공학박사학위논문

Design & Analysis on Robust Defense Algorithm against Sybil Attack Using Information Level

정보 수준을 이용한 강건한 시빌공격 방어 알고리즘 설계 및 분석

2014년 8월

서울대학교 대학원

전기·컴퓨터공학부

노 기 섭

Design & Analysis on Robust Defense Algorithm against Sybil Attack Using Information Level

정보 수준을 이용한 강건한 시빌공격 방어 알고리즘 설계 및 분석

지도교수 김 종 권

이 논문을 공학박사학위논문으로 제출함

2014 년 8 월

서울대학교 대학원

전기·컴퓨터 공학부

노 기 섭

노 기 섭의 박사학위논문을 인준함

2014 년 8 월

위 원 장 _____ 전화숙

부위원장 _____ 김종권

위 원 _____ 박세웅

위 원 _____ 권태경

위 원 _____ 오하영

Abstract

Design & Analysis on Robust Defense Algorithm against Sybil Attack Using Information Level

Giseop Noh

School of Electrical Engineering and Computer Science

The Graduate School

Seoul National University

As the major function of Recommender Systems (RSs) is recommending commercial items to potential consumers (i.e., system users), providing correct information of RS is crucial to both RS providers and system users. The influence of RS over Online Social Networks (OSNs) is expanding rapidly, whereas malicious users continuously try to attack the RSs with fake identities (i.e., *Sybils*) by manipulating the information in the RS adversely. In this thesis, we propose a novel robust recommendation algorithm called RobuRec which exploits a distinctive feature, admission control. RobuRec provides highly Trusted recommendation results since RobuRec predicts appropriate recommendations regardless of whether the ratings are given by honest users or by Sybils thanks to the power of admission control. To demonstrate the performance of RobuRec, we have conducted extensive exper-

iments with various datasets as well as diverse attack scenarios. The evaluation results confirm that RobuRec outperforms the comparable schemes such as Principal Component Analysis (PCA) and Least Trimmed Squared Matrix Factorization (LTSMF) significantly in terms of Prediction Shift (PS) and Hit Ratio (HR).

Keywords: Sybil Attack, Recommendation Systems, Robust Algorithm, Information Level

Student Number: 2011-30791

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

As *Online Social Network services* (OSNs) such as Facebook, YouTube and LinkedIn are getting popular, the Recommender Systems (RSs) are also spreading over OSNs. The major function of RS is recommending commercial items to potential consumers (i.e., system users). For example, RSs provide goods to a user with customized information, or propose locations where the users might want to visit. Providing correct information of RS is crucial to both service providers and system users: (i) From the viewpoints of OSN providers, maintaining and supplying correct information of RS makes their profit increase, enlarges the market influence of newly launched items as well as entices new comers to their company. (ii) The customers save time and unnecessary efforts to search items in mind through conveniently acquiring custom-tailored information from the RS. For these reasons, the RSs keep being proliferated over the Internet world (mainly, OSNs) and the importance of RSs also being magnified.

Since OSNs encourage unregistered potential users to become a member of their systems as many as possible to increase the profit factors such as page views and click counts, unlimited number of dishonest or malicious users (i.e., attackers) could enter the OSNs without rigid qualifications. The attacker creates fake identities (i.e., *Sybils*) and injects them into the target RS. Each injected Sybil starts generating fake rating values (i.e., Sybil's profile) on the target items to distort the outcome of the RS. The objective of the attackers is overwhelming the activities of honest users or making biased outcomes by suppressing normal user's opinions. Such an attack using malicious identities is widely known as Sybil attack [1]. The Sybil attacks are mainly focused on peer-to-peer systems at the beginning, they continue to evolve and are able to severely distort the results of recommender or ranking systems in the OSNs. The RSs are originally designed for providing customized information under the assumption that the most of system users are honest and highly reliable [1, 2]. However, H. Yu *et al.* showed that the RSs are proven to be vulnerable to the Sybil attacks (e.g., only 0.1% or 1% of Sybils out of the honest users is enough to compromise a RS) [3]. Unfortunately, the Sybil attacks keep being reported from commercial sectors repeatedly[1][2][3][4].

In essence, the most important task of robust RSs is minimizing the distortion of the RS information and maintaining the integrity of recommendations to the honest users. To address these challenges, several approaches have been pro-

[1] http://usatoday30.usatoday.com/money/industries/retail/story/2011-11-01/deceptive-product-review-sites/51033028/1
[2] http://nytimes.com/2012/01/27/technology/for-2-a-star-a-retailer-gets-5-star-reviews.html
[3] http://business.time.com/2012/02/06/how-computer-geeks-aim-to-put-a-stop-to-fake-online-reviews
[4] http://bits.blogs.nytimes.com/2013/04/05/fake-twitter-followers-becomes-multimillion-dollar-business

posed [4–7]. All these approaches, however, have a fundamental limitation in the sense that the recommendation predictions are performed with a given RS matrix containing a number of honest ratings as well as Sybils' profiles after their Sybil defense algorithms conducted. In other words, these algorithms start to predict recommendations with unblocked Sybils, causing a delivery of unreliable recommendation results. The faulty results from information systems such as RSs may read the Internet users make their decisions falsely. The threat of the Sybil attacks existing with any suggesting or recommending systems in real world.

1.2 Goal and Contribution

The main goal of this thesis, we pursue to build a robust RS algorithm and validate our algorithm works robustly by performing extensive experiment with real world datasets. The most of existing solutions on robust RS algorithms have a approach to filter out suspicious users using similarity or clustering. The solutions need to examine all possible information to increase the robustness when they perform filtering suspicious or malicious users or their rating information.

Our intuition is that if a RS can suggest appropriate recommendations to the system users regardless of whether the ratings are come from Sybils or not, it intrinsically overcomes the limitations of the previous approaches such as Sybils filtering overhead from RS matrix. Without the filtering process in huge RS matrix, we consider the amount of information given by a RS users. Inspired by this intuition, we propose a **RobuRec** (**Robu**st **Rec**ommendation) algorithm in this thesis which can dramatically relieve the Sybils' impact in the RS without any knowledge of users and under the unlimited number of Sybils. It is worthwhile to

notice that the aim of RobuRec is to mitigate the Sybil's bogus rating activities rather than specify or nullify Sybil's influence. RobuRec exploits *admission control* to monitor and decide whether the newly introduced rating values are acceptable or not. The admission control makes decisions for each rating value according to the *enough information* status where the *enough information* is interpreted as a status of the RS at a specific time instance. If the RS is in enough information status, the RS does not need *additional information* (i.e., additional rating values) to predict rating values. In other words, RobuRec does nothing whenever it enters the *enough information* status, and our scheme RobuRec, however, maintains the results of the recommendation outcomes precisely. Moreover, the RS does not need to consider or identify which rating values are malicious or honest as long as the RS is in the *enough information* status.

One may argue that our work is partial increment of existing work such as [3]. However, we note that even though the concept of *enough information* was introduced in Dsybil [3], our solution has the following significant differences: (i) RobuRec can be deployed onto the general RSs that based on a general scoring system (e.x., 1$\sim$5), whereas the application of Dsybil scheme is restricted in binary feedback systems. (ii) RobuRec considers both positive and negative (enough) information while Dsybil scheme did not.

We will elaborate on the concept of positive and negative enough information in Chapter 4. We have performed extensive empirical evaluations to verify the performance of our proposed scheme, RobuRec. The experimental results demonstrate that RobuRec achieves the best performance against comparable robust RS schemes such as principal component analysis (PCA) approach [6] and Least

Trimmed Squares Matrix Factorization (LTSMF) approach [4] in terms of Prediction Shift (PS) and Hit Ratio (HR) over the various Sybil attack strategies.

The main contributions in this paper are summarized as follows:

- We propose a robust model-based Collaborative Filtering (CF) algorithm called RobuRec leveraging enough information to alleviate the Sybil's negative influence on RSs. RobuRec includes minimum number of users for ensuring enough information on the general RSs through admission control. To the best of our knowledge, RobuRec is the first step harnessing the concept of the enough information against Sybil attack on the general RSs.
- To increase the confidence of the evaluations over diverse purposed RSs, we crawled a real world recommendation dataset, Daum-movie dataset (a famous movie recommendation web site (http://movie.daum.net) in South Korea). The Daum-movie dataset is used for comparing the performance between given datasets and real world dataset. In addition, we fully implemented two popular robust RS schemes, PCA and LTSMF, as our counterpart schemes.
- We have conducted extensive experiments and verified the performance of RobuRec with three datasets (MovieLens, Epinion, and Daum-movie) and various attack scenarios (random, average, bandwagon, and segment attack). The impact of attack sizes is also compared with three different datasets. The evaluation results confirm that RobuRec outperforms the comparable schemes in terms of PS and HR substantially. The PS of RobuRec shows 21% lower than that of LTSMF and 49% lower than that of PCA on average. The

HR shows 15% and 34% lower than that of PCA and LTSMF scheme on average.

1.3 Thesis Organization

The rest of this thesis is organized as follows. We review the previous RS approaches and their Sybil attack defense methodologies in Chapter 2. In Chapter 3, we present a system model including target applications, an attack model, and assumptions. We explain how the RobuRec algorithm is designed and works in Chapter 4. The performance evaluations are presented in Chapter 5. Finally, we close this paper with conclusion in Chapter 6.

Chapter 2

Related Work

Predicting correct information in RSs is highly dependent on the consistency of RS matrix, implying maintaining datasets in RS without manipulations. Following this principle, designing robust RS against Sybil attack has been actively investigated. In this section, we briefly review the major research categories (i.e., RS approaches, Sybil attack defense schemes, and robust RS algorithms) relevant to our approach.

2.1 RS approaches

RSs are defined to a software tool and techniques providing suggestions for products or items to the relevant users [8–10]. The concept of items in RSs is an object in which the generalized term should be denoted as what RSs to users. RSs have many functions such as 'increase the number of items', 'sell more diverse items', 'increase customer satisfaction', 'increase user fidelity', 'better understand what the user wants', and so on [11]. However, the core role among the various functions is to predict what item(s) should be recommended precisely.

To conduct recommending items, knowledge-based RSs were proposed. The knowledge-based RSs recommend items by focusing on specific domain knowledge about how specific feature of items meets the needs of users, which has two major categories (case- and constraint-based RS). Case-based RS was introduced with suggesting the way of estimating similarity between user needs and item descriptions [12, 13]. Constraint-based RSs recommend items using predefined recommender knowledge bases that contain explicit rules [14–16] On the other RS research community, communty-based RS exploiting social relations between users [17–20].

The RSs were initiated as a major research area after CF was introduced since mid-1990s [21]. Firstly, CF exploits interactions between users in a RS (i.e., analyze historical interactions between users) to find out good user-item matches. CF has two sub-classes which are memory-based and model-based (the former, sometimes, called as neighborhood-based) [22–28]. The memory-based models commonly use the similarity between active users in a RS. A common similarity measure is Pearson correlation coefficient (PCC) [27] and other variants can be applied to a RS when the RS computes similarities between users. Item-based CF is an alternative of the model-based CF which harnesses a similarity between items [29]. The model-based CF uses hidden features unlike memory-based CF utilizing statistics between users or items. Matrix Factorization (MF) with gradient descent and alternating least squares was introduced in [30] for predicting hidden rating scores in a RS. The memory-based models with MF were widely appeared in various researches [31–37]. Secondly, the content-based algorithms try to recommend items that has the same feature of the user in the past, or selects the best matched

items with predefined rules [22]. Information retrieval approaches from context and classification method are considered in [24, 38–43], respectively.

Finally, the hybrid-method combines more than two schemes from CF and content-based algorithm in order to increase the performance (prediction accuracy) of RSs [8, 43–48].

2.2 Sybil Attack Defense

As the analyzing social networks, the researches finding the social networks were introduced [49–51]. To neutralize the Sybil attack, random walks [52] in a social graph were considered in [53–59]. The approaches leveraging random walks attempt to find out Sybil identities by estimating the intersection probability between users. Once a system decides that a user is a Sybil, the system controls the admission of such user. However, admission control using the intersection probability of the random walks may neither operate on RSs nor limit the number of Sybils sufficiently. One of the studies on Sybil attacks proposed a scheme called SybilGuard [53] leveraging random walks in a social network. SybilGuard limits the number of accepted Sybils with theoretical bound $O(\sqrt{n}\ log\ n)$ where n is the number of honest users in a social network. Further, Yu *et al.* proposed an advanced scheme, SybilLimit [54], which can limit the number of Sybils with near optimal bound $O(log\ n)$. Tran *et al.* [60] considered admission control with centralized/decentralized version of the ticket distribution scheme based on the concept of adaptive maximum flow. They focused on that bottlenecks in a social network graph exist between normal users and Sybil users. This system controls the number of users' admissions with a fixed threshold on each user. Based on their previous

research, Tran *et al.* proposed GateKeeper [61] which can limit the number of Sybils with theoretical bound $O(log\ k)$ where k is the number of connected edges between honest users and Sybil group. These approaches can control the number of Sybils, however, quite differed from RobuRec. RobuRec limits the admission of users' ratings according to each item's condition dynamically, whereas GateKeeper considered a static threshold under the global view. The performance of GateKeeper showed the better performance than SybilLimit [54] from their experimental study. However, Wei *et al.* (2012) witnessed that GateKeeper does not work well with their real world datasets [62].

As a result, the approaches exploiting the intersection probability of random walks or the bottleneck phenomena in social network graphs are not effectively deployed onto RSs. Also, their admission control schemes are different from RobuRec in that they do not deal with ranking and recommendation algorithms.

2.3 Robust RS Approaches

There are two folds in terms of RS security, which are privacy and robustness issues. In this sub-section we briefly explore the two issues on RSs.

First, regarding privacy, RSs collect as many evaluation information as possible to increase the accuracy of recommendations since RSs provide excellent personalized and customized information to users. This nature of RSs leads negative impact on privacy issue in the system users. To tackle this, researchers explored privacy-protecting RSs focusing on how RSs parsimoniously use user data [63–71]. The researches provide good research insights on the security of RSs, we are mainly interested in robustness as stated belows.

Second, the robustness issue (the main focus of this thesis) was introduce due to the witness on weakness of RSs since 2002 [11]. Under various attacks on RSs (detail description on the attack is stated on Section 3.3.), the main interest of robustness is to design RSs that recommend items properly and robustly even under attacks. Significant researches have done to defend attacks on RSs [72–88]

Similar to our proposal, Yu *et al.* (2009) proposed *Dsybil* which exploits user's feedback by introducing the concept of enough information in RSs. However, the deployment of Dsybil is limited to the systems that operate based on two types of feedbacks, 'good' or 'bad', from users. In contrast, RobuRec considers all possible scoring systems, and thus can be applied on general RSs. Besides, RobuRec reflects both positive and negative enough information, whereas Dsybil considers only positive enough information with 'good' or 'bad' feedback information. For this reason, RobuRec can deal with a push attack as well as a nuke attack, Dsybil, however, treats the push attack only. The push attack intends to boost up the reputation of a target item, and the nuke attack intends to decrease the reputation of a target item, respectively. Mobasher *et al.* (2007) classified and summarized various attack scenarios and RS performance metrics [88]. The authors of [7] proposed probabilistic latent semantic analysis (PLSA). Later the PLSA, Mehta *et al.* (2007) suggested a principal component analysis (PCA)-based algorithm by leveraging the fact that Sybils are highly correlated to each other [5]. Mehta & Nejdl (2008) enhanced the PCA-based algorithm and introduced the VarSelectSVD algorithm by combining PCA and SVD method [6]. Cheng & Hurley (2010) proposed least trimmed squares matrix factorization (LTSMF) by eliminating outliers which have the high value of residual error because Sybils tend to give maximum values

to their target item [4]. However, none of the robust algorithms [4–7] considers the concept of information level (enough information) in the model-based RSs.

Chapter 3

System Model

RSs are the systems that select and return a list of recommended items for the system users. The features or outcomes of the RSs can be applied to various ways such as friends, news, movie, food, music, or book recommendation. However, generic functions of the RS are summarized into two categories: (i) predicting the rating value of a user on a targeted item and (ii) suggesting a certain number of items for a user. In this thesis, we mainly focus on predicting the rating values (the former function of RS). However, we note that if we predict rating values precisely, an RS can recommend a fixed number of items to system users (the latter function of RSs) using the predicted values. Also, we use a basic MF method due to its popularity and accuracy.

