석 사 학 위 논 문
Master's Thesis

# 상품 평가 품질 조작에 견고한 추천 시스템

## A Robust Recommendation System Against Review Quality Manipulation

2017


심 동 진 (沈 東 鎭 Sim, Dongjin)

한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

석 사 학 위 논 문

# 상품 평가 품질 조작에 견고한 추천 시스템

2017


심 동 진

한 국 과 학 기 술 원

전산학부

# 상품 평가 품질 조작에 견고한 추천 시스템

심 동 진

위 논문은 한국과학기술원 석사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2016년 12월 19일

심사위원장 이 윤 준 (인)

심 사 위 원 김 명 호 (인)

심 사 위 원 유 신 (인)

# A Robust Recommendation System Against Review Quality Manipulation

Dongjin Sim

Advisor: Yoon-Joon Lee

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

Daejeon, Korea
2016, December

Approved by

Yoon-Joon Lee
Professor of School of Computing

The study was conducted in accordance with Code of Research Ethics[1].

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

MCS 20153338

심동진. 상품 평가 품질 조작에 견고한 추천 시스템. 전산학부 . 2017년. 29+iii 쪽. 지도교수: 이윤준. (영문 논문)
Dongjin Sim. A Robust Recommendation System Against Review Quality Manipulation. School of Computing . 2017. 29+iii pages. Advisor: Yoon-Joon Lee. (Text in English)


**초 록**

추천 시스템은 사용자들의 구매 결정에 주요한 영향을 주기 때문에 악의적인 조작의 표적이 되어왔다. 대표적인 추천 방법은 상품에 대한 평가들이 정직하다는 가정하에 사용자의 평점을 분석하는 협업 필터링 방법이다. 하지만 협업 필터링 방법은 거짓된 평점을 주입하는 실링 공격에 의해 조작될 위험이 있다. 이로 인해 실링 공격의 영향력을 낮추는 여러 방법이 제안되었다. 최근에는, 많은 추천 시스템에서 사용자가 리뷰의 품질을 평가하는 점을 이용한 방법이 제안되었다.. 거짓된 리뷰는 사용자들에 의해 낮은 품질로 분류될 것이라는 가정하에 연구자들은 리뷰의 품질을 고려한 추천을 통해 실링 공격을 극복하고자 하였다. 하지만 이러한 추천 방법은 거짓된 리뷰 품질 평가 데이터가 주입된다면 오히려 조작의 정도가 심해질 위험이 있다. 본 논문에선 거짓된 리뷰 품질 평가 데이터가 주입되더라도 리뷰의 진실한 품질을 추정하는 견고한 추천 방법을 제안한다. 실제 데이터를 토대로 한 실험 결과를 통해 제안하는 방법의 견고함을 보였다. 제안한 방법은 거짓된 리뷰 품질 평가를 고려하지 않았을 때보다 최대 20배 견고한 실험 결과를 보였다.



**핵 심 낱 말** 견고한 추천 시스템, 협업 필터링, 추천 시스템 조작, 리뷰 품질 측정



**Abstract**

Recommendation systems influence decision making, and have become attractive targets of manipulation. Collaborative filtering, widely adopted in recommendation systems, exploits the observed reviews given by users to provide personalized recommendations under the assumption that all users honestly rate items. Unfortunately, shilling attacks which inject fake reviews can easily manipulate the systems with the naive assumption. Several approaches have been proposed to mitigate the effect of shilling attack. Recently, some researchers have been interested in the fact that most recommendation systems encourage users to write reviews, as well as rate the helpfulness of reviews written by other users. With the assumption that users will evaluate the helpfulness of fake reviews as low, they proposed recommendation systems considering the helpfulness as the quality of reviews.

However, those systems are vulnerable to attacks that inject fake helpfulness ratings to improve the quality of fake reviews. We propose a robust recommendation system to overcome such review quality manipulation attacks. The proposed approach estimates the true quality of reviews even in the presence of both injected fake reviews and helpfulness ratings. Experimental results on a real-world dataset demonstrate the robustness of our method. The proposed approach yields up to 20 times more robust recommendation results than the approaches that do not consider review quality manipulation.

**Keywords** Robust recommendation system, Collaborative filtering, Shilling attack, Review quality

# Contents

# List of Tables

# List of Figures

# Chapter 1. Introduction

Since recommendation systems influence purchase decisions, their positive recommendation can lead to significant monetary benefit for product sellers. According to [1], increasing the overall rating of a business by one star on Yelp[1] can increase the revenue of the business by 9%. Unfortunately, the strong impact of recommendation systems has attracted malicious attackers who try to bias recommendation results to manipulate the overall rating of their target items.

Matrix factorization(MF) model based collaborative filtering(CF) [6] is one of the most common approaches for the recommendation. MF infers users' tastes and items' attributes based on the observed ratings of users for items and recommends products whose attributes match a user's taste. MF naively assumes that all the observed ratings are conducted by honest users. In practice, however, this assumption is easily violated because of the presence of attackers. Due to the open nature of recommendation systems, attackers can inject fake reviews and ratings to increase the overall rating of their target items on the recommendation system. Such injections with intent to bias recommendation results are called, shilling attacks. There are a number of studies about shilling attack strategies [2, 3, 4, 5] and robust recommendation methods to prevent shilling attacks [8, 9, 10, 11, 12, 14].

Recent studies [16, 17] focused on the following dual roles of users in real-world recommendation systems. Users play a role of the reviewer of items by writing reviews about items in the form of a numeric rating score and review text. In addition, they play a role of the helpfulness rater of reviews. Namely, they rate the helpfulness score of reviews written by other users. Figure 1.1a contains a review of *Captain America: Civil War (DVD)* written by user *lmmyvasi29* with a 5-star rating. Figure 1.1b shows that other users rate the helpfulness of the review in Figure 1.1a.

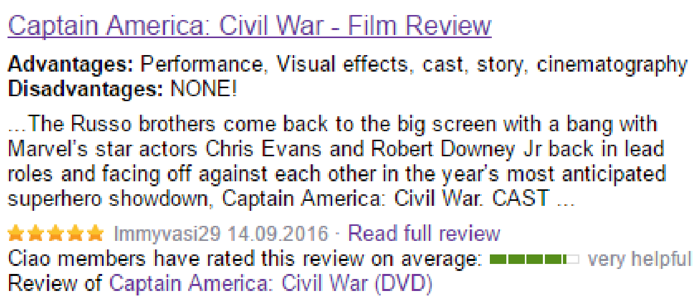
Wang et al. [17] considered helpfulness rating as users' implicit feedback about items, and incorporated the helpfulness rater role in recommendation systems to mitigate data sparsity and cold-start problems. Raghavan et al. [16] proposed collaborative

[1] www.yelp.com

filtering that considers quality of review to improve performance in the presence of fake reviews. Assuming that fake reviews receive a negative helpfulness rating, the proposed model measures the quality of a review by aggregateing the helpfulness ratings for the review, and lowers the impact of poor quality reviews on optimizing parameters of recommendation model. They show that incorporating review helpfulness rating information has potential benefits of improving the performance and robustness of recommendation systems.

