worksharing : 13th International Conference, CollaborateCom 2017*, Edinburg, UK, pp. 459-469, 2018.

[3] Z.A.Wu, Y.Q.Wang, J.Cao, "A survey on shilling attack models and detection techniques for recommender systems", *Chin. Sci. Bull. 59(7)*, pp.551-560, 2014.

[4] Robin Burke, Bamshad Mobasher, Chad Williams, and Runa Bhaumik, "Classification features for attack detection in collaborative recommender systems", *"Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining"*, pp.542-547, 2006.

[5] Wentao Li, Min Gao, Hua Li, Jun Zeng, Qingyu Xiong and Sachio Hirokawa, "Shilling attack detection in recommender systems via selecting patterns analysis", *IEICE TRANSACTIONS on Information and Systems, vol.99(10)*, pp.2600-2611, 2016.

[6] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S.Corrado, and Jeff Dean, "Distributed representations of words and phrases and their compositionality", *Advances in neural information processing systems, vol(26)*, 2013.

[7] Omer Levy and Yoav Goldberg, "Neural word embedding as implicit matrix factorization", *Advances in neural information processing systems, vol.(27)*, 2014.

[8] Meng Jiang, Peng Cui and Christos Faloutsos, "Suspicious behavior detection: Current trends and future directions", in *IEEE Intelligent Systems, vol. 31(1)*, pp. 31-39, Jan-Feb 2016.

[9] Xin Liu, Yingyuan Xiao, Xu Jiao, Wenguang Zheng and Zihao Ling, "A novel Kalman Filter based shilling attack detection algorithm", *arXiv preprint arXiv:1908.06968*, 2019.

[10] G. M. Dakhel and M. Mahdavi, "A new collaborative filtering algorithm using K-means clustering and neighbors' voting," *2011 11th International Conference on Hybrid Intelligent Systems (HIS)*, Melacca, Malaysia, pp. 179-184, 2011.

[11] Gaofeng Cao, Huan Zhang, Yuyou Fan and Li Kuang, "Finding Shilling Attack in Recommender System based on Dynamic Feature Selection", *SEKE*, pp.50-55, 2018.

[12] P. Bedi and S. K. Agarwal, "Managing Security in Aspect Oriented Recommender System," *2011 International Conference on Communication Systems and Network Technologies*, Katra, India, pp. 709-713, 2011.

[13] Paul-Alexandru Chirita, Wolfgang Nejdl, and Cristian Zamfir, "Preventing shilling attacks in online recommender systems", *Proceedings of the 7th annual ACM international workshop on Web information and data management*,New York, pp. 67-74, Oct 2005.

[14] Jie Cao, Zhiang Wu, Bo Mao and Yanchun Zhang, "Shilling attack detection utilizing semi-supervised learning method for collaborative recommender system", *World Wide Web, vol.16*, Springer, pp. 729-748, 2013.

[15] Xiao-Li Li, Philip S.Yu, Bing Liu, and See-Kiong Ng, "Positive unlabeled learning for data stream classification", *Proceedings of the 2009 SIAM international conference on data mining*, SIAM, pp. 259-270, 2009.

[16] Zhiang Wu, Youquan Wang, Yaqiong Wang, Junjie Wu, Jie Cao, Jie and Lu Zhang, "Spammers detection from product reviews: a hybrid model, *2015 IEEE International Conference on Data Mining*, Atlantic City, NJ, USA, pp. 1039-1044, 2015.

[17] Z. Zhang and S. R. Kulkarni, "Graph-based detection of shilling attacks in recommender systems," *2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, Southampton, UK, pp. 1-6, 2013.

[18] Bhaskar Mehta, and Wolfgang Nejdl, "Unsupervised strategies for shilling detection and robust collaborative filtering", *User Modeling and User-Adapted Interaction, vol.19*, Springer, pp. 65-97, 2009.

[19] P. Kaur and S. Goel, "Shilling attack models in recommender system", *2016 International Conference on Inventive Computation Technologies (ICICT)*, Coimbatore, India, pp. 1-5, 2016.