The RS has at least one rating matrix containing users, items, and rating values given by the users on the items. For example, if a RS has m users and n items, the RS can be represented as a matrix form of $R_{m \times n}$ with κ rating values. An attacker creates a number of fake identities to make the RS misbehave or return wrong items to normal users. There might be various rating systems with various

values. For instance, one may use 'good' or 'bad'. Possibly, the other can use 'like' or 'dislike', alternatively. A RS can adopt integer values such as 1, 2, ..., 5 to represent preference on each item. We use general integer scale values ranges from V_{min} to V_{max}. In this paper, we consider $V_{min} = 1$ and $V_{max} = \Lambda$, where Λ is a positive integer. If a rating matrix has zero value or empty value, it means the user does not give a rating value on the items.

There possibly exists the use of implicit feedback such as browsing history or purchase records. Such information not available to RSs in recommendation process. However, the information can be represented by binary ranking system (e.x.,'1' for 'purchased' and '0' for 'not purchased', or integer ranking for visiting frequency). The binary ranking transformation might be considered as little informative, but Marlin *et al.* proved that that kind of implicit data can improve prediction history [89]. Therefore, by transforming the binary information into integer ranking system, we believe not only any ranking systems can be applied into our scheme but also both explicit and implicit ranking work on top of our approach.

When each user enters a RS, he/she reviews and estimates item(s) in some way. We assume that a number of Sybils can enter a RS even under the condition that a Sybil defense mechanism is working. Also, we do not limit the number of Sybils in the RS. Let t_s be discrete time at s, where $s = \{0, 1, 3, \cdots\}$. Let $C_{i_x}^{all}$ be the number of rating values on item i_x at time $t_{s+\tau}$ including newly introduced honest users and Sybils. Let $C_{i_x}^{h}$ and $C_{i_x}^{\xi}$ be the number of honest users and Sybils who rated on an item i_x , where $x = \{1, 2, 3, \cdots\}$, respectively. Clearly, $C_{i_x}^{all} = C_{i_x}^{h} + C_{i_x}^{\xi}$. We illustrate the status of the RS matrix using an example in Fig. 3.1.

In Fig. 3.1, the matrix has m user at time t_s. After time goes, the matrix accepts honest users m' and ξ Sybils additionally, and $C_{i_x}^{all}$ represents the total number of newly accepted users who rated on item. The RS has $R_{m \times n}$ with κ rating values at t_s, where κ is the number of rating values in the initialization phase (i.e., total number of rating values at time t_s).

After some time goes from t_s to $t_{s+\tau}$, the newly introduced honest users and Sybils are included in the RS. Now, the RS has $m + m' + \xi$ users and the updated number of rating values which is $\kappa' = \kappa + \sum_{x=1}^{n} C_{i_x}^{all}$. Although the honest users and Sybils are separated in Fig. 3.1, the RS cannot distinguish honest users and Sybils at any time. Table 3.1 summarizes the notations used in this thesis.

Newly introduced users		*Recommendation items (n items)*					
		i_1	i_2	…	i_x	…	i_n
Honest Users (m')	u_1	1					
	u_2		4		5		
	:		5		3		3
	:	2			2		4
	$u_{m'}$	3	1				3
# of honest users who did rating		$C_{i_1}^{h}$	$C_{i_2}^{h}$	…	$C_{i_x}^{h}$	…	$C_{i_n}^{h}$
Sybil Users (ξ)	u_1	1			5		2
	u_2		2		1		
	:	:	:		:		:
	u_ξ	1	4		4		3
# of Sybils who did rating		$C_{i_1}^{\xi}$	$C_{i_2}^{\xi}$	…	$C_{i_x}^{\xi}$	…	$C_{i_n}^{\xi}$
# of users who did rating		$C_{i_1}^{all}$	$C_{i_2}^{all}$	…	$C_{i_x}^{all}$	…	$C_{i_n}^{all}$

Figure 3.1 A rating matrix of RS at time $t_{s+\tau}$

Table 3.1 Notations in this thesis

Notation	Description
m	the number of honest users in the initialization phase
n	the number of items in the initialization phase
κ	the number of rating values in the initialization phase
τ	updating time interval
m'	newly introduced honest users after τ
κ'	updated number of rating values after τ
ξ	newly introduced Sybils after τ
$C_{i_x}^h$	the number of honest users who rated on the item i_x
$C_{i_x}^\xi$	the number of Sybils who rated on the item i_x
$sumv_{i_x}^h$	the sum of honest users' rating values on the item i_x
α	the multiplicative factor when the rating value is positive rating
β	the multiplicative factor when the rating value is negative rating
C_{max}	upper threshold of *Trust*
C_{min}	lower threshold of *Trust*
V_{max}	maximum rating value
V_{min}	minimum rating value
$v_{i_x}^j$	the rate value on the item i_x of user j
$R_{m\times n}$	the rating matrix before Initialization phase containing m honest users and n items
$R'_{(m+m'+\xi)\times n}$	The updated matrix after admission control phase containing $m+m'$ honest users, ξ Sybil users and n items

3.1 Target Applications

As we stated in the previous section, Dsybil using good or bad rating values was introduced in [3] with theoretical bound of Sybil admission. However, its usage is limited since it does not work with variable rating range of $[V_{min}, V_{max}]$. In contrast, RobuRec can be adopted by general RSs based on integer scale rating values since we do not restrict any rating scale or range.

In addition, the binary feedback terms such as 'good' or 'bad' or 'like' or 'dislike' can be easily converted into integer values (ex., 'good' = 2 and 'bad' = 1, or 'like' = 5 and 'dislike' = 1). This implies that the binary feedbacks of RSs falls into the general ranking range RSs. As are mentioned before, we use the rating scale of integer with range of $[V_{min} = 1, V_{max} = \Lambda]$ in this paper since we observe the majority of RSs use rating scale with variable range. In our experiment, Λ is set to 5 in two datasets (MovieLens and Epinion) and set to 10 in one dataset (Daum-movie), respectively.

3.2 Strong Attacker

Since an attacker can have different kinds of knowledge on the RSs, we assume a strong attacker who knows the whole status of RS before he starts the attack. The item popularity can be provided to the attacker due to the nature of RS that anyone may see the popular items in the RS by just querying random interested items. And we assume that the attacker knows the number of items on the RS.

This assumption is reasonable since the attacker is able to browse items and to acquire which items are populated in the RS, easily. Additionally, the attacker

can identify which items belong to any group, genre, or segment by simply probing the target RS. Furthermore, the attacker knows the overall rating values as well as the mean of each item's rating values in our system model.

3.3 Attack Model

In general, there exist two kinds of Sybil attacks, *push* and *nuke* attack. The push attack attempts to get positive recommendations for the target item(s), while the nuke attack attempts to give negative recommendations for the target item(s) [88]. We focus on push attacks as our main attack scenarios since attackers tend to increase his popularity or recommendation probability in the RS. The push attack has four possible attack scenarios (random, average, bandwagon, and segment attack). RobuRec is also designed to be able to fight against those four push attacks.

Let us now examine in greater detail the push attack model. When an attacker (say 'Alice') succeeds in injecting a number of Sybils in a RS, Alice tries to distort the RS matrix in order to reflect a malicious goal onto the outcomes of the RS. If Alice considers push attack, the target item is assigned with maximum rating value and other target-related items (called filler items) filled with different rating values in terms of intended attack scenario. Alice can give more fake ratings with additional items (called selected items) to make her attack sophisticated in case of bandwagon and segment attack. Let I_F and I_S be the set of filler items and selective items, respectively. The filler item size and selective item size (namely, the number of filler items and selective items) are expressed as $|I_F|$ and $|I_S|$, respectively. We depict the attack related sizes (I_F, I_S, and attack size) in Fig. 3.2.

In Fig. 3.2, the attack size(ξ) is generated according to a certain portion of the train set (i.e., User-Item matrix in this figure) users along with one of possible Sybil attack strategies. The filler items (I_F) can be controlled by an attacker and the size varies. The selective items (I_S) represent the popular items that an attacker gives some rating values to make high correlation with the popular items. The target item is selected by an attacker to boost up its recommendation probability.

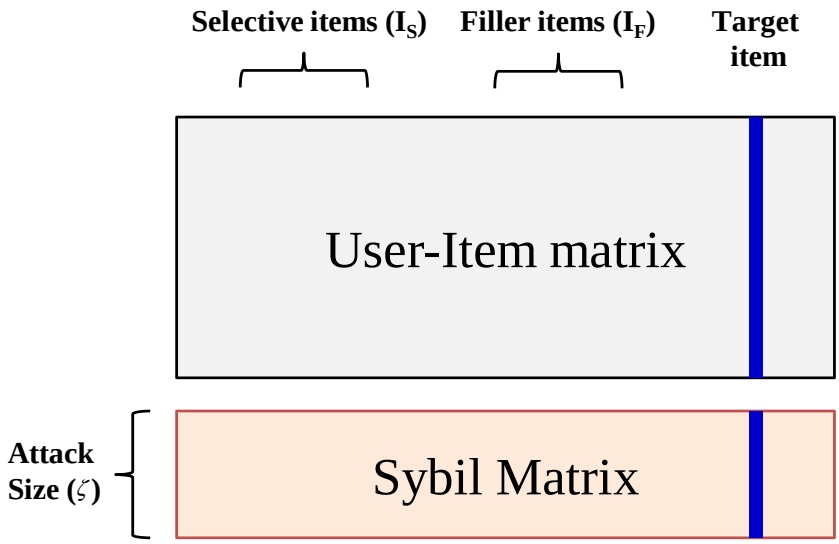


Figure 3.2 Concept of Sybil attack size

The detailed scenarios of four attack strategies are summarized as follows:

- Random attack: an attacker gives V_{max} values to the target item and the rating values with normal distribution around the system mean to the filler items. Since random attack is intuitive and simple approach, the cost of attack is the lowest among attack schemes used in this thesis. Note that the naïve attack is possible by giving V_{max} values to the target item and V_{min} on filler item. The naïve attack is proven to be easily detectable [75, 78]. Therefore, as assumed in Section 3.4, we do not test the detectable

naïve attack but evaluate random attack with the assumption of the strong attacker.

- Average attack: an attacker gives V_{max} values to the target item and the rating values with normal distribution around at each item mean to the filler items. Average attack needs more extensive knowledge since the attacker should know the individual mean and standard deviation for each item. Also, This attack is more powerful than random attacks [78]

- Bandwagon attack: an attacker gives V_{max} values to the filler items, V_{max} values to the selected items. The goal of the bandwagon attack is to associate the target item with a number of popular items (frequently rated items) so as to increase the probability of recommending with the popular items. In the bandwagon attack, the selected items are chosen such that the items have the most frequently rated items (popular items over all items). If Alice uses the selected items, she can make her target item get related with such popular items. As a result, the bandwagon attack is smarter than the average attack in terms of attacker's prior knowledge.

- Segment attack: an attacker gives the normal distribution around the global mean to the filler items. V_{max} values are given to the selected items. In segment attack, the selected items are chosen such that the items have the most frequently rated items (popular items) in a segment (genre or special group). In our setting, we define the selected items of segment attack as the most frequently rating items in the segment.

We summarized the types of attacks in Table 3.2.

3.4 Model Assumptions

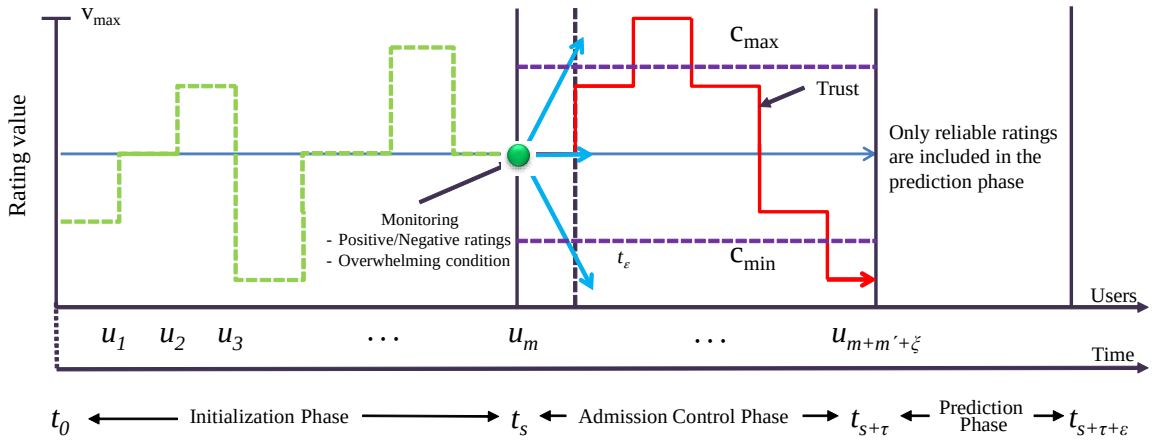


Figure 3.3 The initialization, admission control and prediction phase

Before we go forward, we set up three assumptions as shown in 3.3: (i) Initialization Phase: we assume that RobuRec starts with a given rating matrix. The users and their rating values on each item in the given matrix are assumed as honest. This honest setting will be used for RobuRec's starting status. For example, an item receives rating values from each honest user who gives any ratings between time interval (start-point of RobuRec) and (mid-point of that) in Fig. 3.3. This experimental set-up is commonly used for testing the robustness of algo-

Table 3.2 Sybil attacks on RSs

Attack type	Selected items	Filler items	Target item
Random	Not used	Normal Dist. with Global mean	Max. value
Average	Not used	Normal Dist. with each item mean	Max. value
Bandwagon	Most rated items with Max. value	Normal Dist. with global mean	Max. value
Segment	Popular items in a group with Max. value	Min. value with global mean	Max. value

rithms [2, 4–7, 88]. And this assumption does not conflict with our goal since we test the robustness of algorithm not with initial rating matrix containing honest rating values but with rating matrix containing both honest and Sybil rating values after admission control phase. Note that RobuRec does not consider whether the accepted rating values are honest or not. (ii) Admission Control Phase: After an updating interval (time units), RobuRec performs admission control against newly introduced users regardless of their honesty or dishonesty. (iii) Rating Prediction Phase: Once the admission control is done and RobuRec admits a number of new users, then RobuRec performs a prediction of rating values using any kind of model-based CF approaches. In other words, our scheme can be deployed onto any kind of prediction algorithms with admission control. This flexibility is one of salient features of RobuRec.

Chapter 4

RobuRec Design

4.1 Algorithm Intuition

The key intuition of RobuRec is that if an item has *enough information* to decide the rating values, the item does not use the information of additional users' rating values. We use two concepts, enough information and *overwhelming* condition, already stated in [3]. They assume an object is overwhelming if the total Trust of the identities voting for the object is at least some constant c, where $c > S$. S is a real-valued Trust, initialized to some seed value $S > 0$, for each identity. However, we set the overwhelming condition in two directions; *positive overwhelming condition* if the *Trust* value exceeds C_{max} and *negative overwhelming condition* if the Trust value goes under C_{min}. *Enough information* denotes the status that an item receives the *Trust* value which is higher than *positive overwhelming condition* or lower than *negative overwhelming condition*. Once an item gets in e*enough information*, RobuRec does not reflect the users' ratings on the rating prediction stage until the item escapes the *enough information* status.

And H. Yu, *et al.* (2009) introduced the concepts in which its algorithm ran-

domly chooses a recommended item to a user of the RS under the condition of either *overwhelming* or not. However, RobuRec exploits those two concepts in completely different way. We elaborate on this as follows. With the starting status (m users, n items, and k ratings), RobuRec computes *Agreement* on each item. We set the *Agreement* as a mean value of given rating values on each item since the mean value can be interpreted as the users' final Agreements for each item. *Trust* of individual item is directly computed from each item's *Agreement* using C_{max}. Each item has a number of newly introduced users (m' honest users, ξ Sybil users) at time $t_{s+\tau}$. RobuRec updates the *Trust* with the new users' rating values ($m' + \xi$ ratings). Then, the *Trust* changes whenever RobuRec performs updating the values.

RobuRec consists of three phases: (i) initialization phase, (ii) admission control phase, and (iii) rating prediction phase as shown in Fig. 3. In the initialization phase $[t_0, t_s)$, RobuRec takes $C_{i_x}^h$ honest users and computes *Agreement* and *Trust* values. In the admission control $[t_s, t_{s+\tau})$ and rating prediction phase $[t_{s+\tau}, t_{s+\tau+\epsilon})$, RobuRec uses *Trust* values with thresholds (C_{max}, C_{min}) to decide *overwhelming* condition (i.e., "positive" or "negative") when newly introduced users (m' honest users, ξ Sybil users) are admitted into the RS. If the new users are honest, the Trust will stay around Agreement value of each item. However, if the new users are Sybils, the *Trust* may keep growing or decreasing since Alice will give maximum values to the target item and minimum values to the selective items in the push attack. Therefore, if we set up a threshold value in upper and lower bound of the *Trust* (C_{max} and C_{min}, respectively), we can block malicious ratings by flagging them as Sybils, indicating exceeded *Trust* values over C_{max} or under C_{min} (i.e.,

enough information status). On the other hand, Alice will give rating values along with normal distribution to the filler items in case of random, average, and segment attack under the assumption of 'strong attacker' stated in Section 3.2. In that case, RobuRec limits the number of admitting users based on $sumv_{i_x}^h$ (the sum of the honest users' rating values on the item i_x) and m (the number of honest users in the initialization phase).