However, they did not deal with fake review helpfulness ratings. After injecting fake reviews, malicious attackers can easily inject fake helpfulness ratings to promote the quality of fake reviews. They did not consider review quality manipulations that can promote the negative effect of fake reviews.

We propose a robust recommendation model even in the presence of both fake reviews and helpfulness ratings via a new review quality measure which estimates the true quality of reviews. Our approach consists of three stages. The first stage involves mapping users to a feature vector space to find groups of users suspected of being shilling users. In the second stage, the quality of each review is measured based on the Bayesian weighted mean of the helpfulness ratings associated with each review. If a helpfulness rater is suspicious, that is, he/she is very similar to a review writer, his/her helpfulness rating should have a low weight. In the final stage, the quality of each review measured in the previous stage is used as input for collaborative filtering. We adopt the cost function suggested by [7, 16]. Simulating various attacks on a real-world dataset, we demonstrate the effectiveness of the proposed method We show that our method mitigates the effect




(a) An item review example


(b) A helpfulness rating example

Figure 1.1: Item review and helpfulness rating example from Ciao

of manipulating review quality compared with other methods. In specific, our method yields up to 20 times more robust recommendation results than the approaches that do not consider attacks.

The thesis is organized as follows. Chapter 2 presents the background. Related work is described in Chapter 3. Our problem is defined in Chapter 4 and Chapter 5 describe our approach. In Chapter 6 presents the experimental methodology used to evaluate the robustness of our approach and results of our experiment. Finally, we conclude in Chapter 7.

## 2.1 Notation

Throughout this thesis, sets are denoted as italic capital letters and matrices/tensors are written as boldface capital letters. Let $U = \{u_1, u_2, \ldots, u_n\}$ and $I = \{i_1, i_2, \ldots, i_m\}$ be a set of users and items where $n$ and $m$ are the number of users and items, respectively. In recommendation systems, users can rate items in the form of a numeric rating score accompanied by review text. We use the term *'item rating'* and *'review'* to represent a numeric rating score for an item. We use the matrix $\boldsymbol{R} \in \mathbb{R}^{n\times m}$ to denote the item rating matrix where an entry $R_{u,i}$ indicates the item rating score of user $u$ for item $i$. Note that the item rating matrix $\boldsymbol{R}$ is sparse since users usually rate a small set of items. If user $u$ did not rate item $i$, then we assign "?" to the missing item rating $\boldsymbol{R}_{u,i}$. We use $IR = \{(u, i, ir)|u \in U, i \in I, ir = \boldsymbol{R}_{u,i}, ir \neq ?\}$ to represent item rating dataset where $(u, i, ir)$ means user $u$ rates item $i$ with score $ir$.

Many recommendation systems allow users to evaluate the helpfulness of other users' reviews in order to improve the user experience. After reading review content (numeric rating score and review text), users give the review helpfulness score in the form of a numeric rating score. We use the term *'helpfulness rating'* to indicate such a numeric rating score for a review. $(u_a, u_b, i_c, hr)$ means that user $u_a$ gives helpfulness score $hr$ to the review of user $u_b$ for item $i_c$. We use $\boldsymbol{H} \in \mathbb{R}^{n\times n\times m}$ to denote the helpfulness rating tensor where an entry $\boldsymbol{H}_{a,b,c}$ indicates the review helpfulness score that user $a$ gives to the review of user $b$ for item $c$. Similarly with item rating dataset, $HR = \{(u_a, u_b, i, hr)|u_a, u_b \in U, i \in I, hr = \boldsymbol{H}_{a,b,c}, hr \neq ?\}$ represents helpfulness rating dataset.

Since we consider an attacker who injects fake users, we use $U^g$ and $U^f$ to denote the set of genuine users and fake users, respectively. Genuine item rating and helpfulness rating dataset are denoted by $IR^g = \{(u, i, ir)|u \in U^g\}$ and $HR^g = \{(u, v, i, hr)|u \in U^g\}$, respectively. Similarly, fake item rating and helpfulness rating dataset are denoted by $IR^f = \{(u, i, ir)|u \in U^f\}$ and $HR^f = \{(u, v, i, hr)|u \in U^f\}$, respectively. Unless otherwise noted, the range of item rating score is from 1 to 5, and helpfulness rating score ranges from 0 to 5.

## 2.2 Matrix Factorization Based Collaborative Filtering

The objective of CF is to predict the value of the missing entries in a item rating matrix. Matrix factorization (MF) is the popular technique to predict the missing entries in a matrix by inferring latent patterns from the observed entries of the matrix. In the context of CF, MF infers latent features of user and items based on the known item ratings of users for items. It decomposes a item rating matrix $\boldsymbol{R}$ into two latent matrices $\boldsymbol{U} \in \mathbb{R}^{n \times d}$ and $\boldsymbol{V} \in \mathbb{R}^{d \times m}$ corresponding latent features of user and item, respectively, where the $d$ is the number of latent features. In specific, MF optimizes the two matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ by minimizing the following cost function which is the sum of prediction error terms and regularization terms.

$$Cost(\boldsymbol{U}, \boldsymbol{V}|\boldsymbol{R}) = \sum_{\boldsymbol{R}_{i,j} \neq ?} (\boldsymbol{R}_{i,j} - (\boldsymbol{U}\boldsymbol{V})_{i,j})^2 + \lambda(||\boldsymbol{U}||_F^2 + ||\boldsymbol{V}||_F^2) \tag{2.1}$$

After obtaining the optimized $\boldsymbol{U}$ and $\boldsymbol{V}$, all missing entries in the item rating matrix $\boldsymbol{R}$ are predicted via the dense matrix $\boldsymbol{U}\boldsymbol{V}$ which is the product of $\boldsymbol{U}$ and $\boldsymbol{V}$. We construct a toy example (Figure 2.1) as follows. The item set of the item rating matrix consists of two romance movies, two horror movies, and one bad movie. The user set of the item rating matrix consists of two romance lovers and two horror lovers. After performing MF on the item rating matrix $\boldsymbol{R}$, prediction matrix $\boldsymbol{U}\boldsymbol{V}$ captures the tastes of users and judges that users would not like the bad movie.

| | | Romance | | Horror | | Bad |
|---|---|---|---|---|---|---|
| | | Movie 1 | Movie 2 | Movie 3 | Movie 4 | Movie 5 |
| Romance Lovers | User 1 | 5 | 5 | ? | 2 | ? |
| | User 2 | 4 | ? | 2 | ? | 1 |
| Horror Lovers | User 3 | 2 | ? | 4 | 4 | 1 |
| | User 4 | ? | 2 | 5 | 5 | ? |

(a) Rating matrix $\boldsymbol{R}$

| | | Romance | | Horror | | Bad |
|---|---|---|---|---|---|---|
| | | Movie 1 | Movie 2 | Movie 3 | Movie 4 | Movie 5 |
| Romance Lovers | User 1 | 5 | 5 | 2.1 | 2 | 1.2 |
| | User 2 | 4 | 3.8 | 2 | 2 | 1 |
| Horror Lovers | User 3 | 2 | 1 | 4 | 4 | 1 |
| | User 4 | 3.7 | 2 | 5 | 5 | 1.4 |