[20] W. Zhou, J. Wen, Y. S. Koh, S. Alam and G. Dobbie, "Attack detection in recommender systems based on target item analysis", *2014 International Joint Conference on Neural Networks (IJCNN)*, Beijing, China, pp. 332-339, 2014.

[21] A. P. Sundar, F. Li, X. Zou, T. Gao and E. D. Russomanno, "Understanding Shilling Attacks and Their Detection Traits: A Comprehensive Survey," in *IEEE Access, vol.8*, pp. 171703-171715, 2020.

[22] Chen Li and Cheng Yang, "The research based on the Matrix Factorization recommendation algorithms," *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Xi'an, China, pp. 691-698, 2016.

[23] Y. Koren, R. Bell and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," in *Computer, vol.42(8)*, pp. 30-37, Aug 2009.

[24] Badrul Sarwar, George Karypis, Joseph Konstan and John Riedl, "Item-based collaborative filtering recommendation algorithms", *Proceedings of the 10th international conference on World Wide Web*, pp. 285-295, 2001.

[25] Dheeraj kumar Bokde, Sheetal Girase, and Debajyoti Mukhopadhyay, "Role of matrix factorization model in collaborative filtering algorithm: A survey", *CoRR, abs/1503.07475*, 2015.

[26] Badrul Sarwar, George Karypis, Joseph Konstan and John Riedl, "Application of dimensionality reduction in recommender system-a case study", *Proc. ACM WebKDD Workshop*, pp. 1-12, 2000.

[27] R. Mehta and K. Rana, "A review on matrix factorization techniques in recommender systems," *2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*, Mumbai, India, pp. 269-274, 2017.

[28] Francesco Ricci, Lior Rokach, Bracha Shapira and Paul B. Kantor, "Recommender Systems Handbook", *Springer*, 2011.

[29] Huseyin Polat and Wenliang Du, "SVD-based collaborative filtering with privacy", *Proceedings of the 2005 ACM symposium on Applied computing*, pp. 791-795, 2005.

[30] Tianfeng Chai and Roland R.Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)?-Arguments against avoiding RMSE in the literature", *Geoscientific model development, vol,7(3)*, Copernicus Publications Göttingen, Germany, pp. 1247-1250, 2014.

[31] Jonathan L.Herlocker, Joseph A.Konstan, Loren G.Terveen and John T.Riedl, "Evaluating collaborative filtering recommender systems", *ACM Transactions on Information Systems (TOIS), vol.22(1)*, ACM New York, NY, USA, pp. 5-53, 2004.

[32] M. P. O'Mahony, N. Hurley, and G. C. M. Silvestre, "Promoting recommendations: An attack on collaborative filtering", In *DEXA '02: Proceedings of the 13th International Conference on Database and Expert Systems Applications*, Springer-Verlag, London, UK, pages 494–503, 2002.

[33] R. Z. Robin Burke, Bamshad Mobasher and R. Bhaumik, "Identifying attack models for secure recommendation", In *Beyond Personalization*, 2005.

[34] S. K. Lam and J. Riedl, "Shilling recommender systems for fun and profit", In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, ACM Press, New York, NY, USA, pp. 393–402, 2004.

[35] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre, "Collaborative recommendation: A robustness analysis", *ACM Transactions on Internet Technology (TOIT), vol.4(4)*, ACM New York, NY, USA, pp.344-377, 2004.

[36] H.J. Z. Xue-Feng Su and Z. Chen, "Finding group shilling in recommendation system", In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, 2005.

[37] B. N. Miller, J. A. Konstan, and J. Riedl, "Pocketlens: Toward a personal recommender system",*ACM Transactions on Information Systems (TOIS), vol.22(3)*, ACM New York, NY, USA, pp. 437-476, 2004.

[38] H. Polat and Wenliang Du, "Privacy-preserving collaborative filtering using randomized perturbation techniques", *Third IEEE International Conference on Data Mining*, Melbourne, FL, USA, pp. 625-628, 2003.

[39] J. Canny, "Collaborative filtering with privacy", *Proceedings 2002 IEEE Symposium on Security and Privacy*, Berkeley, CA, USA, pp. 45-57, 2002.

[40] Chih-Chao Ma, "A guide to singular value decomposition for collaborative filtering", *Computer (Long Beach, CA)*, pp. 1-14, 2008.