4.2 Initialization Phase

Algorithm 1: Initialization

Input: $R_{m\times n}$, k, V_max
Output: $Agreement$, $Trust$, $RatingCount$

```
for x = 1 to n do
    compute C_{i_x}^h, sumv_{i_x}^h;
    compute Agreement of i_x;
    add RatingCount.append(C_{i_x}^h);
    add Agreement.append(Agreement_{i_x});
    compute Trust of i_x;
    Trust.append(Trust_{i_x});
end
Return Agreement, Trust, RatngCount;
```

Figure 4.1 The initialization phase procedure

For the initialization phase, let $\Sigma_{i_x}^h$ be the sum of honest users' rating values for the item i_x, where $x \in \{1, 2, \ldots, n\}$. With the given users admitted during the time interval: $[t_0, t_s)$ in Fig. 3.3, RobuRec sets *Agreement* and *Trust* for each item (i_x) as follows:

$$Agreement_{i_x} = \left\| \frac{\Sigma_{i_x}^h}{C_{i_x}^h} \right\|, \tag{4.1}$$

where $\|\cdot\|$ *is the nearest integer function.*

$$Trust_{ix} = \frac{Agreement_{i_x}}{V_{max}} \times C_{max} \tag{4.2}$$

We set the mean of the rating values on an item as *Agreement* as Eq. 4.1 since we consider that the mean can be considered as Agreement level of i_x and additional users may give rating value around the Agreement level. In Eq. 4.2, the *Trust* is normalized by V_{max} with respect to C_{max}. Also, RobuRec sets up the C_{max} and C_{min} values as an upper and lower threshold as stated in Section 4.1. The parameters (C_{min}, C_{max}, and V_{max}) are given by the RobuRec algorithm as an input, which are the given fixed values. In the initialization phase, RobuRec computes all *Agreement* and emphTrust values on every item, and uses those values for the admission control phase. The detailed procedure of initialization phase is presented in Fig.4.1.

4.3 Admission Control Phase

Once the values of the *Agreement* and *Trust* are set up from the initialization phase, RobuRec starts the admission control phase to block unnecessary users who are not needed for RobuRec to predict rating values. This implies the item has already gotten in enough information for the prediction. RobuRec performs admission control with newly introduced users (i.e., $m' + \xi$ users) on each item, and then updates *Trust* of every items, iteratively. Thus, the *Trust* value of each item keeps changing until RobuRec finishes the admission control phase. Note that RobuRec uses the Agreement from Algorithm 1 as a fixed value during the admission control phase while the values of *Trust* keep varying using Algorithm 2. To verify the enough information status of an item, we define a few concepts

stated in Section 4.1 as follows:

- *Positive (negative) rating*: if the rating of a user is greater (less) than *Agreement*, RobuRec decides the rating is a *positive (negative) rating*.
- *Same rating*: if a rating of a user is equal to the *Agreement*, RobuRec decides the rating is same rating.
- *Positive (negative) overwhelming* condition: if the *Trust* of an item is greater (less) than C_{min} (C_{min}), RobuRec decides this condition is the *positive (negative) overwhelming* condition.

RobuRec considers that an item has *enough information* if the *Trust* value enters either *positive* or *negative overwhelming* condition. In any *overwhelming* cases, RobuRec does not allow the new user's rating(s) to be included in the prediction phase. As shown in Fig. 4.2, if a new user's rating is a positive rating or negative rating, then RobuRec compares *Trust* with C_{max} or C_{max} to decide whether *Trust* should be updated or not. By examining the rating value, RobuRec divides the result as one of following three cases (i.e., positive rating, negative rating, or same rating) and decides that the rating value is reliable or unreliable. Only the reliable rating values are used for prediction phase. The operations of RobuRec on each case are summarized as follows:

- If *Trust* is greater than or equal to C_{max} in case of positive rating, RobuRec decides the item has a *positive overwhelming* condition and flags the rating value as unreliable, otherwise it sets the rating value as reliable and updates *Trust* multiplied by α where $1 < \alpha < 2$.

Algorithm 2: Admission Control

Input: $R'_{(m+m'+\xi)\times n}$, α, β, C_{max}, C_{min}, V_{max}, $Agreement$, $Trust$, $RatingCount$

Output: $Agreement$, $Trust$, $RatingCount$

```
for x = 1 to n do
    for j = m to m + m' + ξ do
        if v_{i_x}^j then
            //positive rating;
            if v_{i_x}^j > Agreement[i_x] then
                //positive overwhelming condition;
                if Trust[i_x] ≥ C_max then
                    flag v_{i_x}^j as Unreliable;
                end
                else
                    flag v_{i_x}^j as Reliable;
                    Trust[i_x] = Trust[i_x] × α;
                end
            end
            //negative rating;
            if v_{i_x}^j ≤ Agreement[i_x] then
                //negative overwhelming condition;
                if Trust[i_x] ≤ C_min then
                    flag v_{i_x}^j as Unreliable;
                end
                else
                    flag v_{i_x}^j as Reliable;
                    Trust[i_x] = Trust[i_x] × β;
                end
            end
            //same rating ;
            if v_{i_x}^j == Agreement[i_x] then
                count++;
                if count ≤ max(m × 0.2, RatingCount[i_x] × 2) then
                    flag v_{i_x}^j as Reliable;
                end
            end
        end
    end
    count = 0 //reset counter ;
end
Return flagged matrix R'_{(m+m'+ξ)×n};
```

Figure 4.2 The admission phase procedure

- For the case of negative rating, if *Trust* is less than or equal to C_{min}, RobuRec decides the item reaches the *negative overwhelming* condition and flags the rating values as unreliable, otherwise it sets the rating value as reliable and updates *Trust* multiplied by β where $\beta = 2 - \alpha$.
- In case of same rating, RobuRec flags the rating as reliable and does not update *Trust*.

Variable α values are settable. To fix the α, we conducted parameter test with three datasets. The detailed analysis on the α is stated in Section 5.

However, if Alice gives rating values with normal distribution to filler items in each case of random, average, and segment attack, the same rating can be occurred continuously. Als,o there exists a chance that the *Agreement* is equal to filler items' values in segment attack. Since RobuRec does not limit the admission of the rating values in case of the same rating, Sybil's rating may keep being admitted. To prevent this situation, RebuRec limits the number of same rating (the number of same rating counts) such that RobuRec admits rating values when the following inequality is satisfied.

$$\textit{the number of same rating counts} \leq min(m \times \phi, RatingCount[C_{i_x}^h] \times \varphi), \quad (4.3)$$

where m is the number of honest users obtained from initialization phase and $RatingCount[C_{i_x}^h]$ is the number of honest users' rating values on item i_x.

In Equation 4.3, the parameter ϕ and φ is the maximum tolerance level for an item according to the total number of users and the total number of honest

ratings on item i_x, respectively. And the $C_{i_x}^h$ is equivalent to the number of flagged (reliable) users on item i_x at time t_s.

Our setting implies that RobuRec keeps receiving the same rating values, regardless the rating values honest or not, until reaching the number of honest users that is proportional to ϕ, or the number of honest rating values that is proportional to φ. Same rating implies basically honest ratings, however, extremely large same rating results the distorted outcomes in a RS. Our intuition is that RobuRec accepts as many as possible same rating with upper bound ϕ. However, in the scenario of having a few honest users, RobuRec only admits small same rating. To prevent block same rating in the scenario, we set another parameter, φ. We set empirically the maximum tolerance level such that $\phi = 0.2$ and $\varphi = 2$.

4.4 Rating Prediction Phase

With the updated matrix ($R'_{(m+m'+\xi)\times n}$) which contains only flagged (reliable) users from the admission control phase, RobuRec starts the rating prediction phase. For that, any rating prediction schemes among the model-based CF can be applied. We used the MF approach since its prediction accuracy is high and vastly used in model-based CF. The MF methods used are expressed by mathematical form in [36, 90–93].

For the explanation of prediction, we explain what we used MF for rating prediction phase in this thesis. A low-rank MF seeks to an approximated rating matrix matrix by multiplication of l-rank factors. We follow the prediction approach with [91]. The brief explanation of MF is as followings.

$$R \approx U^T V, \tag{4.4}$$

In Eq. 4.4, U, V matrix where R is $m \times n$ matrix, U is $l \times m$ matrix (l is l-rank factors in R). V is $l \times n$ matrix, and and $l < min(m, n)$.

Traditionally in Singular Value Decomposition (SVD) [94], minimizing error is performed using following equation:

$$\frac{1}{2}||R - U^T V||_F^2, \tag{4.5}$$

where $||\cdot||_F^2$ *denotes the Frobenius norm.*

Since a recommendation matrix contains a large portion of empty elements (i.e., missing ratings), we perform on observed elements in matrix R in Eq. 4.4 and Eq. 4.5 facilitating a indicator function. Therefore, the Eq. 4.5 can be represented as stated in Eq. 4.6.

$$\min_{U,V} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{i,j}(R_{i,j} - U_i^T V_j)^2 \tag{4.6}$$

where $I_{i,j}$ *denotes the indicator function 1 if user* i *rated on item* j*.*

To avoid overfitting, two requlariza tion terms are generally added into Eq. 4.6.

$$\min_{U,V} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{i,j}(R_{i,j} - U_i^T V_j)^2 + \frac{\gamma}{2}(||U||_F^2 + ||V||_F^2) \tag{4.7}$$

The matrix, $R'_{(m+m'+\xi)\times n}$, returned from admission control phase is the input of rating prediction phase (i.e., $R \leftarrow R'_{(m+m'+\xi)} \times n$). We used the well-known

Gradient descent method to find matrix U and V to predict rating values in this thesis. The changes of the matrix (R) after each phase are portrayed in Fig. 4.3 $\sim$ Fig. 4.7.

		Recommendation items (n *items*)					
		i_1	i_2	…	i_x	…	i_n
Honest Users (m)	u_1	1					
	u_2		4		5		
	…		5		3		3
	…	2			2		4
	u_m	3	1				2

Figure 4.3 The rating matrix at time t_0

		Recommendation items (n *items*)					
		i_1	i_2	…	i_x	…	i_n
Honest Users (m)	u_1	1,R		…		…	
	u_2		4,R	…	5,R	…	
	…		5,R	…	3,R	…	3,R
	…	2,R		…	2,R	…	4,R
	u_m	3,R	1,R	…		…	2,R
Returned values After init. Phase		$C_{i_1}^h$ $\Sigma v_{i_x}^h$ $Agree'_1$	$C_{i_2}^h$ $\Sigma v_{i_x}^h$ $Agree'_2$	…	$C_{i_x}^h$ $\Sigma v_{i_x}^h$ $Agree'_x$	…	$C_{i_n}^h$ $\Sigma v_{i_x}^h$ $Agree'_n$

Figure 4.4 Matrix status of the initialization phase

RobuRec start with a given matrix at time t_0 Fig. 4.3. Firstly, RobuRec computes all required variables with the matrix a time t_0. The initialization phase generates $C_{i_x}^h$, $C_{i_x}^{\xi}$, and $\sum v_{i_x}^h$ as shown in Fig. 4.4.

		Recommendation items (n *items*)					
		i_1	i_2	…	i_x	…	i_n
Honest Users (m)	u_1	1,R		…		…	
	u_2		4,R	…	5,R	…	
	…		5,R	…	3,R	…	3,R
	…	2,R		…	2,R	…	4,R
	u_m	3,R	1,R	…		…	2,R
Returned values After init. Phase		$C^h_{i_1}$ $\Sigma v^h_{i_x}$ *Agree'*$_1$	$C^h_{i_2}$ $\Sigma v^h_{i_x}$ *Agree'*$_2$	…	$C^h_{i_x}$ $\Sigma v^h_{i_x}$ *Agree'*$_x$	…	$C^h_{i_n}$ $\Sigma v^h_{i_x}$ *Agree'*$_n$
New Honest Users (m')	u_1		3	…	5	…	
	u_2	3		…	2	…	5
	…	…	…	…	…	…	3
	$u_{m'}$	3	1	…	4	…	
Sybil Users (ξ)	u_1	1	2	…	5	…	1
	u_2	2	1	…	5	…	1
	…	…	…	…	…	…	…
	u_ξ	1	1	…	5	…	2

Figure 4.5 Matrix status under a Sybil attack

Figure 4.6 Matrix status of the admission control

After the initialization phase, RobuRec keeps the generated values in memory for the next phase. During the time period $[t_0, t_s]$, a RS receives new ratings both from honest users as well as Sybil users. In Fig. 4.5, the ratings from the honest user are colored in blue; those from Sybil users are colored in Red. As a result, the matrix of the RS grows from R to $R'_{(m+m'+\xi)\times n}$ at time t_s. The admission control phase started at time t_s and ended at time $t_{s+\tau}$. At this phase, RobuRec decides whether the rating values $(C^h_{i_x} + C^\xi_{i_x})$ are reliable or not using $Trust$ for each item.

RobuRec marks ratings R if it is reliable, otherwise, marks U (unreliable) as shown in Fig. 4.6. After the admission control phase, RobuRec performs the prediction phase for computing the hidden rating values (the elements with question marks in Fig. 4.7). In the prediction phase, RobuRec only uses ratings flagged as R (reliable) and using Eq. 4.7.

	i_1	i_2	…	i_x	…	i_n
u_1	1		…		…	?
u_2	?	4	…	5	…	?
…	?	5	…	3	…	3
…	2	?	…	2	…	4
	3	1	…	?	…	2
		?		?		
	?	1	…	?	…	?
	…	…	…	…	…	…
$U_{m+m'\ \xi}$		?	…	5	…	

Figure 4.7 Matrix status of prediction phase

4.5 Dynamic Parameter Control

As we discussed in Section 4.3, RobuRec requires several control parameters (C_{max}, C_{min}, α, and β). Complex control parameters, sometimes, make the prediction results artificial or man-controlled. To simplify RobuRec algorithm and automatically configure the control parameters, we additionally introduce dynamic parameter control approach in this Section. Our goal is to eliminate all control parameters having man-in-the-middle characteristic, and finally set all parameters

reflecting the condition of a RS accordingly. We will show all steps in this section step by step. We simplify the parameter (C_{min}) in Sub-section 4.5.1, then we elaborate new approaches using dynamic and automatic parameter configuration in terms of C_{max} and C_{min} in Sub-section 4.5.2 and 4.5.3, respectively.

4.5.1 Simplifying Control Parameters

To simplify a parameter, we firstly focus on C_{min} since if we consider a ranking system in a RS, the minimum value can be configured with V_{min}. According to Eq. 4.2, the possible range of $Trust$ is in [0.0, 1.0]. Also, we can easily notice the value of $Agreement_{i_x}$ is the same to one of the integer ranking values. However, the case of $Agreement_{i_x} = 0$ happens rarely since the case is possible only when the item i_x has no ratings. Therefore, we conclude possible minimum value of $Agreement_{i_x}$ is approaching to 1.0 as stated in Eq.4.8.

$$PossilbeMin(Agreement_{i_x}) \simeq 1.0 \tag{4.8}$$

To determine C_{min} from C_{max}, we set the C_{min} as follow equation.

$$C_{min} = \frac{PossilbeMin(Agreement_{i_x})}{V_{max}} \times C_{max} = \frac{1.0}{V_{max}} \times C_{max} \tag{4.9}$$

From Eq. 4.9, we eliminate one artificial parameter (C_{min}) by automatic configuration with Eq. 4.8. Note that the β is the deterministic parameter since $\beta = 2.0 - \alpha$. Now, we have two artificial man-in-the-middle parameters (C_{max} and α). We will eliminate the two manual parameters in the following two Subsections.