(b) Prediction matrix $\boldsymbol{U}\boldsymbol{V}$

Figure 2.1: Matrix factorization example

## 2.3 Shilling Attack

There are malicious attackers who try to manipulate a CF-based recommendation system to gain monetary benefit. These attackers attempt to manipulate the recommendation system that operates through user rating data by injecting fraudulent user (Shillers) and fake rating data, and such attempts are called 'Shilling attacks.' Shilling attacks are categorized into two categories: *push attacks* inject shillers which give high item ratings to particular items to promote the recommendation score for the particular items, while *nuke attacks* inject shillers which give low item ratings to particular items aiming at decreasing the popularity of the items. However, injecting fake item rating associated with target items only is not enough to manipulate a CF-based recommendation system, because CF predicts normal users' item rating for target items in the way attackers wish if tastes of shillers are similar to those of normal users. In order to fully exploit the principle of collaborative filtering, shillers have to mimic rating behaviors of normal users. There are various attack models about how to mimic rating behavior of normal users: Random attack, Average attack, Bandwagon attack. [REF] In the context of push attack, Random attack injects fake users who give the highest rating to their target items and rate the randomly chosen items around the overall mean. Average attack generates fake users that give the highest rating to their target items and the mean rating of each item to randomly chosen items. Bandwagon attack consists of fake users whose ratings for their target items and popular items are maximum.

The figure 2.2 shows an example of shilling attack and its effect. We inject two shillers whose aims are boosting the prediction score of the bad movie. Shillers rate the

| | | Romance | | Horror | | Bad |
|---|---|---|---|---|---|---|
| | | Movie 1 | Movie 2 | Movie 3 | Movie 4 | Movie 5 |
| Romance Lovers | User 1 | 5 | 5 | ? | 2 | ? |
| | User 2 | 4 | ? | 2 | ? | 1 |
| Horror Lovers | User 3 | 2 | ? | 4 | 4 | 1 |
| | User 4 | ? | 2 | 5 | 5 | ? |
| Shillers | User 5 | ? | 3 | ? | 3 | 5 |
| | User 6 | 3 | ? | 3 | ? | 5 |

(a) Rating matrix $\boldsymbol{R}$

| | | Romance | | Horror | | Bad |
|---|---|---|---|---|---|---|
| | | Movie 1 | Movie 2 | Movie 3 | Movie 4 | Movie 5 |
| Romance Lovers | User 1 | 5 | 5 | 5 | 2 | 5 |
| | User 2 | 3 | 1 | 3 | 4 | 1 |
| Horror Lovers | User 3 | 3 | 1 | 3 | 4 | 1 |
| | User 4 | 5 | 2 | 5 | 5 | 3.9 |
| Shillers | User 5 | 4.3 | 3 | 4.3 | 3 | 5 |
| | User 6 | 3 | 3.2 | 3 | 1 | 5 |

(b) Prediction matrix $\boldsymbol{UV}$

Figure 2.2: Matrix factorization in the presence of shilling attack

bad movie with the highest possible rating value, i.e. 5, and rate other movies in a similar way to other genuine users. Matrix factoring misjudges the prediction scores of the bad movie because it has to reduce the error of the fake ratings to optimize its cost function. Compared with figure 2.1, figure 2.2 contains high predicted rating of genuine users for the bad movie.

## 2.4 Weighted Matrix Factorization for Robust Collaborative Filtering

The reason MF is vulnerable to shilling attack is that MF does not consider the presence of fake item ratings. Therefore, various attempts have been proposed to eliminate or mitigate the impact of fake item ratings on MF. The cost functions used in these attempts are expressed in the form of weighted matrix factorization(WMF) [7].

$$Cost(\boldsymbol{U}, \boldsymbol{V}|\boldsymbol{W}, \boldsymbol{R}) = \sum_{\boldsymbol{R}_{u,i} \neq ?} \boldsymbol{W}_{u,i}(\boldsymbol{R}_{u,i} - (\boldsymbol{U}\boldsymbol{V})_{u,i})^2 + \lambda(||\boldsymbol{U}||_F^2 + ||\boldsymbol{V}||_F^2) \tag{2.2}$$

$\boldsymbol{W} \in \mathbb{R}^{n \times m}$ is a weight matrix where $\boldsymbol{W}_{u,i}$ is the weight for the item rating $\boldsymbol{R}_{u,i}$. In this cost function, prediction error term changes from sum of squared errors to weighted sum of squared errors, which allows item ratings whose weight is small to have significant prediction error. This property helps WMF based on weight matrix where fake item ratings have a small weight to yield desirable predictions robust to fake item ratings.

Figure 2.3 shows an example of WMF where the weights of genuine reviews are larger than those of fake reviews. With well-assigned weight matrix, latent features of users and items are optimized to describe genuine review better, so shillers fail to manipulate prediction of genuine users for the bad movie.

| | | Romance | | Horror | | Bad |
|---|---|---|---|---|---|---|
| | | Movie 1 | Movie 2 | Movie 3 | Movie 4 | Movie 5 |
| Romance Lovers | User 1 | 5 | 5 | ? | 2 | ? |
| | User 2 | 4 | ? | 2 | ? | 1 |
| Horror Lovers | User 3 | 2 | ? | 4 | 4 | 1 |
| | User 4 | ? | 2 | 5 | 5 | ? |
| Shillers | User 5 | ? | 3 | ? | 3 | 5 |
| | User 6 | 3 | ? | 3 | ? | 5 |

(a) Rating matrix $\boldsymbol{R}$

| | | Romance | | Horror | | Bad |
|---|---|---|---|---|---|---|
| | | Movie 1 | Movie 2 | Movie 3 | Movie 4 | Movie 5 |
| Romance Lovers | User 1 | 0.9 | 0.9 | | 0.9 | |
| | User 2 | 0.9 | | 0.9 | | 0.9 |
| Horror Lovers | User 3 | 0.9 | | 0.9 | 0.9 | 0.9 |
| | User 4 | | 0.9 | 0.9 | 0.9 | |
| Shillers | User 5 | | 0.1 | | 0.1 | 0.1 |
| | User 6 | 0.1 | | 0.1 | | 0.1 |

(b) Weight matrix $\boldsymbol{W}$

| | | Romance | | Horror | | Bad |
|---|---|---|---|---|---|---|
| | | Movie 1 | Movie 2 | Movie 3 | Movie 4 | Movie 5 |
| Romance Lovers | User 1 | 5 | 5 | 2.2 | 2 | 1.3 |
| | User 2 | 4 | 3.8 | 2 | 1.9 | 1.1 |
| Horror Lovers | User 3 | 2 | 1 | 4 | 4 | 1 |
| | User 4 | 3.8 | 2 | 5 | 5 | 1.5 |
| Shillers | User 5 | 4.6 | 3.5 | 4 | 3.9 | 1.5 |
| | User 6 | 3.7 | 2.5 | 3.6 | 3.6 | 1.3 |

(c) Prediction matrix $\boldsymbol{U}\boldsymbol{V}$

Figure 2.3: Weighted matrix factorization with a well assigned weight matrix

However, if the weights of fake item ratings are large, then WMF would yield more manipulated predictions. Therefore, the key to WMF for robust CF is how to build a good weight matrix $\boldsymbol{W}$, in other words, how to capture fake item ratings and decrease their weights.