[41] B. Mehta and T. Hofmann, "A survey of attack-resistant collaborative filtering algorithms", *IEEE Data Eng. Bull., vol. 31(2)*, pp. 14–22, Jun. 2008.

[42] F. Zhang, "A survey of shilling attacks in collaborative filtering recommender systems," in *Proc. Int. Conf. Comput. Intell. Softw. Eng.*, pp. 1–4, Dec. 2009.

[43] I. Gunes, C. Kaleli, A. Bilge, and H. Polat, "Shilling attacks against recommender systems: A comprehensive survey", *Artif. Intell. Rev., vol. 42(4)*, pp. 767–799, Dec. 2014.

[44] K. Patel, A. Thakkar, C. Shah, and K. Makvana, "A state of art survey on shilling attack in collaborative filtering based recommendation system", in *Proc. 1st Int. Conf. Inf. Commun. Technol. Intell. Syst., vol. 1*, Cham, Switzerland: Springer, pp. 377–385, 2016.

[45] R. Burke, M. P. O'Mahony, and N. J. Hurley, "Robust collaborative recommendation," in *Recommender Systems Handbook*, Boston, MA, USA: Springer, pp. 961–995, 2015.

[46] M. Si and Q. Li, "Shilling attacks against collaborative recommender systems: A review", *Artif. Intell. Rev., vol. 53(1)*, pp. 291–319, Jan. 2020.

[47] R. A. Zayed, L. F. Ibrahim, H. A. Hefny, and H. A. Salman, "Shilling attacks detection in *collaborative recommender system: Challenges and promise," in Proc. Workshops Int. Conf. Adv. Inf. Netw. Appl*,Cham, Switzerland: Springer, pp. 429–439, 2020.

[48] M.E.K.Badran,W.Jurdi, and J. B. Abdo, "Survey on shilling attacks and their detection algorithms in recommender systems," in *Proc. Int. Conf. Secur. Manage. (SAM) Steering Committee World Congr. Comput. Sci., Comput*, pp. 141–146, 2019.

[49] G. Linden, B. Smith and J. York, "Amazon.com recommendations: item-to-item collaborative filtering", *IEEE Internet Computing, vol. 7(1)*, pp. 76-80, 2003.

[50] B. Mobasher, R. Burke, R. Bhaumik and C. Williams, "Effective Attack Models for Shilling Item-Based Collaborative Filtering Systems", In *Proc. of the 2005 WebKDD Workshop*, Chicago, Illinois, 2005.

[51] M. Mahony, N. Hurley and C. Silvestre, "Recommender Systems: Attack Types and Strategies", *American Association for Artificial Intelligence*, pp. 334-339, 2005.

[52] B. Mobasher, R. Burke, R. Bhaumik and J. Sandvig, "Attacks and Remedies in Collaborative Recommendation", *IEEE Intell. Syst., vol.22(3)*, pp. 56-63, 2007.

[53] R. Burke, B. Mobasher, R. Bhaumik and C. Williams, "Segment-based injection attacks against collaborative filtering recommender systems", In *Data Mining, Fifth IEEE International Conference*, 2005.

[54] F. Zhang, "Average Shilling Attack against Trust-Based Recommender Systems", *2009 International Conference on Information Management, Innovation Management and Industrial Engineering*, pp. 588-591, 2009.

[55] F. Zhang, "Analysis of Bandwagon and Average Hybrid Attack Model against Trust-based Recommender Systems", *Fifth International Conference on management of e-Commerce and e-Government*, pp. 269-273, 2011.

[56] J. O'Donovan and B. Smyth, "Trust in Recommender Systems", *IUI, Association for Computing Machinery*, New York, NY, USA, 2005.

[57] P. Massa and P. Avesani, "Trust-aware recommender systems", in Proceedings of the 1st ACM Conference on Recommender Systems (RecSys '07), pp. 17-24, 2007.

[58] R. Burke, B. Mobasher, e. Williams and R. Bhaumik, " Classification features for attack detection in collaborative recommender systems", In: *Proceedings of the 12th international conference on knowledge discovery and data mining*, pp. 542-547, 2006.