4.5.2 Dynamic C_{max} Control

In this Sub-section, we elaborate how C_{max} is configured automatically and accordingly under the RS condition. Our intuition is that if we observe the high fluctuation of ratings in a RS, we set C_{max} with higher value. Since the high fluctuation means RobuRec can reach to the *overwhelming condition* easily, which leads RobuRec do not accept honest ratings. To prevent this phenomenon, we monitor the standard deviation (STD) of rating values so that we can analyze the fluctuation of ratings. However, deciding the ratings are highly fluctuated or not is not trivial. To tackle this challenge, we facilitate the fuzzy rule (FR) approach as introduced in [95, 96] and practically tested in [97, 98]. We propose two dynamic C_{max} control schemes. We firstly propose a global STD approach which can configure C_{max} according to global STD value in a RS. Secondly, we propose a FR-based scheme which configures individual C_{max} for each item.

Global STD Approach Our intuition of dynamic C_{max} control is as follow: (i) If the STD of all ratings in $R_{m\times n}$ is large, we increase C_{max} in proportion to STD. (ii) Otherwise, we decrease C_{max} in proportion to STD We mathematically express our intuition as follows:

$$C_{max} = \Upsilon \times STD_{global}, \tag{4.10}$$

where Υ *is set to be one of* C_{max} *maximizing the detection accuracy by observing* $C_{max} \in \{2, 3, 4\}$.

In Eq. 4.10, we set the value of Υ after observing overall PS using three possible

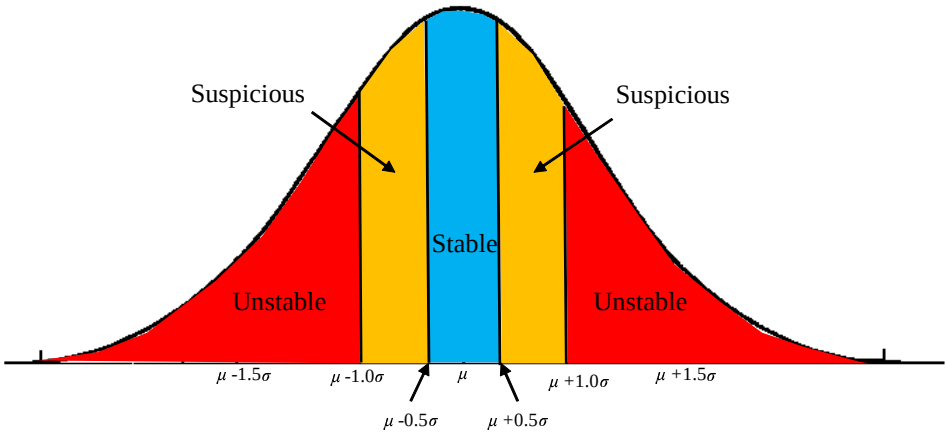


Figure 4.8 Three confidence levels from the observations on rating scores

C_{max}. We found that the overall error (PS) is minimized when C_{max} is equal to 2 (vise versa, maximize the overall detection accuracy).

FR-based Approach. Our intuition on FR-based approach is that we can monitor two criteria (mean and STD): (i) if the mean of rating scores on an item is significantly differs from global mean, it could be an evidence that the ratings on the item are from Sybils with distorted values. (ii) if the STD of rating scores on an item is abnormally higher deviation than normal condition, it may contain the Sybils' perverted ratings leading distorted rating distribution.

To model our intuition formally, we elaborate our idea mathematically. If μ_{i_x} of an item i_x significantly differs from global μ_m, the ratings on the item i_x is *unstable* (i.e., i_x shows significant deviation.) as portrayed in Fig. 4.8. And if σ_{i_x} significantly differs from global σ_s, the ratings on the item i_x is *unstable* as shown in Fig. 4.8. However, our intuition is fuzzy since there is no clear boundary between *stable* and *unstable*. To tackle the fuzziness on our idea, we exploit fuzzy logic to deal with reasoning approximate value.

The first step to the fuzzy logic, we need to build up FR for fuzzy decision. For FR decision, we categorize the condition of a RS (i.e., $R_{m\times n}$) as three conditions: *stable*, *suspicious* and *unstable*. The only information that we can extract from $R_{m\times n}$ are mean and STD for each item. Let μ_{i_x} and σ_{i_x} be the mean and STD of the rating scores on item i_x, respectively. We compute the mean and STD of all μ_{i_x} and σ_{i_x}. Let μ_m and σ_m be the mean and STD of all μ_{i_x}s; let μ_s and σ_s be the mean and STD of all σ_{i_x}s. To monitor system condition, we assume that two distribution follows the normal distribution such that $N(\mu_m, \sigma_m)$ and $N(\mu_s, \sigma_s)$.

To build up FR decision, by directly following the approach [99], we divide the normal distribution $N(\mu_m, \sigma_m)$ and $N(\mu_s, \sigma_s)$ into three intervals, respectively. We portray the concept of the three levels in Fig. 4.8. We use the converted $\mu'_{i_x} \leftarrow \mu_{i_x}$ and $\sigma'_{i_x} \leftarrow \sigma_{i_x}$ via z-transformation, where $\mu'_{i_x} = \frac{\mu_{i_x} - \mu_*}{\sigma_*}$ and $\sigma'_{i_x} = \frac{\sigma_{i_x} - \mu_*}{\sigma_*}$, were $* \in \{m,\ s\}$. We set the decision rules of the analog input values as follows:

- Rule 1: *IF* μ'_{i_x} or σ'_{i_x} is in the range of [-0.5, 0.5), *THEN* the condition is *stable*
- Rule 2: *IF* μ'_{i_x} or σ'_{i_x} is in the range of [-1.0, -0.5) or [0.5, 1.0), *THEN* the condition is *suspicious*
- Rule 3: *IF* μ'_{i_x} or σ'_{i_x} is in the range of $[-\infty$, -1.0) or [1.0, $\infty)$, *THEN* the condition is *unstable*

We select the $max-min$ inferencing and center of gravity approach for defuzzification since the two approaches are widely used in fuzzy rule-based algorithm [95]. The membership functions are designed following the approach in [99] and depict the functions in Fig. 4.9.

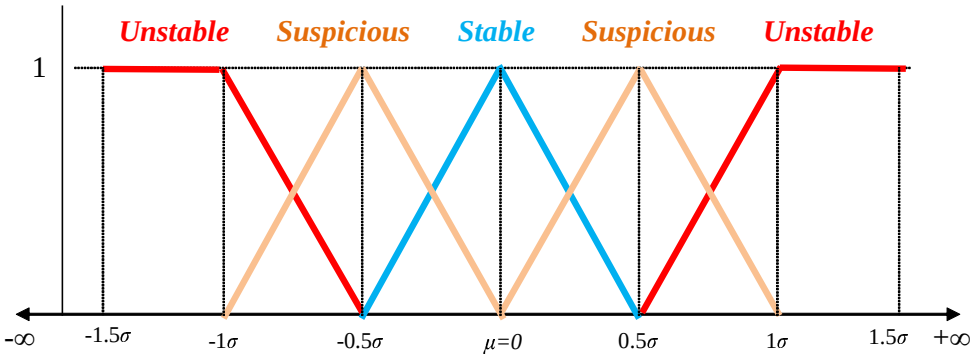


Figure 4.9 Membership functions.

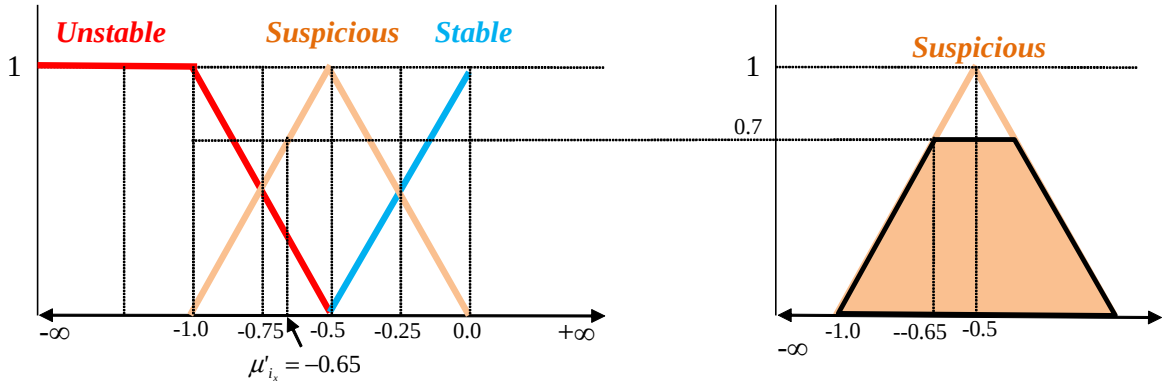
We now elaborate on an example of FR-based decision. Assume that we now monitor an item (μ'_{i_x} and σ'_{i_x}) among n items in $R_{m\times n}$. If we observe $\mu'_{i_x} = -0.65$, the decision is *suspicious* according to decision Rule 2. By applying the membership functions of *suspicious* as shown in Fig. 4.9, RobuRec notice that the inference level is equal to 0.7 as depicted in Fig. 4.10(a). In the case of $\sigma'_{i_x} = 0.2$, the decision is *stable* according to decision Rule 1. Same as computing the inference level of μ'_{i_x}, the inference level of σ'_{i_x} is equal to 0.6 as depicted in Fig. 4.10(b).

The fuzzy number (denoted by A) can be computed by a centroid function in Eq. 4.11. We take the $\overline{x}(A_\odot)$ value directly following [96].

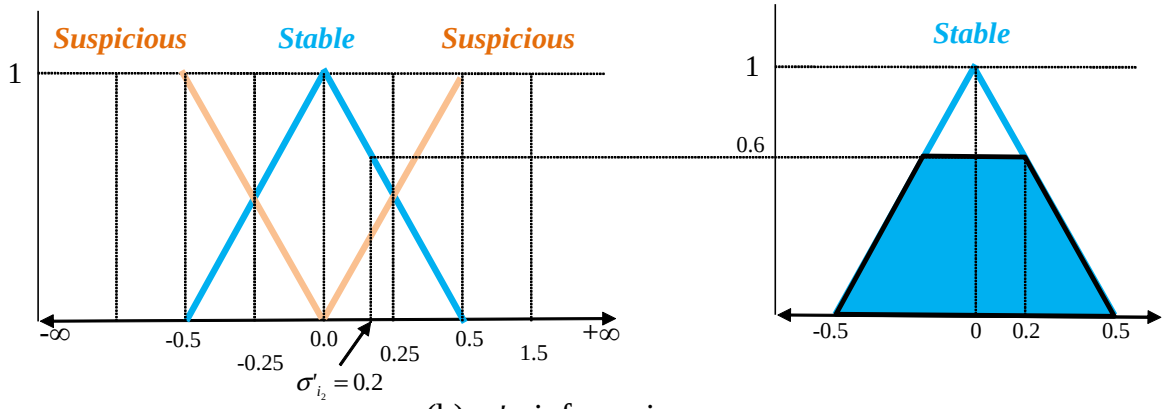
$$\overline{x}(A_\odot) = \frac{\int_{min}^{max} x \cdot \mu(x)\, dx}{\int_{min}^{max} \mu(x)\, dx}, \tag{4.11}$$

where $\odot \in \{\sigma'_{i_x}, \mu'_{i_x}\}$, *and* x *is the output variable,* $\mu(x)$ *is the membership function to which* x *belongs to after mapping, the max is upper limit for defuzzification, and min is lower limit for defuzzification of* $\mu(x)$.

The inferencing steps are depicted in Fig. 4.10(a) and Fig. 4.10(b) of μ'_{x_i} and σ'_{x_i}, respectively. The fuzzy number of A (i.e., $\overline{x}_{A_\odot}$) using Eq. 4.11 can be calculated as follows:



(a) μ'_{i_x} inferencing



(b) σ'_{i_x} inferencing

Figure 4.10 Decision results on two items (μ'_{x_i} and σ'_{x_i}) using max-min inferencing.

$$\overline{x}(A_{\odot}) = \frac{-0.48 - 0.2275 + 0.2275}{0.48 + 0.455} = -0.513 \tag{4.12}$$

For the conceptual understanding, we present the snapshot of computing the inference value in Fig. 4.11. Since FR-based approach decides the system conditions as trinary basis (*stable*, *unstable*, *suspicious*), we apply the value of Eq. 4.12 to the decision rules again. Since the final value of the FD-based approach $\overline{x}(A_{\odot}) = -0.513$, the Rule 2 is applicable and we conclude the rating condition on i_x is *suspicious*.

By following FD-based approach, we can identify condition of all i_x. From the

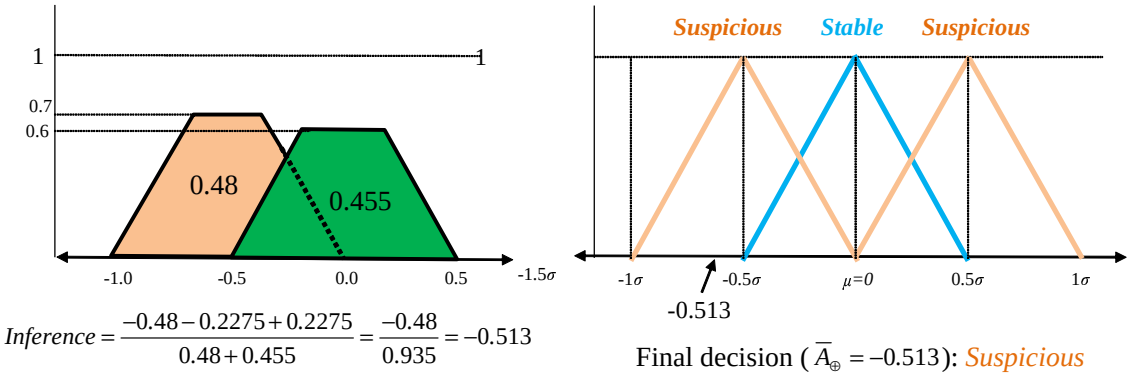


Figure 4.11 Defuzzification and final decision.

condition identification, we can manage C_{max} individually according to each item's condition using the following rule. Note that if system is stable, we adopt small C_{max} since the stable ratings' distribution leads the time to reach the overwhelming condition; if system is unstable, we set larger C_{max} value to increase the time to reach the overwhelming condition.

- If fuzzy-decision is *stable* on item i_x, set $C^x_{max} = C_{global} \times 0.5$.
- If fuzzy-decision is *suspicious* on item i_x, set $C^x_{max} = C_{global} \times 1.0$.
- If fuzzy-decision is *unstable* on item i_x, set $C^x_{max} = C_{global} \times 1.5$.

By facilitating our FD-based approach, we can automatically manage C_{max} accordingly, that also reflects all the items' condition into individual C^x_{max}.

4.5.3 Dynamic Global and Local α Control

In the previous Sub-section 4.5.1 and 4.5.2, the manual setting of required parameters (C_{min} and C_{max}) can be substituted with automatically configured parameters reflecting all items' condition in $R_{m \cdot n}$.

In this Sub-section, we will explore the way of automatic configuration on the parameter α. To facilitate the impact of α, we elaborate our idea. Each item has

large standard deviation (STD) in terms of its rating distribution, it needs more Trust updating to fall into positive or negative overwhelming condition. To control the speed of updating the Trust on each item, we set two rules as follows from the give honest dataset:

- Rule 1: if STD of an item is large, RobuRec performs more aggressive update with such that: High STD $\rightarrow$ increase the number of updates before reaching overwhelming condition $\rightarrow$ decrease in proportional to STD.
- Rule 2: if STD of an item is small, RobuRec performs conservative update with such that: Low STD $\rightarrow$ decrease the number of updates before reaching overwhelming condition $\rightarrow$ increase in proportional to STD.

To hook up α updating with individual STD on each item, we need to formulate updating rule more in formal. First, we analyze the updating procedure of α. As stated in Section 5.4.6, the $Trust$ is updated at most $\lceil \log_{\alpha}(\alpha \cdot C_{max}/Trust) \rceil$ times. Let the number of updates on an item i_x be $n_{i_x}^{update}$. And the STD of an item i_x be STD_{i_x}. We set $n_{i_x}^{update}$ to be inverse proportional to STD_{i_x} of each item (i.e., RobuRec updates $Trust$ more aggressively if STD_{i_x} is large and less aggressively if STD_{i_x} is small.)

More formally, we specify the exact updating value as follows:

$$n_{i_x}^{update} \propto \frac{1}{STD_{i_x}} \times \lceil \log_{\alpha}(\alpha \cdot C_{max}/Trust) \rceil \tag{4.13}$$

To make Eq. 4.13 effective under α updating rules, we set the exact update value as follow:

$$\alpha' \leftarrow \alpha^{1/STD_{i_x}}, \tag{4.14}$$

where the Trust values are fixed constant at time t_τ.