# Chapter 3. Related Work

This section describes various approaches to build a weight matrix. The common goal of following approaches is to detect suspicious item ratings and assign them low weight.

Early research for robust CF focused on detecting manipulated ratings by only examining a given rating matrix. For example, Bhaskar et al. [8] proposed Robust Matrix Factorization (RMF) using M-estimators to bound the effect of outliers and noisy data. [10, 11, 9] apply PCA-based variable selection to detect suspicious users in unsupervised setting.

Recently, many researchers have started to incorporate various types of additional information into the recommendation algorithm in order to improve the accuracy of the recommendation algorithm. In response to this trend, defense mechanisms against attacks using additional information have also been proposed. Three popular additional information used in the recommendation algorithm are review text and review helpfulness rating information.

Text-based approaches [12, 14, 13, 15] exploit review textual features and meta features to detect fake reviews. [12] detects spam reviews by finding duplicate and near-duplicate reviews and using learned logistic regression with manually labeled data. Ott et al. [14] collected training data through crowd-sourcing and accurately classified the deceptiveness of reviews based on n-grams. However, text-based approaches have several drawbacks. They require labeled data to train an accurate classifier, and the labeled data depends on the item domain.

Many review sites encourage users to rate the helpfulness of reviews. With helpfulness ratings by other users, users can examine the helpfulness of reviews with statistics such as “90 (out of 100) people found this review helpful” or “40 members have rated this review on average (somewhat helpful)”. Motivated by this, some researchers [15, 16] exploit review helpfulness rating information to measure the quality of reviews. Kim et al. [15] proposed the measure below to quantify the quality of a review by aggregating helpfulness ratings for the review.

$$Quality(review(u,i)) = \frac{1}{N} \sum_{\boldsymbol{H}_{v,u,i} \neq ?} \boldsymbol{H}_{v,u,i} \tag{3.1}$$

where $N$ is the number of helpfulness ratings for item rating $\boldsymbol{R}_{u,i}$. Under the assumption that spam reviews would receive bad helpfulness ratings, [16] builds the weight matrix used in the WMF with the above review quality measure. They demonstrate considering review helpfulness could improve the overall performance of recommendation in the presence of spam review.

## 4.1 Review Quality Manipulation

The review quality measure 3.1 relies on the naive assumption which all helpfulness ratings are genuine. However, this assumption is easily violated if attackers inject fake helpfulness ratings to promote the quality of their fake reviews. From an adversarial perspective, the cost of fake helpfulness rating injection would be not much more expensive than the cost of fake item rating injection. Therefore, when constructing weight matrix based on review helpfulness ratings, attempts to review quality manipulation must be thoroughly considered

Figure 4.1 shows the case an attacker successes to manipulate the helpfulness of fake reviews. The fake ratings for the bad movie have a larger weight than other normal users' ratings as shown in Figure 4.1. Due to the manipulated weight matrix, collaborative filtering outputs prediction matrix weighted toward fake reviews.

| | | Romance | | Horror | | Bad |
|---|---|---|---|---|---|---|
| | | Movie 1 | Movie 2 | Movie 3 | Movie 4 | Movie 5 |
| Romance Lovers | User 1 | 5 | 5 | ? | 2 | ? |
| | User 2 | 4 | ? | 2 | ? | 1 |
| Horror Lovers | User 3 | 2 | ? | 4 | 4 | 1 |
| | User 4 | ? | 2 | 5 | 5 | ? |
| Shillers | User 5 | ? | 3 | ? | 3 | 5 |
| | User 6 | 3 | ? | 3 | ? | 5 |

(a) Rating matrix $\boldsymbol{R}$

| | | Romance | | Horror | | Bad |
|---|---|---|---|---|---|---|
| | | Movie 1 | Movie 2 | Movie 3 | Movie 4 | Movie 5 |
| Romance Lovers | User 1 | 0.1 | 0.1 | | 0.1 | |
| | User 2 | 0.1 | | 0.1 | | 0.1 |
| Horror Lovers | User 3 | 0.1 | | 0.1 | 0.1 | 0.1 |
| | User 4 | | 0.1 | 0.1 | 0.1 | |
| Shillers | User 5 | | 0.9 | | 0.9 | 0.9 |
| | User 6 | 0.9 | | 0.9 | | 0.9 |

(b) Weight matrix $\boldsymbol{W}$

| | | Romance | | Horror | | Bad |
|---|---|---|---|---|---|---|
| | | Movie 1 | Movie 2 | Movie 3 | Movie 4 | Movie 5 |
| Romance Lovers | User 1 | 4.5 | 5 | 2.4 | 2.2 | 5 |
| | User 2 | 1.7 | 1.9 | 1.3 | 1.2 | 2.8 |
| Horror Lovers | User 3 | 1 | 1 | 3.8 | 4 | 1.8 |
| | User 4 | 2.9 | 2.1 | 4.9 | 5 | 5 |
| Shillers | User 5 | 3 | 3 | 3 | 3 | 5 |
| | User 6 | 3 | 3 | 3 | 3 | 5 |

(c) Prediction matrix $\boldsymbol{UV}$

Figure 4.1: Weighted matrix factorization with a badly assigned weight matrix

## 4.2 Attack Model

In this section, we define our attack model. The objective of our attack model is to manipulate collaborative filtering, which considers review quality, to increase overall predicted ratings for the target items. This thesis only focuses on push attack, because push attack is a more direct way to promote monetary benefit than nuke attack. We leave nuke attack for future work. Our attack model involves in injecting fake item rating and helpfulness rating dataset, $IR^f$ and $HR^f$.

### 4.2.1 Fake Item Rating Injection

Our attack model injects fake item ratings (reviews) to bias predicted ratings of collaborative filtering for the target items Popular strategies for fake item rating injection are Random attack, Average attack and Bandwagon attack. However, Since Random attack is known to be ineffective, this thesis focuses on Average attack, and Bandwagon attack. Similarity to [5], from the view of each fake user, the item set $I$ is partitioned into 4 subsets, $I^{target}, I^{popular}, I^{filler}$ and $I^{none}$ where $I^{target}$ is a set of target items; $I^{popular}$ is a set of items that many authentic users like i.e. popular items; $I^{filler}$ is a set of randomly chosen items which are referred to filler items; and $I^{none}$ is the set of the remaining items, i.e. $I^{none} = I - I^{target} - I^{popular} - I^{filler}$.