[59] B. Mobasher, R. Burke, R. Bhaumik and e. Williams, "Toward trustworthy recommender systems: an analysis of attack models and algorithm robustness", *ACM Transactions on Internet Technology, vol. 7, no A, Article 23*, pp. 1-38, Oct. 2007.

[60] J. A. Calandrino, A. Kilzer, A. Narayanan, E. W. Felten, and Y.Shmatikov, "You might also like: privacy risks of collaborative filtering", *Proc. IEEE Symposium on Security and Privacy*, pp. 231-246, May 2011.

[61] N.Hurley, Z.Cheng, M.Zhang, "Statistical attack detection", In: *Proceedings of the 3rd ACM international conference on recommender systems*, New York, NY, USA, pp. 149-156, 2009.

[62] C.Li, Z.Luo, "Detection of shilling attacks in collaborative filtering recommender systems", In: *Proceedings of the international conference of soft computing and pattern recognition*, pp. 190-193, 2011.

[63] B.Mehta, "Unsupervised shilling detection for collaborative filtering", In: *Proceedings of the 22nd international conference on artificial intelligence*, pp. 1402-1407, 2007.

[64] R.Bhaumik, B.Mobasher, R.Burke R, "A clustering approach to unsupervised attack detection in collaborative recommender systems", In: *Proceedings of the 7th IEEE international conference on data mining*, pp. 181-187, 2011.

[65] 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, 2018.

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

[67] L. Zhang, Z. Wu, and J. Cao, "Detecting spammer groups from product reviews: A partially supervised learning model", *IEEE Access, vol. PP, no. 99*, pp. 1–1, 2017.

[68] Z. Wu, J. Wu, J. Cao, and D. Tao, "Hysad:a semi-supervised hybrid shilling attack detector for trustworthy product recommendation," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 985–993, 2012.

[69] F. Zhang, "Analysis of Love-Hate Shilling Attack Against E-commerce Recommender System", *2010 International Conference of Information Science and Management Engineering*, Shaanxi, China, pp. 318-321, 2010.

[70] C.Williams,B.Mobasher,R.Burke,J.Sandvig,andR.Bhaumik,"Detection of obfuscated attacks in collaborative recommender systems", in *Proc. Workshop Recommender Syst.(ECAI),vol.94*, pp.19–23, 2006.

Appendix

Timeline Chart

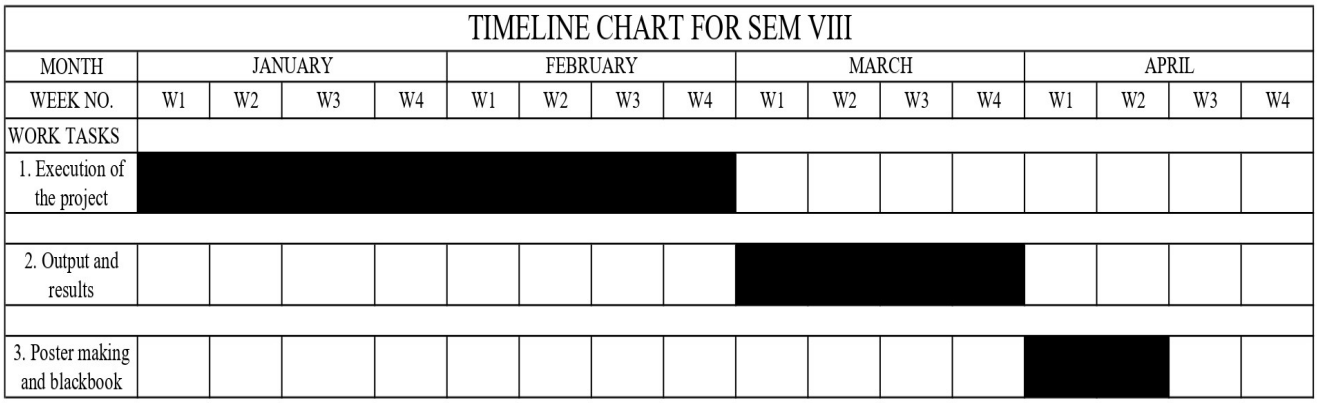

Figure 5.1: Timeline of project for Sem VIII