To validate the bound of STD_{i_x}, we explore the lower bound of STD_{i_x} with α'. The possilbe STD_{i_x} is a number in a positive real number. However, if $STD_{i_x} \approx 0.0$, $n_{i_x}^{update}$ is possibly less than 1 meaning that RobuRec never accepts any ratings due to the aggressive updating (i.e., extremely high α). Therefore, we adjust RobuRec such that it has updating at least two times.

$$\begin{aligned}
\log_{\alpha'}(\alpha' \cdot C_{max}/Trust) &\geq 2 && (4.15)\\
\log_{\alpha'}(\alpha' \cdot C_{max}/Trust) &\geq \log_{\alpha'} \alpha'^2 && (4.16)\\
\alpha' \cdot C_{max}/Trust &\geq \alpha'^2 && (4.17)\\
C_{max}/Trust &\geq \alpha' && (4.18)\\
C_{max}/Trust &\geq \alpha^{1/STD_{i_x}}, \textit{ since } \alpha' \leftarrow \alpha^{1/STD_{i_x}} && (4.19)\\
\log_{\alpha}^{C_{max}/Trust} &\geq \frac{1}{STD_{i_x}} && (4.20)\\
STD_{i_x} &\geq \log_{\alpha} C_{max}/Trust = \log_{C_{max}/Trust} \alpha && (4.21)
\end{aligned}$$

Finally, if $STD_{i_x} < \log_{C_{max}/Trust} \alpha$, we set $STD_{i_x} = \log_{C_{max}/Trust} \alpha$.

As shown in this Sub-section, we control α dynamically according to individual condition for each item. Observing individual item could be an approach remove man-in-the-middle parameter setting.

Chapter 5

Evaluation and Analysis

5.1 Evaluation Metrics

To evaluate the performance of our RobuRec, we use two performance metrics, prediction shift (PS) and hit ratio (HR) in this thesis. PS is the measure of changes in prediction value. Let U_T and I_T be the set of users and items in the test set, respectively. The mathematical form of PS can be stated as follows:

$$PS = \frac{1}{N} \sum_{u \in U_T} \sum_{i \in I_T} |p'_{u,i} - p_{u,i}|, \tag{5.1}$$

where N is the number of rating values in test set, $p'_{u,i}$ is the predicted rating value, and $p_{u,i}$ is the actual value of user u on item i, respectively.

The PS, in Eq. 5.1, is the measure of sensitivity against an attack. The PS shows the higher value if an attack is successful. Additionally, We define HR to measure how much an attack impacts to the RS. Since the PS shows the measure of prediction errors on the RS, it cannot show the target item of Alice is finally recommended to the normal users after the push attack. To capture the effective-

ness of the push attack from Alice, the HR measures the number of target item that appears at the end of K-top recommendation to the normal users. The HR can be defined as follows:

$$HR_{\psi} = \frac{1}{I_T} \sum_{i \in I_T} H_i \times 100, \tag{5.2}$$

where H_i is the indicator function that has the value of 1 if the target item (ψ) is included K-top recommendation, 0 otherwise.

We tested all items in I_T. We implemented K-top recommendation using item-based CF algorithm (namely, *adjusted cosine similarity*) as shown in [91]. The item-based CF tests all items in I_T to return K-recommendations (i.e., K-recommended items will be returned for each item in I_T). Therefore, HR_{Psi} is the mean value after testing I_T.

$$HR_{\Psi} = \frac{1}{|\Psi|} \sum_{\psi \in \Psi} HR_{\psi}, \tag{5.3}$$

where ψ is a target item in Ψ ($\psi \in \Psi$).

The HR_{Ψ} is the mean value when we tested the number of items ($|\Psi|$). We computed K-top recommendation for each item in varying the value of K in 10, 20, 30, 40, and 50 items, then checked whether the target item is appeared in K-recommended items or not. If the recommendation algorithm is robust, the HR shows lower score under the push attack.

5.2 Parameter (α) Study

As stated in Section 4.3, we explore the parameter, α, with empirical analysis in this Section. Note that if α is set, β is also deterministically decided ($\beta = 2-\alpha$). We measured that PSs for each dataset (MovieLens, Epinion and Daum-movie) with different filler sizes (1%, 3%, 5%, 7%, 10%, 20%, 30%, 40%, and 50%) against three different attacks (random, average and bandwagon attack). We finally averaged all PSs and portrayed the results in Fig. 5.1.

The maximum PSs are appeared in $\alpha = 1.3$, $\alpha = 1.4$, $\alpha = 1.4$ from the random, average and bandwagon attack, respectively; The minimum PSs are appeared in $\alpha = 1.9$, $\alpha = 1.6$, $\alpha = 1.6$ from the random, average and bandwagon attack, respectively. On overall average including the three attacks, we noticed the empirical value of α can be set with 1.2 or 1.6 (one of two minimum points in terms of PS).

To set α, we consider *overwhelming condition* in admission control phase. If

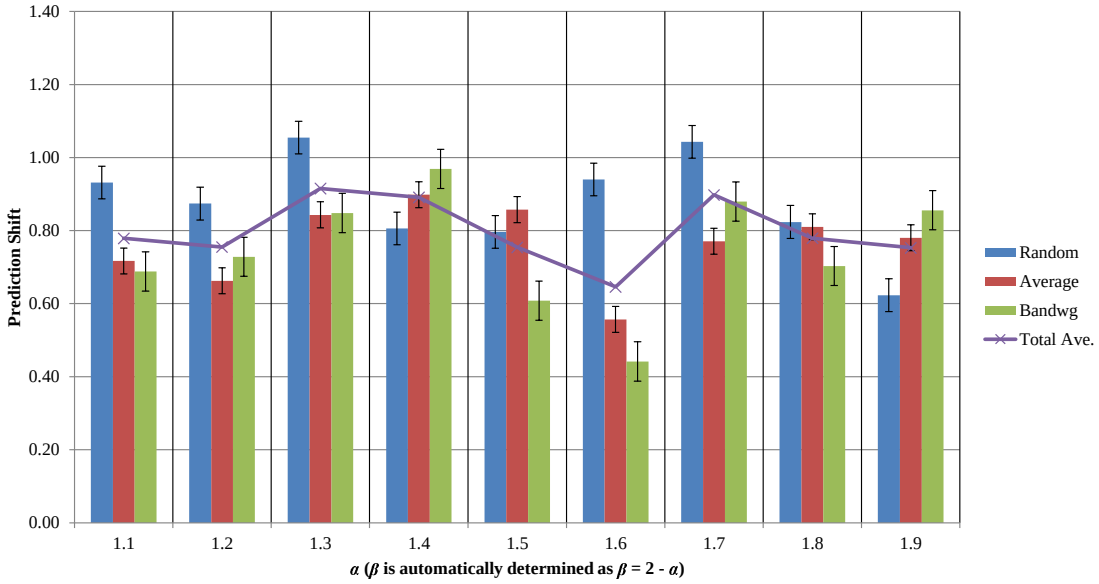


Figure 5.1 Results of varying α with different datasets and attack strategies.

$\alpha > 1.5$, the *Trust* is updated aggressively leading the system condition falls into *overwhelming condition* quickly. Since if RobuRec enters *overwhelming condition*, it does not accept additional ratings, RobuRec fastly converges into *overwhelming condition* may cause the problem that our system gets rid of honest or more required ratings.

To make our system works stable, we select $\alpha = 1.2$. It is worthy noting that finding proper α is trivial task since a system administrator of a RS can figure out α after training their dataset in little amount of time.

5.3 Datasets and Setup

We use three datasets for testing our RobuRec which are the MovieLens[1], Epinion[2], and Daum-movie dataset[3].

Table 5.1 Dataset characteristics

Name	#users	#items	#ratings	rating scale
MovieLens	943	1,682	100,000	$\{1, 2, \ldots, 5\}$
Epinion	1,395	2,590	53,258	$\{1, 2, \ldots, 5\}$
Daum-movie	1,161	4,518	75,493	$\{1, 2, \ldots, 10\}$

The characteristics of the datasets are summarized in Table 5.1. MovieLens is the recommendation web pages to recommend movie items to system users, and the Epinion is a web site that it recommends commercial goods (movies, books, electronics, and so on) to the system users based on the users' reviews. MovieLens contains 943 users and 1,682 items with 100K rating values. Additionally,

[1] http://movielens.umn.edu, the dataset is downloadable from GroupLens Research, http://www.grouplens.org/system/files/ml-100k.zip
[2] http://www.trustlet.org/datasets/downloaded_epinions/ratings_data.txt.bz2
[3] http://movie.daum.net

MovieLens dataset contains the information of the genre. RobuRec uses this genre information when we test the segment attack. Since the MovieLens dataset contains at least 20 rating values for each user, we extract 1,395 users and 2,590 items from the original Epinion dataset which contains 49,290 users and 139,738 items to make datasets be similar experimental environments.

Additionally, we crawled on a famous movie recommendation data from http://movie.daum.net which provides recommendations of various movies in South Korea. It contains 73,060 users, 10,742 items, and 300,629 ratings from January 2005 until December 2011. Since the number of items in the Daum-movie dataset is 2.68 orders of magnitude larger than that of MovieLens dataset and extremely sparse, we filter out users who have less than 30 rating values, and items which have less than 5 rating values. Since the main goal of this paper is to find robust algorithm on the prediction phase, these filtering is still valid for validating the robustness of RobuRec. And we also extract 1,161 users and 4,518 items from the original Daum-movie dataset to make three datasets be similar experimental environments.

Unlike the rating scale of the MovieLens and Epinion dataset which is {1, 2, ..., 5}, Daum-movie dataset has rating scale of {0, 1, 2, ..., 10}. We replaced the zero rating values with ones since RobuRec uses the rating value zero as non-rating. We think the replacement is reasonable since the rating value of zero or one is almost same feeling such that a user dislikes the item (movie). The MovieLens dataset can be tested the performance against the segment attack due to its genre information. However, RobuRec could not test the performance on the segment attack with the Epinion and Daum-movie datasets since they do not have sufficient

genre information.

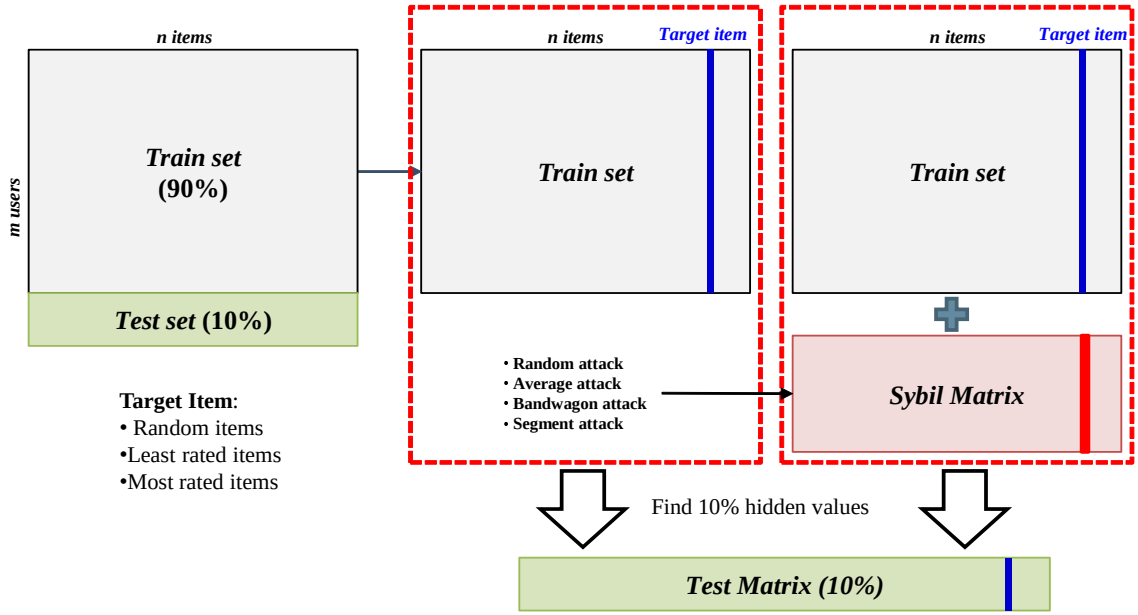


Figure 5.2 Experimental scenarios with four different attack types

To impose attacks on the datasets, we divided dataset into two sub-datasets which are the train and test set in 5.2. For the train set, we randomly select 90% of users and their rating values from original dataset. The rest of original set is set to be a test set. In the attack phase, Alice selects an item as a target of her attack and generates four attacks (random, average, bandwagon, and segment attack) according to the attack scenarios stated in 3. We appended attack profiles onto the train sets.

We vary the attack size (the number of Sybils) 1%, 3%, 5%, 7%, 10%, 20%, 30%, 40%, and 50% of the number of honest users, respectively. Firstly, RobuRec predicts the rating values of the test set which is hidden to RobuRec using Equation 4.7, and it performs the prediction of the test set values in which the test set is now combined with Sybils' ratings, respectively.

The former means the performance of the test set without Sybil attacks, and

the latter is the measure of performance with the Sybil attacks. Since the filler item size can be a major fact of attacks, we measure the prediction performance with varying value of the filler item (in our experiment, we set the five different filler size 10%, 20%, 30%, 40%, and 50% of the number of items, respectively).

We also select target items in three different ways. First, we select 10 items randomly among n items, which can be considered a brute force attack. Second, 10 items which has the least count of ratings among all items. This could be the main goal of the push attack since the push attack pursues boosting the reputation of unpopular item. Finally, mostly rated 10 items selected as target items, which could be considered a situation that an attacker may want to distort the outcome of RS against popular items.

The LTSMF and PCA-based robust RS are proposed in [4,6], respectively. We select those two algorithms to compare to RobuRec since they run on the basic MF approach in which RobuRec uses the baseline method in the prediction phase. We fully implemented PCA-based and LTSMF algorithms, and passed the three datasets onto the algorithms. In the next subsection, we will show the performance of RobuRec compared to PCA-based and LTSMF algorithms.

To evaluate our RobuRec algorithm, the parameters used in our experiment are as follows: $\alpha = 1.2$, $\beta = 0.8$ (automatically determined from α), $C_{max} = 2$, $l = 20$ and $C_{min}= 0.2$. Different parameter settings should be possible and more extensive analysis can be conducted. We leave the impact of the varying parameter evaluation as our future work. The number of the filler items varies in order to monitor the performance of the PS for each algorithm. The number of selected items is set to 10 since an attacker can obtain the 10 popular items reasonably by

browsing or querying the RS.

Table 5.2 PS scores tested against four attacks with three different datasets

Dataset[a]	Algorithm[b]	RobuRec	LTSMF	PCA
MovieLens	random attack	0.80	0.93	1.27
	average attack	0.80	0.94	1.28
	bandwagon attack	0.79	0.94	1.23
	segment attack	0.80	0.96	1.26
	Average	0.80	0.94	1.26
Epinion	random attack	1.03	1.14	1.63
	average attack	0.85	0.86	1.04
	bandwagon attack	0.94	1.21	1.59
	Average	0.94	1.07	1.42
Daum-movie	random attack	0.79	1.23	2.32
	average attack	0.86	1.43	2.64
	bandwagon attack	1.19	1.21	1.59
	Average	0.95	1.29	2.18

[a] Target items are randomly selected, and filler size is set to be 10%.
[b] The PS scores for each attack type and algorithm are averaged.