Each fake user rates items of $I^{target}, I^{popular}$ and $I^{filler}$ as follows. First, each fake user rates all target items with the highest possible rating score. Secondly, with the aim of being similar to other users, each fake user rates each filler item of $I^{filler}$ and its rating score is the average of all ratings given to the filler item. Finally, to increase the probability of being similar to a large number of other users, each fake user gives the highest possible rating score for all items of $I^{popular}$. Note that the size of each subset can vary. We assume a recommendation system manager could easily detect attacks associated with the too big size of $I_{target}$, $I_{filler}$, $I_{popular}$. Therefore, we limit the size of each subset to less than 1% of the size of an entire item set $I$. In summary, general form of fake item rating dataset $IR^f$ is defined as follows.

$$IR^f = \bigcup_{u^f \in U^f} \{(u^f, i, ir_{max}) | i \in I^{target}\} \cup \{(u^f, i, ir_{avg}(i)) | i \in I^{filler}\} \cup \{(u^f, i, ir_{max}) | i \in I^{popular}\} \tag{4.1}$$

$ir_{max}$ is the highest rating on the item rating scale and $ir_{avg}(i)$ is the average item rating of item $i$.

### 4.2.2 Fake Helpfulness Rating Injection

Our attack model attempts to manipulate collaborative filtering that takes into account review quality by injecting fake helpfulness ratings for review quality manipulation. Such injection makes our study differ from existing attack models. Each fake user gives the highest helpfulness rating to all the fake reviews about target items. Additionally, each fake user generates random helpfulness ratings for normal reviews to avoid attack detections. Formally, fake helpfulness rating dataset $HR^f$ is defined as follows:

$$HR^f = \bigcup_{u^f \in U^f} \{(u^f, v, i, hr_{max}) | v \in U^f, i \in I^{target}\} \cup \{(u^f, v, i, hr_{random}) | v \in U^g, i \in I, \boldsymbol{R}_{v,i} \neq ?\} \tag{4.2}$$

$hr_{max}$ is the highest helpfulness rating on the helpfulness rating scale and $hr_{random}$ represents random rating.

In the presence of fake helpfulness ratings, computing the quality of a review as an average of helpfulness ratings for the review misjudges fake reviews as high quality, which increases the negative impact of fake reviews on collaborative filtering considering review quality.

## 4.3 Problem Definition

With our attack model, we define our problem as follows: given reviews and helpfulness ratings in the presence of fake data injected by our attack model, estimate the true quality of reviews which is used to construct the weight matrix of WMF. We express our problem as follows.

$$Quality(review(u, i)) = F(Helpfulness\ ratings\ of\ review(u, i)\ under\ attack) \tag{4.3}$$

Then the problem can be divided into two sub-problems. One is to detect fake helpfulness ratings, and the other is to devise a robust estimator $F$ that produces consistent review quality regardless of the presence of fake helpfulness ratings.

# Chapter 5. Proposed Method

This section describes our method in detail. The following sections describe how to capture suspicious helpfulness ratings and how to estimate the true quality of reviews.

## 5.1 User2Vec

Recently, various prediction tasks [18, 19, 20, 21] have improved performance by learning the desirable features themselves, instead of manually determining domain-specific features. Skip-gram model [18] is a popular model proposed for natural language processing task by Mikolov et al. The goal of the Skip-gram model is to capture semantic relationships between words. With the hypothesis that words which frequently appear together in sentences have semantic relationships, the Skip-gram model takes large real-world text corpus as training data and learns feature representations for words. In specific, it maps words to a feature space such that words frequently appear together in sentences have similar feature vectors. The Skip-gram model is widely adopted for natural language processing task due to the efficiency and ability to capture useful relationships in the text data. Inspired by the success of the Skip-gram model, some researchers apply the Skip-gram model to learning a mapping of vertices of a network to vectors which encode social relation [20, 21]. With the assumption random walk traces contain social relation between vertices, they generate samples of random walk traces as sequences of vertices and feed them into the Skip-gram model.

In this thesis, we propose User2Vec, an algorithm for learning feature representations for users in recommendation system to detect such suspicious relationships between users. [18] uses real-world sentences as sequences of semantically related words [20, 21] generates random walk traces as sequences of socially related vertices to obtain useful features for various prediction tasks. Similar to such approaches, we generate sequences of attack-related users and feed them into the Skip-gram model to obtain feature representations of users which are useful to detect fake users.

To generate sequences of attack-related users, we focus on behaviors of fake users. Several studies [10, 11] reported that fake users need to work together to maximize the effect of their attack. This strategy is referred to as group attack. In our attack model, fake users equally give the highest item rating to target items and the highest review helpfulness rating to their fake reviews. Taking this into consideration, we regard following relationships between users as clues to the group attack

1. Both user X and Y give $ir_{max}$ for an item

2. Both user X and Y give $hr_{max}$ for a review

3. User X gives $hr_{max}$ for a review written by user Y

We refer to the users rate an item with $ir_{max}$ as enthusiasts for the item. Similarly, we refer to the users rate the helpfulness of a review with $hr_{max}$ as supporters for the review. The first (second) relationship represents the pair of enthusiasts (supporters) whose opinions about some item (review) are same. If user $u_c$ and $u_d$ always rate in the same way items or reviews, it is reasonable to suspect that $u_c$ and $u_d$ are performing group attack. The last relationship indicates the pair of a reviewer and a supporter. If user $u_a$ always assigns the maximum helpfulness rating to all reviews written by another user $u_b$, then one can doubt that user $u_a$ intentionally promote the influence of user $u_b$. Note that we only target the ratings with the highest score only since we focus on push attack. Of course, pairs of normal users could reveal clues to the group attack due to the coincidence of opinions about items or reviews. However, all fake users have to involve in many connections through the relationships associated with group attack as a necessity to maximize the degree of manipulation. With this in mind, we suspect the truthfulness of a helpfulness rating if the rater and the reviewer are frequently connected through the mentioned relationships.

User2Vec consists of two steps. In the first step, we sample user pairs that reveal clues to the group attack. Sampling user pairs corresponding to the first relationship proceeds as follows. Among the items having at least two reviewers who rate the item with the highest possible rating, we first sample an item with the probability proportional to the cardinality of the users associated with the item and choose two reviewers for the sampled item uniformly at random. Sampling user pairs corresponding to the second and last relationship associated with the group attack involves in sampling reviews. For a review to be a sample, it should receive at least two highest ratings. The probability of sampling a review is proportional to the number of the helpfulness raters for the review. We choose two helpfulness raters who rate the review with the highest possible rating for the sample related to the second relationship. With a sampled review, we sample one supporter for the review and produce a pair of reviewer and supporter. In the last step, we feed the sampled user pairs into the Skip-gram model and obtain feature vectors of users. We expect the obtained feature vectors encode group attack patterns. In other words, fake users are very closely located to each other in the feature space, while normal users are scattered. Note that User2Vec, which places the fake users very close to each other in the feature space, does not guarantee that normal users are positioned away from each other in the feature space. However, if the dimension of the feature space is moderately high, the probability that the similarity of two arbitrarily selected users is high is very small. Therefore, although there is a risk of judging false positives, we judge that the relationship between two users with very high similarity is not trustful and define the suspiciousness of a helpfulness rating as a function of the similarity between the feature vectors of the helpfulness rater and the reviewer.