5.4 Results and Analysis

5.4.1 Performance on PS

Table 5.2 shows the performance of the three robust algorithms in terms of PS values. We did experiment with the attack size (the number of Sybils) of 1%, 3%, 5%, 7%, 10%, 20%, 30%, 40%, and 50% of the number of the honest users, then averaged the PS scores. Note that the segment attack could not be tested in Epinion and Daum-movie dataset since they do not have genre information. However, the PS scores from varying attack size showed same patterns in which RobuRec shows the lowest PS scores over all attack sizes and all datasets. RobuRec keeps the level of the PS low, while LTSMF and PCA show higher PS level than RobuRec. The performance of PS in MovieLens dataset is portrayed in Fig. 5.3∼

Table 5.3 The performance (PS score) comparison of varying filler size with MovieLens dataset

	Algorithm		RobuRec	LTSMF	PCA
MovieLens	random attack	10%	0.92	1.17	1.68
		20%	0.99	1.13	1.75
		30%	0.97	1.25	1.59
		40%	0.94	1.06	1.49
		50%	1.03	1.17	1.53
	average attack	10%	0.91	1.50	1.17
		20%	1.01	1.40	1.09
		30%	0.87	1.11	1.53
		40%	0.94	1.10	1.40
		50%	0.92	1.10	1.42
	bandwagon attack	10%	1.13	1.36	1.78
		20%	0.92	1.18	1.64
		30%	0.86	1.12	1.41
		40%	0.65	0.85	1.16
		50%	1.06	1.27	1.73
	segment attack	10%	0.99	1.27	1.63
		20%	0.68	0.85	1.19
		30%	0.68	0.81	1.02
		40%	0.96	1.18	1.46
		50%	1.02	1.14	1.52

Table 5.4 The performance (PS score) comparison of varying filler size with Epinion and Daum-movie datasets

	Algorithm		RobuRec	LTSMF	PCA
Epinion	random attack	10%	1.04	1.47	1.68
		20%	1.04	1.16	1.67
		30%	1.09	1.38	1.76
		40%	1.08	1.33	1.72
		50%	1.07	1.44	1.73
	average attack	10%	0.91	1.31	1.84
		20%	0.71	0.81	0.99
		30%	0.90	1.25	1.63
		40%	1.12	1.27	1.61
		50%	1.04	1.32	1.70
	bandwagon attack	10%	0.94	0.93	1.50
		20%	0.92	0.94	1.54
		30%	1.10	0.94	1.76
		40%	0.97	0.94	1.61
		50%	0.98	0.94	1.61
Daum-movie	random attack	10%	0.79	1.23	2.32
		20%	0.55	0.75	1.28
		30%	0.91	1.31	2.15
		40%	1.15	1.34	2.23
		50%	1.04	1.48	2.12
	average attack	10%	1.19	1.60	3.02
		20%	1.09	1.68	2.65
		30%	0.99	2.07	2.75
		40%	0.96	1.44	2.41
		50%	0.91	1.60	2.39
	bandwagon attack	10%	0.86	1.43	2.64
		20%	0.96	1.45	2.71
		30%	1.37	2.10	3.29
		40%	0.50	0.93	1.42
		50%	1.38	2.12	3.26

Table 5.5 Averaged PS scores from three different targets

Algorithm	RobuRec	LTSMF	PCA
Random targets	0.83	1.20	1.84
Most rated targets	1.16	1.35	2.60
Least rated targets	1.05	1.70	3.20

Fig. 5.5. For detailed enhancement, RobuRec can reduce PS score by 21% lower than that of LTSMF and 49% lower than that of PCA on average.

In case of PCA-based approach, the PS level is higher than those of RobuRec and LTSMF when the attack size grows up. The PS of the PCA-based approach shows the worst performance and does not working well in a RS showing the almost PS s are greater than 1.0. For example, let the actual rating value be 3.0, if the predicted value is 4.5 because of its poor prediction performance, a user tends to judge the item is good even though its reputation is staying in the middle level of recommendation. The PS which is greater than 1.0 leads mis-understanding on the item easily. The PS values in Daum-movie dataset also shows much higher except RobuRec. Since the Daum-movie dataset uses the rating range of {1, 2, ... , 10}, the PS scores of Daum-movie dataset easily grows up more than those of the range of {1, 2, ... , 5}. In overall attack strategies regardless attack size and the datasets, RobuRec shows the best performance in terms of PS scores which implies that RobuRec is the most robust (least sensitive) algorithm against four possible push attacks.

5.4.2 Impact of Filler Size

The filler size can be a major factor of an attacker since the filler size is manageable from an attacker. To investigate the impact of filler size, we experiment

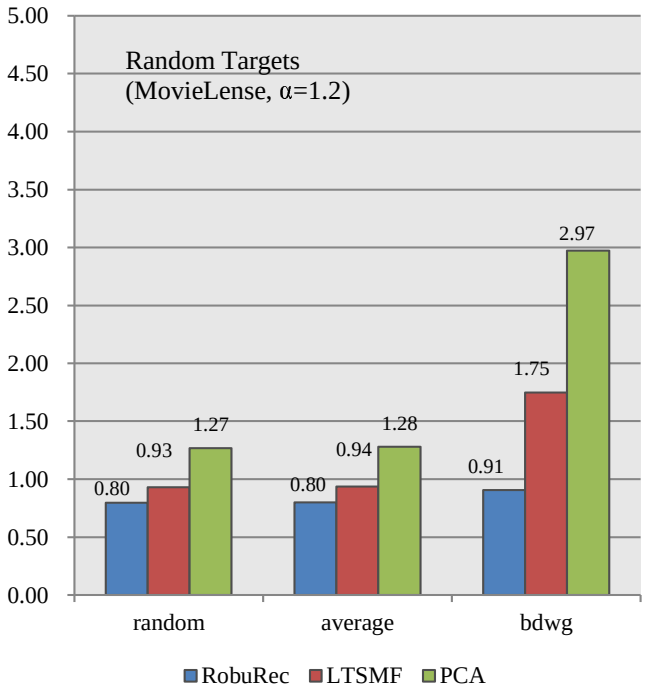


Figure 5.3 PS score comparison against random target selection

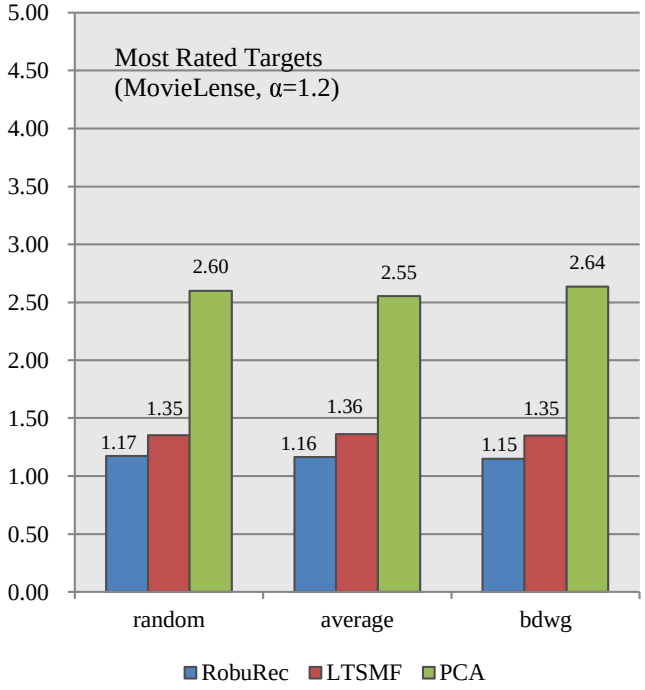


Figure 5.4 PS score comparison against most rated target selection

the three algorithms with variable filler sizes in the three datasets. We varied the number of filler items in the range of 10%, 20%, 30%, 40%, and 50% of the total number of items I_T. The filler items of each filler size are selected randomly from I_T. Table 5.3 and 5.4 show the PS against varying filler size. Note that Table 5.3 contains the PS results of segment attacks since MovieLens dataset has genre information; Table 5.4 does not. We select 10 random items in each attack size 1%, 3%, 5%, 7%, 10%, 20%, 30%, 40%, and 50% at each filler size (totally, 90 target items are randomly chosen in each stage of the filler size test). The averaged PS values are recorded in Table 5.5.

As shown in Table 5.5, RobuRec shows the lower PS values compared to LTSMF and PCA-based schemes. In the case of bandwagon attack, we find that the three algorithms behave differently at the random and average attack. The PS values of

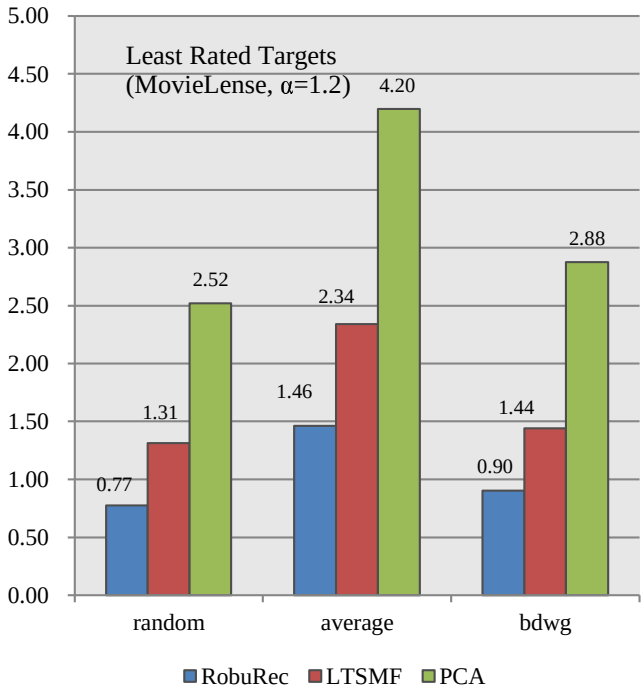


Figure 5.5 PS score comparison against least rated target selection

the bandwagon attack show the low values in the middle size of filler size (40% in our experiment) in MovieLens and Daum-movie dataset while the Epinion dataset does not show significant changes. The bandwagon attack has the selective items to increase the impact of the attack, while the random and average attack does not. We infer the difference comes from the characteristics of the attacks and datasets. We will leave the more detailed analysis of the PS dependencies according to the attack strategies as our future work.

5.4.3 Impact of Target Selection Strategy

As stated in Section 3.3, Alice can choose different target item(s) according to her attack goals. We evaluated three different target selection strategies (random, most rated, and least rated targets). The random targets show the lower PS scores compared to those of the most and least rated targets.

Among the attack types against the random targets, the bandwagon attack is the most efficient attack method. Three robust algorithms show the stable PS scores against three attacks in the most rated targets. PCA scheme is the weakest for the three attacks which is the same result in the random targets. The least rated targets are the weakest targets since PS scores of the targets is higher than the random and most rated targets on average except for RobuRec (refer to the Table 5.5 for more details). Especially, the average attack is the most powerful with the least rated targets.

The least rated targets were clearly considered to be unpopular items, which can be potential targets of Alice. In the attacker's position, launching the average attack onto the least rated targets is one of the best strategies. In opposite side

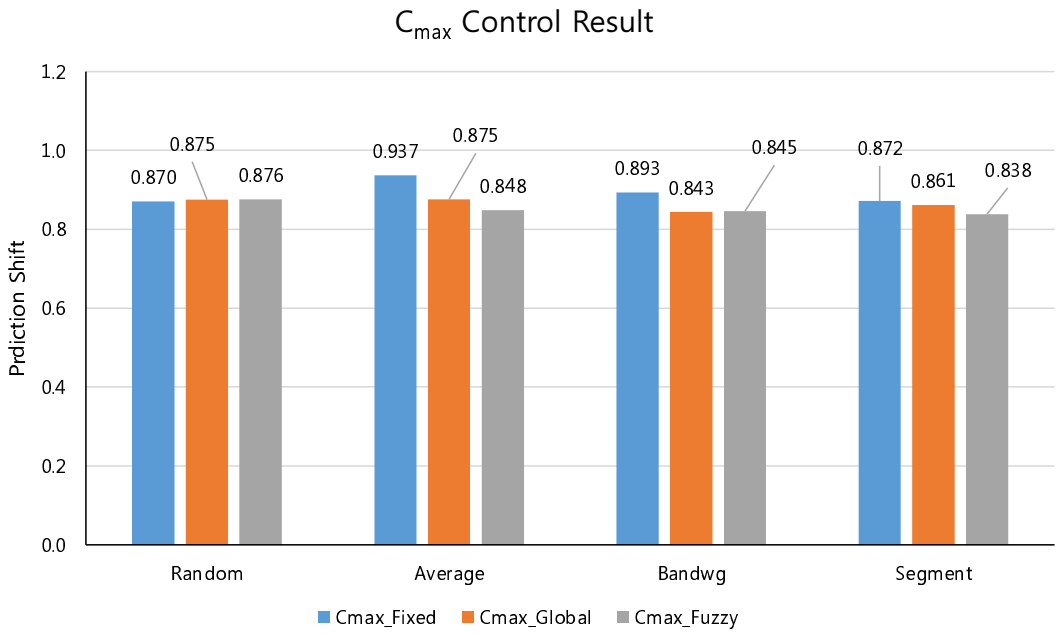


Figure 5.6 C_{max} control results against four different attack types.

(defenders' point), one should pay more attention or needs to carefully design his defense algorithm to be robust against the average attack. Also, the bandwagon attack works commonly on the three target selection strategies. We infer that the bandwagon attack works efficiently since it considers the popularity of items in I_T. Note that RobuRec shows the lowest PS level against possible attacks and target selection strategies.

5.4.4 Dynamic Parameter Control

Dynamic C_{max} Control We elaborated how to remove manual parameter setting in Sub-section 4.5.2. Firstly, We tested the dynamic parameter control on C_{max} and α, and report the results in Fig 5.6. The C_{max_fixed} approach represents there is no control during admission control phase. The C_{max_fixed} uses $C_{max} = 2.0$ as stated in Section 5.3. The C_{max_global} uses C_{max} is configured using Eq. 4.10, where we set $\Upsilon = 2.0$ and configure STD_{global} from the value of observation on

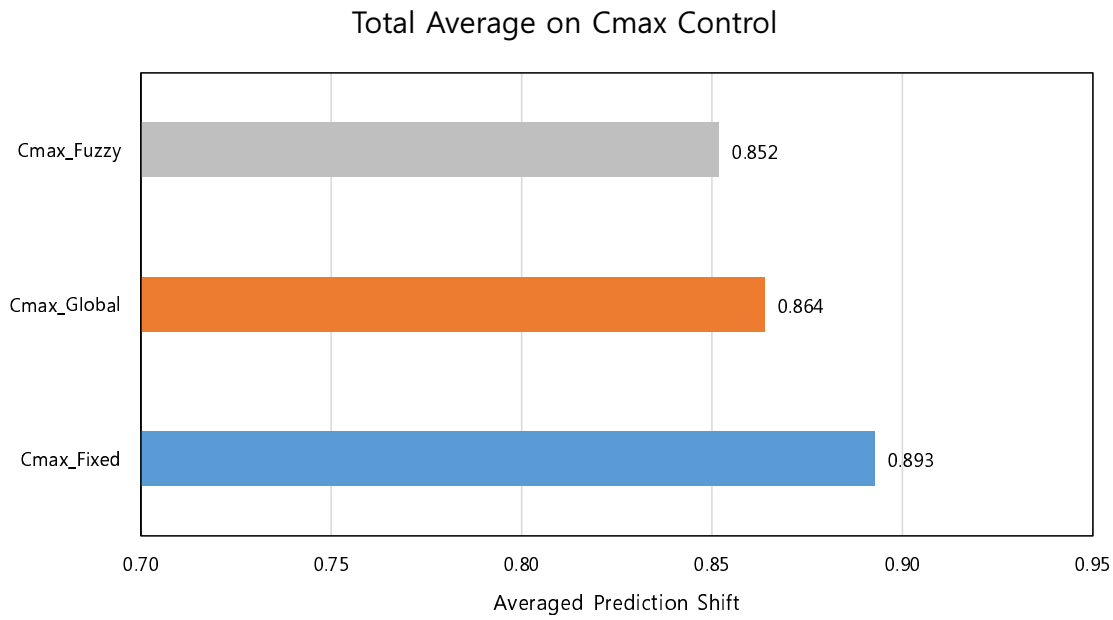


Figure 5.7 Total average of C_{max} control results.

$R_{m\times n}$. The C_{max_fuzzy} uses the FR-based C_{max} control based on decision rules, membership functions and defuzzification as we designed in Section 4.5. We tested the C_{max_fixed} and C_{max_global} in the MovieLens dataset against four possible attack types. For considering C_{max_fixed} as a baseline scheme, As can be seen in Fig 5.6, the C_{max_fuzzy} shows the lowest PS except the random attack, and the C_{max_global} shows lower PS than C_{max_fixed} but higher than that of C_{max_fuzzy}. We conjecture that dynamic C_{max} control is not effective under the random attack since the random rating according to the global mean and STD countervail the detection enhancement from dynamic control. However, the differences between the three schemes under random attack are negligible (PS difference is on 0.006). We report the total averaged result of C_{max} control in Fig. 5.7. As can be seen in Fig. 5.7, C_{max_fuzzy} shows the lowest PS, and C_{max_global} is the middle PS. We conclude that the dynamic C_{max} control reflecting a status of a RS is more effective. Also, if

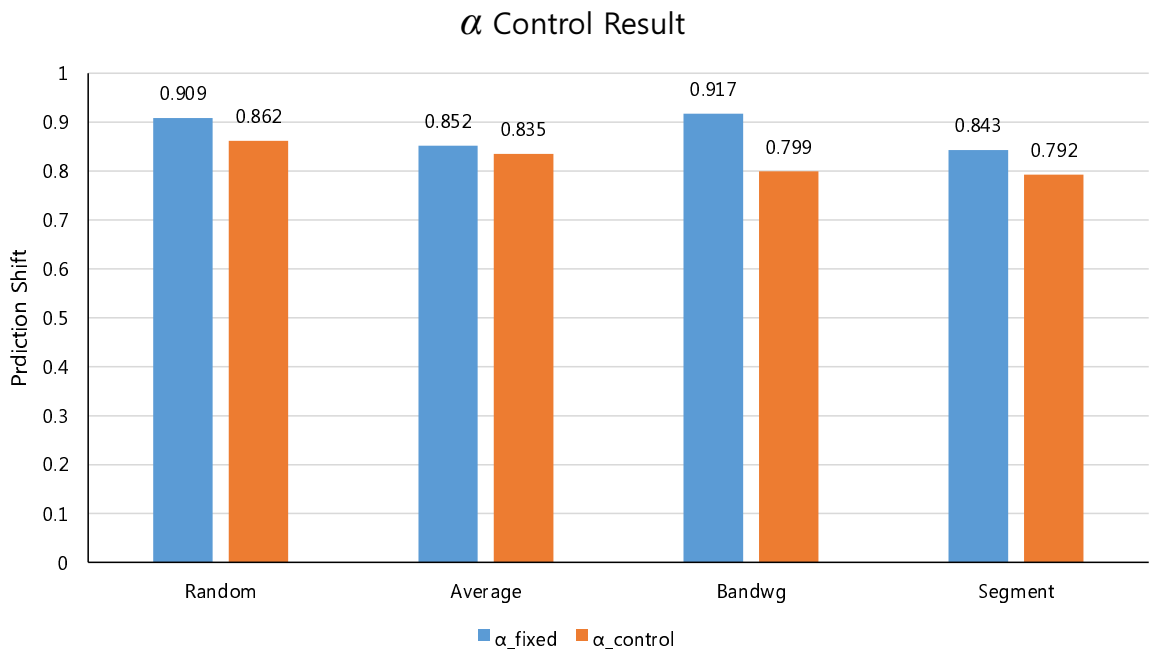


Figure 5.8 α control results against four different attack types.

we can control C_{max} more carefully, we can achieve more robust recommendation with low PS.

Dynamic α Control In this paragraph, we report the result of controlling the parameter, α based on individual item's condition. We implemented the scheme $\alpha_{control}$ based on the updating rule (i.e., Eq. 4.14). On the other hand, for considering α_{fixed} as a baseline scheme, we fix the $\alpha = 1.2$ as we stated in Section 5.3. As can be seen in Fig 5.8, the $\alpha_{control}$ shows lowest PS than α_{fixed} over all possible attack types. The individual and dynamic α control can reduce the PS and make RobuRec more robust. We report the total averaged results in Fig. 5.9. As can be seen in Fig. 5.9, the $\alpha_{control}$ can reduce 7% compared to α_{fixed} in terms of PS, which is better than dynamic C_{max_fuzzy} scheme. We note that $\alpha_{control}$ by updating parameter accordingly can work against four possible attack types, while C_{max_fuzzy} is not effective against the random attack. We partially conclude that

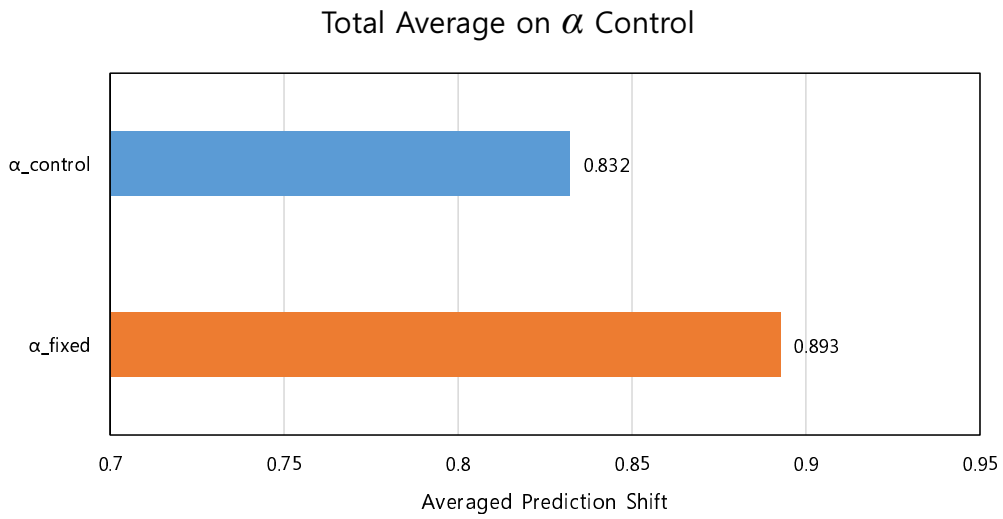


Figure 5.9 Total average of α control results.

the controlling parameter α is more effective way than C_{max} to reduce PS.

5.4.5 Performance on HR

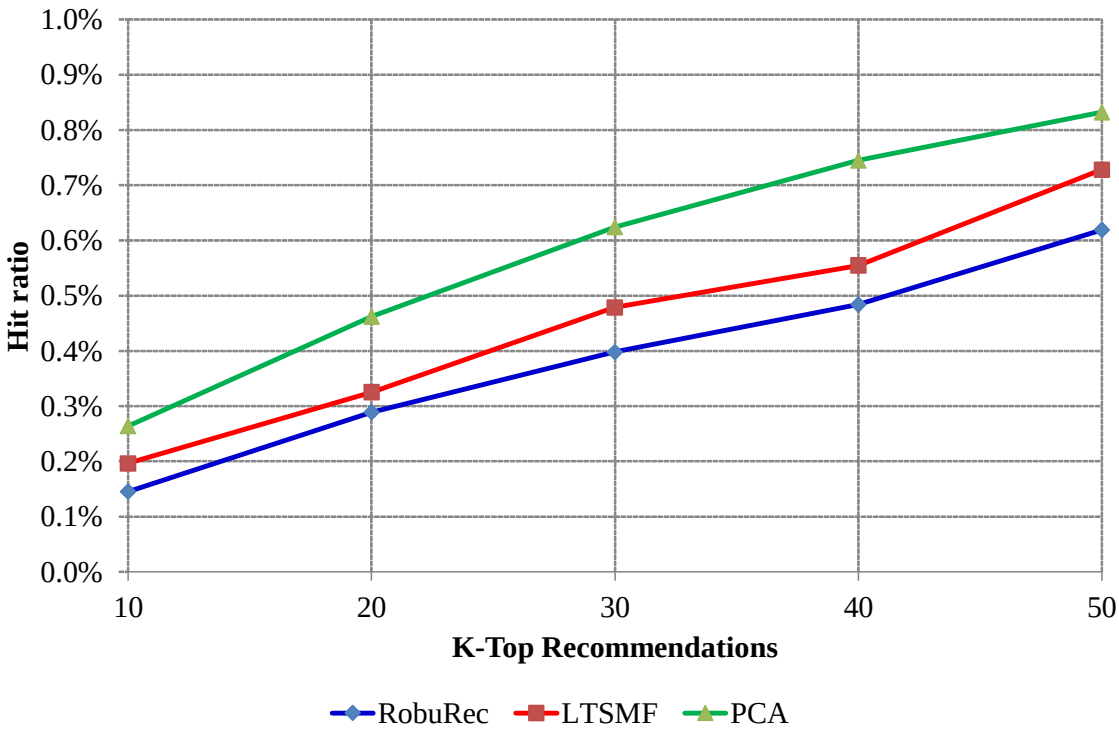


Figure 5.10 HR comparison of the three algorithms

Now we explore the algorithms how robust algorithm works to the system users in terms of HR. The three algorithms compute similarity between items and return K items as a recommendation to the users using Equation 5.2. We simulated with 50 target items (i.e., $|\Psi|$ in Equation 5.3), then averaged the HRs. Fig. 5.10 shows the fraction of targeted item to be included in K-top recommendation.

As K grows up, the HR increases linearly. This result is not surprising since the growth of K value means that the target item has more possibility to be included in recommendation items. The three algorithms work efficiently with low HR values in which all the HR values are smaller than 1% since the algorithms are focused on the goal of being robust to the Sybils' activity in RS. However, one that we notice is that RobuRec showed the best performance over LTSMF and PCA-based algorithm. As we can see in Fig. 5.10, the HR shows 15% and 34% lower than that of PCA and LTSMF schemes on average, respectively. Therefore, RobuRec is robust with low sensitivity of PS as well as HR.

5.4.6 Analysis on Escaping Probability

In this sub section, we discuss the probability of escaping users. We explore how the honest users' opinions (rating information) can be preserved even RobuRec rejects the ratings in the admission control phase.

RobuRec can limit the number of Sybil admissions on a RS. The *Agreement* is computed using Eq. 5.4 as portrayed in the middle of Fig. 5.11.

$$\overline{C}^{h}_{all} = \frac{\sum_{i=1}^{m} C^{h}_{i}}{m} \tag{5.4}$$

In normal condition with honest users, RobuRec can admit almost all honest

users' ratings, since the honest users gives their ratings following on their own experiences. This lead the ratings from honest users to be accepted mostly.

However, the honest users' admissions also possibly limited during the admission control phase. We explore the probability that *"what is the probability that k users' ratings are not reflected on item i?"*

Let us define *escaping users* such that the number of users who rated on item i at time t, but do not have $\boldsymbol{R}$ (tagged as *reliable* (i.e., the number of users gave ratings on item i but not included into the prediction phase))

As shown in Figure 5.11, escaping users can be appeared in three two cases as follows:

- In the case that a user's *Trust* is equal to C_{max} or lower than C_{max}, if the user

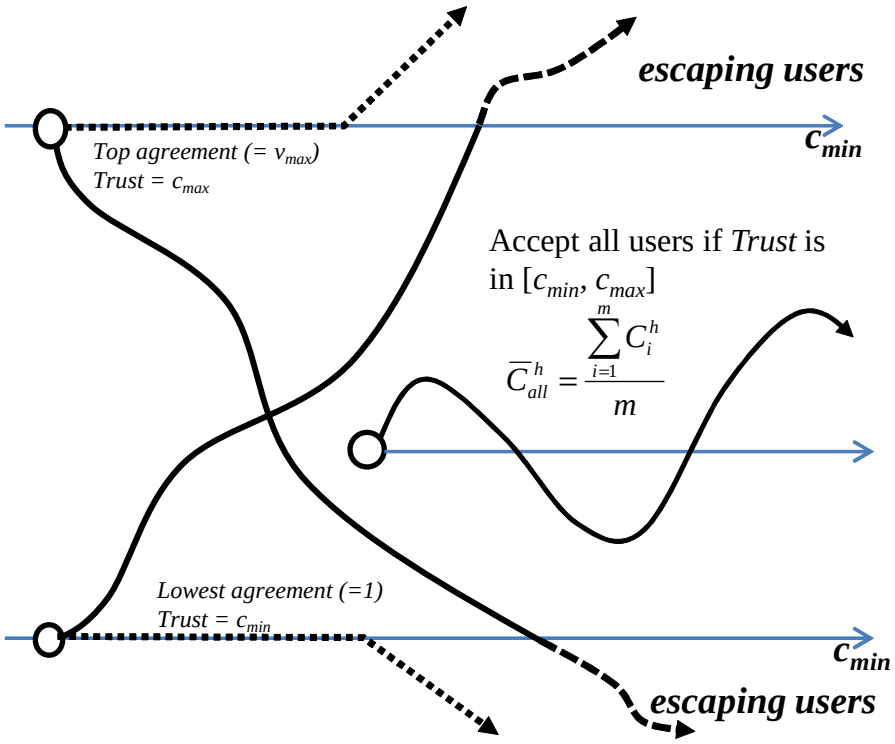


Figure 5.11 The concept of the escaping probability

keep give positive ratings (we do not consider the user is Sybil or not), the user's rating keeps being denied by RobuRec's admission control mechanism.

- In the case that a user's *Trust* is equal to C_{min} or lower than C_{min}, if the user keep give negative ratings, the user's rating keeps being also denied by RobuRec's admission control mechanism.

We only consider the case that a user gives positive ratings repeatedly (note that the negative rating case can be considered as same to the positive case). Let us consider the case that a user recommends items under non-overwhelming condition. After the item is recommended with a positive rating, RobuRec will increase the $Trust$. At this time, the $Trust$ is multiplied with the parameter α. Otherwise, the $Trust$ value can stay in the value of C_{max}. Therefore, the $Trust$ can be multiplied at most $\lceil \log_\alpha (\alpha \cdot \ C_{max}/\ Trust) \rceil$ times. The probability that the user's new rating on that item i is $1/V_{max}$. The probability that a rating value is greater than $Agreement$ is $p_{agreement} = 1 - Agreement/v_{max}$, where $Agreement \in \{1, 2, ..., v_{max}\}$ and $1/v_{max} \leq p_{agreement} \leq 1$.

Now, we compute the escaping probability. Let p_e^k be the probability such that k ratings from the escaping users are not reflected on item i. We define p_e^k as followed Eq. 5.5.

$$p_e^k = Pr\{X = k\} \triangleq p_{agreement}^{\lceil \log_\alpha (\alpha \cdot \ C_{max}/Trust) \rceil + k} \cdot (1 - p_{agreement}) \tag{5.5}$$

In Eq. 5.5, $(1 - p_{agreement})$ is small constant. Also, α, c_{max} and $Trust$ is given fixed value during admission control phase. Therefore, the escaping probability can be represented as Eq. 5.6.

$$p_e^k = {p_{agreement}}^{\Omega}, \tag{5.6}$$

$$where\ \Omega = \lceil \log_{\alpha} (\alpha \cdot\ C_{max}/Trust) \rceil + k.$$

In Eq. 5.6, p_e^k depends on k. Moreover, $p_{agreement}$ is less than 1.0, unless otherwise $p_{agreement} = 1.0$ that happens with low probability such that all ratings are given with V_{max} implying strongly Sybil's ratings. Therefore, p_e^k is quickly converges to 0.0 as the value of k is increased. Finally, we conclude that the escaping users' impact in our approach can be neglected with small or no effect.

Chapter 6

Conclusion

In this thesis, we design a novel robust recommendation system named RobuRec that supports general ratings for RS users. The distinctive feature of RobuRec such as admission control with information level makes it possible to predict recommendation results without identifying Sybil's profiles in a RS matrix. Through the extensive evaluations using three datasets (two publicly obtainable datasets and one real world dataset we crawled), we demonstrated that RobuRec works robustly against Sybil attacks in RSs compared to other approaches such as PCA and LTSMF in terms of PS, filler size, target selection strategies, and HR. The experimental results confirmed our approach performs robustly and reliably in all possible settings against various attack strategies. Furthermore, we believe that because RobuRec insists on no Sybil's profile in a RS matrix, it can be easily deployed in existing RSs and the benefits from this flexible adoption could be immediately achievable. RobuRec is an important step to realize the potential of information level in RSs because it presents a new axis that can be manipulated to extract more robustness. Even we analyzed the parameter such as α, the impact

of varying system parameters (e.g., C_{max}, C_{min}) should be explored with extensive experiment. We believe we tested the representative datasets. However, the characteristics of the datasets remain open and should be analyzed and inspected in our future work.

Bibliography

[1] J. Douceur, "The sybil attack," in *Peer-to-Peer Systems* (P. Druschel, F. Kaashoek, and A. Rowstron, eds.), vol. 2429 of *Lecture Notes in Computer Science*, pp. 251–260, Springer Berlin Heidelberg, 2002.

[2] N. J. Hurley, "Robustness of recommender systems," in *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, (New York, NY, USA), pp. 9–10, ACM, 2011.

[3] H. Yu, C. Shi, M. Kaminsky, P. Gibbons, and F. Xiao, "Dsybil: Optimal sybil-resistance for recommendation systems," in *Security and Privacy, 2009 30th IEEE Symposium on*, pp. 283–298, May 2009.

[4] Z. Cheng and N. Hurley, "Robust collaborative recommendation by least trimmed squares matrix factorization," in *Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on*, vol. 2, pp. 105–112, Oct 2010.

[5] B. Mehta, T. Hofmann, and P. Fankhauser, "Lies and propaganda: Detecting spam users in collaborative filtering," in *Proceedings of the 12th International*

Conference on Intelligent User Interfaces, IUI '07, (New York, NY, USA), pp. 14–21, ACM, 2007.

[6] B. Mehta and W. Nejdl, "Attack resistant collaborative filtering," in *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, (New York, NY, USA), pp. 75–82, ACM, 2008.