## 5.2 Robust Review Quality Measure

We assume that the true quality of a review can be estimated by the mean of authentic helpfulness ratings. However, in the presence of fake helpfulness ratings, we need to estimate the true quality of a review by the weighted mean of helpfulness ratings where the fake helpfulness ratings have very low weight. If the number of helpfulness ratings is sufficiently high, then this estimation get high confidence. However, for reviews that have few helpfulness ratings and reviews having only fake helpfulness ratings, the weighted mean is not robust estimator for such reviews. Say a fake review which received only fake helpfulness ratings. Then the weighted mean output is biased toward fake helpfulness ratings even if the weight of fake helpfulness ratings is almost zero. With all of these things in mind, we define the following review quality measure to estimate the true quality of a review.

$$Quality(u,i) = \frac{w_{prior}Q_{default} + \sum_{v\in\{x|\boldsymbol{H}_{v,u,i}\neq ?\}} T(v,u) \times \boldsymbol{H}_{v,u,i}}{w_{prior} + \sum_{v\in\{x|\boldsymbol{H}_{v,u,i}\neq ?\}} T(v,u)} \tag{5.1}$$

We take Bayesian approach that incorporates both a prior belief and a weighted mean of review helpfulness ratings associated with a review r. The prior quality $Q_{default}$ works as prior belief. $w_{prior}$ is the weight given to the prior belief ($Q_{default}$). In this work, we set $Q_{default}$ as the mean of helpfulness rating range (e.g. 2.5 in the range from 0 to 5), and $w_{prior}$ as 1. The weight of a review helpfulness rating in review quality estimation is determined by the following function $T : U \times U \rightarrow R$.

$$T(v, u) = \begin{cases} exp(-\mu \times (cosine(userVec_v, userVec_u) - \theta)) & \text{if } cosine(userVec_v, userVec_u) \geq \theta \\ 1 & \text{otherwise} \end{cases} \tag{5.2}$$

The function $T$ takes a rater and a reviewer, and output the trustfulness of the relationship between them. $T$ penalizes the trustfulness if the cosine similarity between the two feature vectors of the reviewer and the helpfulness rater is larger than the threshold $\theta$. As mentioned earlier, this policy might lower the weight of authentic helpfulness ratings, but the likelihood of making such a misjudgment is low in the moderately high dimensional feature spaces. $\mu$ is a constant for amplification of similarity. In this work, we set the $\theta$ as 0.8 and the $\mu$ as 100

With the above mentioned robust measure, the quality of reviews with few helpfulness ratings will be close to the default quality $Q_{default}$, while reviews with many helpfulness ratings given by the users whose similarity to the reviewer is not that high will have a quality score close to its average helpfulness rating. Most importantly, reviews with many helpfulness ratings from the users similar to reviewer will have a helpfulness score close to the default quality $Q_{default}$. In other words, our measure prevents fake helpfulness ratings from manipulating the quality of their fake review.

**Algorithm 1** User2Vec algorithm

**function** USER2VEC(dimensions $d$, num_samples $n$, item rating matrix $\boldsymbol{R}$, helpfulness rating tensor $\boldsymbol{H}$)

  Initialize *clues* to *empty*

  **for** $iter = 1$ to $n$ **do**

    append EnthusiastPair($\boldsymbol{R}$) to *clues*

    append SupporterPair($\boldsymbol{H}$) to *clues*

    append ReviewerSupporterPair($\boldsymbol{H}$) to *clues*

  **end for**

  $userVec =$ Skip-Gram$(clues, d)$

  **return** $userVec$

**end function**

**function** ENTHUSIASTPAIR(item rating matrix $\boldsymbol{R}$)

  let $Enthusiast(item)$ be $\{u|\boldsymbol{R}_{u,item} = ir_{max}\}$

  let $EI$ be $\{item||Enthusiast(item)| \geq 2\}$

  sample item $i$ from $EI$ with the prob. proportional to the cardinality of $Enthusiast(i)$

  sample user $u, v$ from $Enthusiast(i)$ uniformly at random

  **return** $(u, v)$

**end function**

**function** SUPPORTERPAIR(review helpfulness rating tensor $\boldsymbol{H}$)

  let $Supporter(u, i)$ be $\{v|\boldsymbol{H}_{v,u,i} = hr_{max}\}$

  let $SU$ be $\{(u, i)||Supporter(u, i)| \geq 2\}$

  sample review $(u, i)$ from $SU$ with the prob. proportional to the cardinality of $Supporter(u, i)$

  sample user $u_a, u_b$ from $Supporter(u, i)$ uniformly at random

  **return** $(u_a, u_b)$

**end function**

**function** REVIEWERSUPPORTERPAIR(review helpfulness rating tensor $\boldsymbol{H}$)

  let $Supporter(u, i)$ be $\{v|\boldsymbol{H}_{v,u,i} = hr_{max}\}$

  let $SU$ be $\{(u, i)||Supporter(u, i)| \geq 1\}$

  sample review $(u, i)$ from $SU$ with the prob. proportional to the cardinality of $Supporter(u, i)$

  sample user $v$ from $Supporter(u, i)$ uniformly at random

  **return** $(u, v)$

**end function**

**Algorithm 2** Robust recommendation system

**function** RRS(dimensions $d$, num_samples $n$, item rating matrix $\boldsymbol{R}$, helpfulness rating tensor $\boldsymbol{H}$)

$userVec = \text{User2Vec}(d, n, \boldsymbol{R}, \boldsymbol{H})$

**for** all review $(u, i)$ **do**

$\boldsymbol{W}_{u,i} = Quality(u, i) = \frac{w_{prior} Q_{default} + \sum_{v \in \{x | \boldsymbol{H}_{v,u,i} \neq ?\}} T(v,u) \times \boldsymbol{H}_{v,u,i}}{w_{prior} + \sum_{v \in \{x | \boldsymbol{H}_{v,u,i} \neq ?\}} T(v,u)}$

**end for**

Optimize $Cost(\boldsymbol{U}, \boldsymbol{V} | \boldsymbol{W}, \boldsymbol{R}) = \sum_{\boldsymbol{R}_{i,j} \neq ?} \boldsymbol{W}_{i,j} (\boldsymbol{R}_{i,j} - (\boldsymbol{U}\boldsymbol{V})_{i,j})^2 + \lambda(||\boldsymbol{U}||_F^2 + ||\boldsymbol{V}||_F^2)$

**return** $\boldsymbol{U}$ and $\boldsymbol{V}$

**end function**

# Chapter 6. Experiment

## 6.1 Experimental Setting

We use the publicly available dataset provided by [22], namely CiaoDVD. The CiaoDVD dataset contains users' review and helpfulness rating information from ciao.dvd.co.uk where users rate DVDs and others' reviews. In the CiaoDVD dataset, users can rate items with a score from 1 to 5 as reviewer, and rate the helpfulness of reviews with a score from 0 to 5 as helpfulness rater. From the original dataset, we filter out the reviewers who rated less than five items and item that received less than five ratings. The statistics of the resulting dataset are shown in Table 6.1.

Table 6.1: Statistics of CiaoDVD dataset

| **Features** | **CiaoDVD** |
|---|---|
| Reviewers | 1822 |
| Items | 2069 |
| Reviews | 28374 |
| Helpfulness Rater | 27900 |
| Helpfulness Rating | 661040 |

We assume the original data is authentic. To this data, we inject fake users, item ratings(reviews) and helpfulness ratings as mentioned in Chapter 4. Attack size, i.e. the number of injected fake users, ranges from 1% to 3% of total users. Filler size is the number of $I^{filler}$ and popular size is the number of $I^{filler}$. We restrict the sum of filler size and popular size from exceeding 1% of the total number of items. We assume attacks with larger attack size and filler/popular size would be detected easily, so we decide to exclude them in our experiment setting. For performance evaluation, we perform 10-fold cross validation. In each fold, the test set contains random 10% original reviews, and the training set contains the remaining 90% original reviews and all the fake reviews. We choose a set of target items $I^{target}$ as items which have been rated by at least 1% users with below the median of the rating scale (3 in our rating scale [1,5]). $I^{popular}$ consists of items rated items by at least 1% users with the $ir_{max}$. Note that $I^{filler}$ is randomly selected for each fake user.

## 6.2 Metrics

**Average quality of reviews** measures the average quality of reviews belonging to each category. We compare the average quality values of fake reviews and authentic reviews to find out the robustness of a quality measure. A robust quality measure should produce the small average quality of fake reviews even in the presence of fake helpfulness ratings.

$$AverageHelpfulness(IR) = \frac{1}{|IR|} \sum_{(u,i) \in IR} helpfulness(u,i) \tag{6.1}$$

where $N_{IR}$ is the number of reviews in the set of review $IR$

**Prediction shift on the target items** measures the average of the change in the prediction of genuine users for the attacked items before and after a shilling attack. In other words, this metric measures the degree of success of an attack. The smaller the value of this metric, the more robust the recommendation method is.

$$PredictionShift(U,V,U',V') = \frac{1}{|U^g||I^{target}|} \sum_{u \in U^g} \sum_{i \in I^{target}} (U'V')_{u,i} - (UV)_{u,i} \tag{6.2}$$

where $(U'V')_{u,i}$ is the predicted rating value of user $u$ for item $i$ after an attack.

**Mean Average Error(MAE) on test set** is the overall prediction error on ratings in the test set which contains 10% of all item ratings of original users in the dataset. MAE is commonly used to compare the predictive accuracy of recommendation algorithms. We use MAE to measure the accuracy loss that is sacrificed to improve robustness.

$$MAE(U,V) = \frac{1}{|testset|} \sum_{R_{u,i} \in testset} |R_{u,i} - (UV)_{u,i}| \tag{6.3}$$

## 6.3 Results and Analysis

We first visualize the results of User2Vec to show User2Vec's ability to capture fake users. We use learned users' feature vectors as the input to the visualization tool, t-SNE [24]. The users are mapped to the 2-D space. x-shaped points represent fake users,

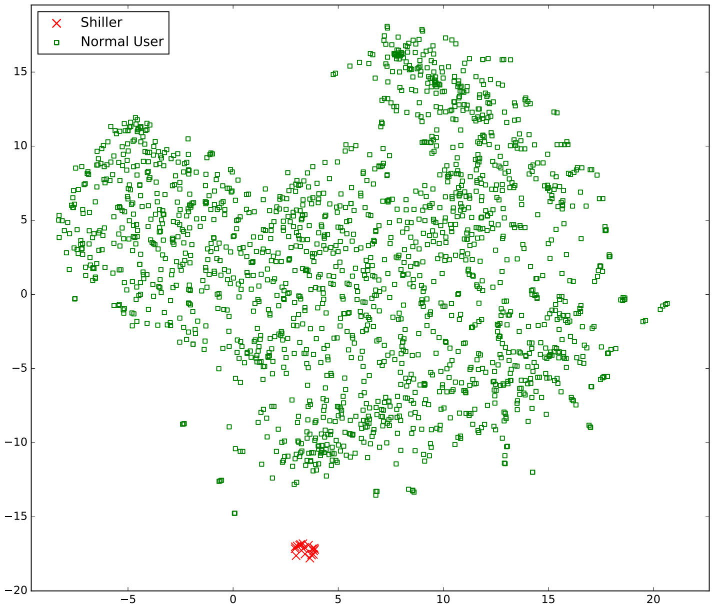



Figure 6.1: Visualization of the User2Vec result

while circle-shaped green colored points represent normal users. We observed that feature vectors of fake users are very close to each other.

We compute the average of the quality of fake reviews and authentic reviews. We inject fake review through attacks with 1% attack size, 0.5% filler size, and 0.5% popular size. We set dimensions of feature vectors to 32, which is used in User2Vec. As shown in table 6.2, the naive quality measure results in fake reviews have a higher quality than authentic reviews, whereas our quality measure yields the opposite. In specific, while the naive quality measure computes the quality of fake reviews at a value close to $hr_{max}$, our quality measure computes the quality of fake reviews at a value close to the default quality $Q_{default}$. In other words, our quality measure can prevent fake helpfulness ratings from increasing the quality of fake reviews. We observed a slight decrease in the quality of authentic reviews when using our method. The reason for this is that the feature vector of an authentic user might be very similar to that of another authentic user by coincidence, which leads to false positive detection. However, since the probability that false positive detection happens is very low, authentic helpfulness ratings which are wrongly sacrificed have no significant impact on estimating review quality. Hence, our quality measure shows its ability to estimate the true quality of reviews.

To compare the robustness, we compute the prediction shift on the target items of

Table 6.2: Review helpfulness results. The range of helpfulness is from 0 to 5

| Attack Size | Naïve Helpfulness Measure | | Our Helpfulness Measure | |
|---|---|---|---|---|
| | Fake Reviews | Authentic Reviews | Fake Reviews | Authentic Reviews |
| 1% | 5.0 | 3.5388 | 2.5 | 3.45 |

algorithms using different review quality measures under attacks in the variety of attack sizes, filler sizes, and popular sizes. From Table 6.3, we observe that the WMF using our quality measure leads to the lowest prediction shift on the target items in all conditions. The reason for this is that fake item ratings have the least impact on prediction when applying our quality measure rather than applying other methods. Since fake helpfulness ratings increase the influence of fake item ratings on prediction when applying the naive quality measure than when ignoring review quality, the naive quality measure causes larger prediction shift on the target items than MF in the presence of fake helpfulness ratings. Therefore we argue that our method is resistant to review quality manipulations.