[7] B. Mobasher, R. Burke, and J. J. Sandvig, "Model-based collaborative filtering as a defense against profile injection attacks," in *AAAI*, vol. 6, p. 1388, 2006.

[8] R. Burke, "Hybrid web recommender systems," in *The adaptive web*, pp. 377–408, Springer, 2007.

[9] T. Mahmood and F. Ricci, "Improving recommender systems with adaptive conversational strategies," in *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pp. 73–82, ACM, 2009.

[10] P. Resnick and H. R. Varian, "Recommender systems," *Communications of the ACM*, vol. 40, no. 3, pp. 56–58, 1997.

[11] P. B. Kantor, L. Rokach, F. Ricci, and B. Shapira, *Recommender systems handbook*. Springer, 2011.

[12] F. Ricci, B. Arslan, N. Mirzadeh, and A. Venturini, "Itr: a case-based travel advisory system," in *Advances in Case-Based Reasoning*, pp. 613–627, Springer, 2002.

[13] D. Bridge, M. H. Göker, L. McGinty, and B. Smyth, "Case-based recommender systems," *The Knowledge Engineering Review*, vol. 20, no. 03, pp. 315–320, 2005.

[14] A. Felfernig, G. Friedrich, D. Jannach, M. Stumptner, *et al.*, "Consistency-based diagnosis of configuration knowledge bases," in *ECAI*, pp. 146–150, 2000.

[15] A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker, "An integrated environment for the development of knowledge-based recommender applications," *International Journal of Electronic Commerce*, vol. 11, no. 2, pp. 11–34, 2006.

[16] A. Felfernig and R. Burke, "Constraint-based recommender systems: technologies and research issues," in *Proceedings of the 10th international conference on Electronic commerce*, p. 3, ACM, 2008.

[17] J. Golbeck, *Generating predictive movie recommendations from trust in social networks*. Springer, 2006.

[18] G. Groh and C. Ehmig, "Recommendations in taste related domains: collaborative filtering vs. social filtering," in *Proceedings of the 2007 international ACM conference on Supporting group work*, pp. 127–136, ACM, 2007.

[19] I. Guy, N. Zwerdling, D. Carmel, I. Ronen, E. Uziel, S. Yogev, and S. Ofek-Koifman, "Personalized recommendation of social software items based on social relations," in *Proceedings of the third ACM conference on Recommender systems*, pp. 53–60, ACM, 2009.

[20] P. Massa and P. Avesani, "Trust-aware collaborative filtering for recommender systems," in *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, pp. 492–508, Springer, 2004.

[21] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 6, pp. 734–749, 2005.

[22] P. Melville and V. Sindhwani, "Recommender systems," in *Encyclopedia of machine learning*, pp. 829–838, Springer, 2010.

[23] J. Delgado and N. Ishii, "Memory-based weighted majority prediction," in *SIGIR Workshop Recomm. Syst. Citeseer*, Citeseer, 1999.

[24] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 5–53, 2004.

[25] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "Grouplens: applying collaborative filtering to usenet news," *Communications of the ACM*, vol. 40, no. 3, pp. 77–87, 1997.

[26] A. Nakamura and N. Abe, "Collaborative filtering using weighted majority prediction algorithms.," in *ICML*, vol. 98, pp. 395–403, 1998.

[27] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: an open architecture for collaborative filtering of netnews," in *Proceedings of the*

1994 ACM conference on Computer supported cooperative work, pp. 175–186, ACM, 1994.

[28] U. Shardanand and P. Maes, "Social information filtering: algorithms for automating "word of mouth"," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 210–217, ACM Press/Addison-Wesley Publishing Co., 1995.

[29] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 143–177, 2004.

[30] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[31] R. M. Bell and Y. Koren, "Scalable collaborative filtering with jointly derived neighborhood interpolation weights," in *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pp. 43–52, IEEE, 2007.

[32] R. Bell, Y. Koren, and C. Volinsky, "Modeling relationships at multiple scales to improve accuracy of large recommender systems," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 95–104, ACM, 2007.

[33] T. Hofmann and D. Hartmann, "Collaborative filtering with privacy via factor analysis," in *Proceedings of the 2005 ACM Symposium on Applied Computing*, pp. 791–795, 2005.

[34] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 426–434, ACM, 2008.

[35] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *Proceedings of KDD cup and workshop*, vol. 2007, pp. 5–8, 2007.

[36] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization.," in *NIPS*, vol. 1, pp. 2–1, 2007.

[37] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Matrix factorization and neighbor based algorithms for the netflix prize problem," in *Proceedings of the 2008 ACM conference on Recommender systems*, pp. 267–274, ACM, 2008.

[38] K. Lang, "Newsweeder: Learning to filter netnews," in *in Proceedings of the 12th International Machine Learning Conference (ML95*, 1995.

[39] R. Baeza-Yates, B. Ribeiro-Neto, *et al.*, *Modern information retrieval*, vol. 463. ACM press New York, 1999.

[40] R. C. Holte and J. N. Y. Yan, "Inferring what a user is not interested in," in *Advances in Artifical Intelligence*, pp. 159–171, Springer, 1996.

[41] J. J. Rocchio, "Relevance feedback in information retrieval," 1971.

[42] G. Chowdhury, *Introduction to modern information retrieval*. Facet publishing, 2010.

[43] P. Melville, R. J. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," in *AAAI/IAAI*, pp. 187–192, 2002.

[44] R. Burke, "Hybrid recommender systems: Survey and experiments," *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[45] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin, "Combining content-based and collaborative filters in an online newspaper," in *Proceedings of ACM SIGIR workshop on recommender systems*, vol. 60, Citeseer, 1999.

[46] C. Basu, H. Hirsh, W. Cohen, *et al.*, "Recommendation as classification: Using social and content-based information in recommendation," in *AAAI/IAAI*, pp. 714–720, 1998.

[47] R. J. Mooney and L. Roy, "Content-based book recommending using learning for text categorization," in *Proceedings of the fifth ACM conference on Digital libraries*, pp. 195–204, ACM, 2000.

[48] D. C. Wilson, B. Smyth, and D. O. Sullivan, "Sparsity reduction in collaborative recommendation: A case-based approach," *International journal of pattern recognition and artificial intelligence*, vol. 17, no. 05, pp. 863–884, 2003.

[49] R. Andersen and K. J. Lang, "Communities from seed sets," in *Proceedings of the 15th international conference on World Wide Web*, pp. 223–232, ACM, 2006.

[50] J. P. Bagrow, "Evaluating local community methods in networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 05, p. P05001, 2008.

[51] A. Clauset, "Finding local community structure in networks," *Physical review E*, vol. 72, no. 2, p. 026132, 2005.

[52] A. Mohaisen, A. Yun, and Y. Kim, "Measuring the mixing time of social graphs," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pp. 383–389, ACM, 2010.

[53] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "Sybilguard: defending against sybil attacks via social networks," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 267–278, 2006.

[54] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, "Sybillimit: A near-optimal social network defense against sybil attacks," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pp. 3–17, IEEE, 2008.

[55] L. Shi, S. Yu, W. Lou, and Y. T. Hou, "Sybilshield: An agent-aided social network-based sybil defense among multiple communities," in *INFOCOM, 2013 Proceedings IEEE*, pp. 1034–1042, IEEE, 2013.

[56] J. Xue, Z. Yang, X. Yang, X. Wang, L. Chen, and Y. Dai, "Votetrust: Leveraging friend invitation graph to defend against social network sybils," in *INFOCOM, 2013 Proceedings IEEE*, pp. 2400–2408, IEEE, 2013.

[57] N. Hadlee and S. Kayalvizhi, "Increasing sybil attack detection probability in open-access distributed systems," in *Recent Trends in Information Technology (ICRTIT), 2011 International Conference on*, pp. 1137–1142, IEEE, 2011.

[58] Y. Boshmaf, K. Beznosov, and M. Ripeanu, "Graph-based sybil detection in social and information systems," in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 466–473, ACM, 2013.

[59] G. Danezis and P. Mittal, "Sybilinfer: Detecting sybil nodes using social networks.," in *NDSS*, 2009.

[60] D. N. Tran, B. Min, J. Li, and L. Subramanian, "Sybil-resilient online content voting.," in *NSDI*, vol. 9, pp. 15–28, 2009.

[61] N. Tran, J. Li, L. Subramanian, and S. S. Chow, "Optimal sybil-resilient node admission control," in *INFOCOM, 2011 Proceedings IEEE*, pp. 3218–3226, IEEE, 2011.

[62] W. Wei, F. Xu, C. C. Tan, and Q. Li, "Sybildefender: Defend against sybil attacks in large social networks," in *INFOCOM, 2012 Proceedings IEEE*, pp. 1951–1959, IEEE, 2012.

[63] E. Aïmeur, G. Brassard, J. M. Fernandez, and F. S. M. Onana, "Alambic: a privacy-preserving recommender system for electronic commerce," *International Journal of Information Security*, vol. 7, no. 5, pp. 307–334, 2008.

[64] S. Berkovsky, N. Borisov, Y. Eytani, T. Kuflik, and F. Ricci, "Examining users' attitude towards privacy preserving collaborative filtering," *Proceedings of DM. UM*, vol. 7, 2007.

[65] S. Berkovsky, Y. Eytani, T. Kuflik, and F. Ricci, "Enhancing privacy and preserving accuracy of a distributed collaborative filtering," in *Proceedings of the 2007 ACM conference on Recommender systems*, pp. 9–16, ACM, 2007.

[66] J. Canny, "Collaborative filtering with privacy," in *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pp. 45–57, IEEE, 2002.

[67] Z. Cheng and N. Hurley, "Effective diverse and obfuscated attacks on model-based recommender systems," in *Proceedings of the third ACM conference on Recommender systems*, pp. 141–148, ACM, 2009.

[68] A. Kobsa, "Privacy-enhanced personalization," *Communications of the ACM*, vol. 50, no. 8, pp. 24–33, 2007.

[69] K. Shyong, D. Frankowski, J. Riedl, *et al.*, "Do you trust your recommendations? an exploration of security and privacy issues in recommender systems," in *Emerging Trends in Information and Communication Security*, pp. 14–29, Springer, 2006.

[70] N. Ramakrishnan, B. J. Keller, B. J. Mirza, A. Y. Grama, and G. Karypis, "When being weak is brave: privacy issues in recommender systems," *Technical paper posted on the Computing Research Repository at http://xxx. lanl. gov/abs/cs. CG/0105028*, 2001.

[71] R. Shokri, P. Pedarsani, G. Theodorakopoulos, and J.-P. Hubaux, "Preserving privacy in collaborative filtering through distributed aggregation of offline profiles," in *Proceedings of the third ACM conference on Recommender systems*, pp. 157–164, ACM, 2009.

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

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

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

[75] R. Burke, B. Mobasher, R. Zabicki, and R. Bhaumik, "Identifying attack models for secure recommendation," in *Beyond Personalization: A Workshop on the Next Generation of Recommender Systems*, 2005.

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

[77] Z. Fu-guo and X. Sheng-hua, "Analysis of trust-based e-commerce recommender systems under recommendation attacks," in *Data, Privacy, and E-Commerce, 2007. ISDPE 2007. The First International Symposium on*, pp. 385–390, IEEE, 2007.

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

[79] P. Massa and P. Avesani, "Trust-aware recommender systems," in *Proceedings of the 2007 ACM conference on Recommender systems*, pp. 17–24, ACM, 2007.

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

[81] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, "Effective attack models for shilling item-based collaborative filtering systems," in *Proceedings of the 2005 WebKDD Workshop, held in conjuction with ACM SIGKDD'2005*, 2005.

[82] M. P. O'Mahony, N. J. Hurley, and G. C. Silvestre, "Promoting recommendations: An attack on collaborative filtering," in *Database and Expert Systems Applications*, pp. 494–503, Springer, 2002.

[83] M. P. O'Mahony, N. J. Hurley, and G. C. Silvestre, "An evaluation of the performance of collaborative filtering," in *14th Irish Artificial Intelligence and Cognitive Science (AICS 2003) Conference*, Citeseer, 2003.

[84] J. J. Sandvig, B. Mobasher, and R. Burke, "Robustness of collaborative recommendation based on association rule mining," in *Proceedings of the 2007 ACM conference on Recommender systems*, pp. 105–112, ACM, 2007.

[85] X.-F. Su, H.-J. Zeng, and Z. Chen, "Finding group shilling in recommendation system," in *Special interest tracks and posters of the 14th international conference on World Wide Web*, pp. 960–961, ACM, 2005.

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

[87] B. Van Roy and X. Yan, "Manipulation-resistant collaborative filtering systems," in *Proceedings of the third ACM conference on Recommender systems*, pp. 165–172, ACM, 2009.

[88] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, "Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness," *ACM Transactions on Internet Technology (TOIT)*, vol. 7, no. 4, p. 23, 2007.

[89] B. Marlin, R. S. Zemel, S. Roweis, and M. Slaney, "Collaborative filtering and the missing at random assumption," *arXiv preprint arXiv:1206.5267*, 2012.

[90] J. D. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *Proceedings of the 22nd international conference on Machine learning*, pp. 713–719, ACM, 2005.

[91] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *Proceedings of the fourth ACM international conference on Web search and data mining*, pp. 287–296, ACM, 2011.

[92] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *Proceedings of the 25th international conference on Machine learning*, pp. 880–887, ACM, 2008.

[93] K. Yu, S. Zhu, J. Lafferty, and Y. Gong, "Fast nonparametric matrix factorization for large-scale collaborative filtering," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pp. 211–218, ACM, 2009.

[94] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische Mathematik*, vol. 14, no. 5, pp. 403–420, 1970.

[95] Y.-J. Wang and H.-S. Lee, "The revised method of ranking fuzzy numbers with an area between the centroid and original points," *Computers & Mathematics with Applications*, vol. 55, no. 9, pp. 2033–2042, 2008.

[96] W. Pedrycz, P. Ekel, and R. Parreiras, *Fuzzy multicriteria decision-making: models, methods and applications*. John Wiley & Sons, 2011.

[97] C. Porcel, A. Tejeda-Lorente, M. Martínez, and E. Herrera-Viedma, "A hybrid recommender system for the selective dissemination of research resources in a technology transfer office," *Information Sciences*, vol. 184, no. 1, pp. 1–19, 2012.

[98] Z. Zhang, H. Lin, K. Liu, D. Wu, G. Zhang, and J. Lu, "A hybrid fuzzy-based personalized recommender system for telecom products/services," *Information Sciences*, vol. 235, pp. 117–129, 2013.

[99] G. Noh, H. Oh, Y. myoung Kang, and C. kwon Kim, "Psd: Practical sybil detection schemes using stickiness and persistence in online recommender systems," *Information Sciences*, vol. 281, no. 0, pp. 66 – 84, 2014.

요 약

추천 시스템(Recommender System, RS)은 궁극적인 소비자 (즉, 추천 시스템 사용자)에게 상업적인 아이템들을 추천해 주는 것이 주요 기능이다. 추천 시스템에서 정확한 정보를 제공하는 것은 추천 서비스 공급자와 시스템 사용자 모두에게 중요하다. 온라인 소셜 네트워크의 확산으로 추천 시스템의 영향력은 급격히 증가하고 있다. 반면에 추천 시스템의 의도와는 반대로 정보를 조작하는 거짓 아이덴터티들을 사용한 악의적인 사용자들의 추천 시스템에 대한 공격이 증가하고 있다. 이러한 거짓 아이덴터티들을 활용한 공격을 시빌(Sybil) 공격이라 부른다. 본 논문에서는 다른 연구에서 소개된 적이 없는 어드미션 통제 개념을 활용한 RobuRec이라 불리는 새로운 강건한 추천 시스템을 제안한다. 어드미션 통제라는 강력한 개념을 활용하여 정직한 사용자가 생성한 평가인지 혹은 시빌 아이덴터티들을 활용한 악의적인 평가인지에 관계없이 고신뢰 수준의 추천을 예측할 수 있다. RobuRec 시스템의 성능을 보이기 위해, 본 논문에서는 여러가지 가능한 시빌 공격 시나리오는 물론 다양한 데이터셋을 활용하여 광범위한 실험을 수행하였다. RobuRec은 실험 및 분석을 통해 RobuRec과 비교 가능한 PCA (Principal Component Analysis) 방식 및 LTSMF (Least Trimmed Squared Matrix Factorization) 방식보다 프리딕션 쉬프트 (Prediction Shift, PS) 및 적중 비율(Hit Ratio, HR)에서 월등한 성능을 보여 주었다.

주요어: 시빌 공격, 추천 시스템, 강건한 알고리즘, 정보 수준

학번: 2011-30791