Table 6.3: Prediction shift on the target items

| **Attack size** | **Filler Size** | **Popular Size** | **Base** | **Naive** | **Ours** |
|---|---|---|---|---|---|
| 1% | 1% | 0% | 1.01465 | 1.7805 | **0.092686** |
| | 0.50% | 0.50% | 1.48154 | 1.90081 | **0.239489** |
| | 0% | 1% | 1.72337 | 1.79765 | **0.246163** |
| 2% | 1% | 0% | 1.5178 | 2.28872 | **0.172815** |
| | 0.50% | 0.50% | 1.91454 | 2.30772 | **0.387328** |
| | 0% | 1% | 2.08557 | 2.1092 | **0.469354** |
| 3% | 1% | 0% | 1.79902 | 2.5618 | **0.307992** |
| | 0.50% | 0.50% | 1.89458 | 2.49592 | **0.580003** |
| | 0% | 1% | 1.99723 | 2.24161 | **0.659305** |

We also investigate the predictive performance of each method. We compute MAE on the test set. According to 6.4, MF performs better than WMFs. In fact, MAE is not ideal metric to measure the predictive performance of WMF, because MAE gives equal importance to the errors of all the ratings in the test set. Even though MAE is not proper metric for WMF, MAE results of WMF are not significantly different from those of MF. We compute Cohen's $d$ for the effect size based on means of predictive errors between MF and WMF using our quality measure. The value of Cohen's $d$ is near 0.02, which is a small value according to [23]. Consequently, we conclude that WMF using our quality measure provides robustness at a not significant additional cost of predictive accuracy.

Table 6.4: MAE on test set

| Attack Size | Filler Size | Popular Size | Base | Naive | Ours |
|---|---|---|---|---|---|
| 1% | 1% | 0% | **0.821944** | 0.831673 | 0.843252 |
| | 0.5% | 0.5% | **0.825909** | 0.836473 | 0.842159 |
| | 0% | 1% | **0.825078** | 0.837706 | 0.84074 |
| 2% | 1% | 0% | **0.820305** | 0.826343 | 0.840074 |
| | 0.5% | 0.5% | **0.826043** | 0.834851 | 0.836148 |
| | 0% | 1% | **0.822775** | 0.847235 | 0.843203 |
| 3% | 1% | 0% | **0.816267** | 0.828714 | 0.840653 |
| | 0.5% | 0.5% | **0.825463** | 0.846773 | 0.841859 |
| | 0% | 1% | **0.829685** | 0.85275 | 0.841454 |

# Chapter 7. Conclusion

추천 시스템은 사용자 경험이 손상되지 않게 하기 위해 가짜 리뷰를 고려해야한다. 가짜 리뷰가 있는 상황속에서도 견고한 추천 결과를 얻기 위해 리뷰 품질을 고려하는 추천 모델들이 제안되어왔다. 하지만 제안된 모델의 대부분은 리뷰 품질도 조작하려는 공격들을 고려하지 않는다. 우린 이러한 공격의 심각성을 인지하여 리뷰 품질 조작으로부터 견고한 추천 모델을 제안한다. User2Vec와 Bayesian을 제안함으로써 가짜 리뷰의 영향력을 낮추고 이를 통해 이전보다 더 견고한 추천 결과를 도출한다. 실제 데이터에서의 실험 결과를 통해 우리 모델이 리뷰 품질 조작을 고려하지 않은 모델보다 최대 20배 견고하다는 것을 보였다. Future work의 방향으론 nuke attack등에도 견고한 추천 모델을 고안하는 방향이 있다.

Recommendation systems should tackle fake reviews to ensure that user experience is not compromised. Several recommenderdation models have been proposed that take into account the quality of reviews to yield robust recommendation results even in situations with fake reviews. However, most of the proposed models do not deal with attacks that attempt to manipulate review quality. We address the severity of these attacks and propose a robust recommendation model against review quality manipulation. We propose User2Vec and Bayesian Weighted Mean to reduce the negative effect of fake reviews and thereby obtain more robust recommendation results. Experimental results on a real-world dataset show that our model is up to 20 times more robust than the models which do not consider review quality manipulation. Future research directions include developing robust models against nuke attacks and more elaborate attacks.

# Bibliography


[1] M. Luca. Reviews, Reputation, and Revenue: The Case of Yelp.com. Harvard Business School Working Papers, 2011.

[2] S. Lam and J. Riedl. Shilling recommender systems for fun and profit. In Proceedings of the Thirteenth International Conference on World Wide Web. ACM, 2004.

[3] R. Burke, B. Mobasher, R. Zabicki, and R. Bhaumik. Limited knowledge shilling attacks in collaborative filtering systems. In Proceedings of the Third IJCAI Workshop in Intelligent Techniques for Personalization, 2005.

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

[5] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik. Analysis and detection of segment-focused attacks against collaborative recommendation. 2006.

[6] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. Computer Journal, 42, 2009.

[7] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In Proceedings of the Eighth IEEE International Conference on Data Mining, 2008.

[8] B. Mehta, T. Hofmann, and W. Nejdl. Robust Collaborative Filtering. In Proceedings of the 1st ACM Conference on Recommender Systems, 2007.

[9] B. Mehta and W. Nejdl. Attack resistant collaborative filtering. In Proceedings of the Thirty-First Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2008.

[10] B. Mehta, T. Hofmann, and P. Fankhauser. Lies and propaganda: detecting spam users in collaborative filtering.Proceedings of the 12th international conference on Intelligent user interfaces, 2007.

[11] B. Mehta. Unsupervised shilling detection for collaborative filtering. In Proceedings of the Twenty-Second Conference on Artificial Intelligence. AAAI, 2007.

[12] N. Jindal and B. Liu. Opinion spam and analysis. In WSDM, 2008.

[13] J. Liu, Y. Cao, C.-Y. Lin, Y. Huang, and M. Zhou. Low-quality product review detection in opinion summarization. In EMNLP-CoNLL, 2007.

[14] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock. Finding Deceptive Opinion Spam by Any Stretch of the Imagination. ACL, 2011.

[15] S. Kim, P. Pantel, T. Chklovski, and M. Pennacchiotti. Automatically assessing review helpfulness. EMNLP, 2006.

[16] S. Raghavan, S. Gunasekar, and J. Ghosh. Review quality aware collaborative filtering. In Proceedings of the sixth ACM conference on Recommender systems, 2012.

[17] S. Wang, J. Tang, and H. Liu, Toward Dual Roles of Users in Recommender Systems, In CIKM, 2015.

[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In ICLR, 2013.

[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In NIPS, 2013.

[20] B. Perozzi, R. Al-Rfou, and S. Skiena. DeepWalk: Online learning of social representations. In KDD, 2014.

[21] A. Grover and J. Leskovec. node2vec: Scalable Feature Learning for Networks. In KDD, 2016.

[22] G. Guo, J. Zhang, D. Thalmann, and N. Yorke-Smith. Etaf: An extended trust antecedents framework for trust prediction. In ASONAM, 2014.

[23] J. Cohen. Statistical Power Analysis for the Behavioral Sciences. 1988.

[24] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. Journal of Machine Learning Research, 2008.

# Acknowledgments in Korean

이 논문을 작성하기까지 도움을 주신 모든 분들께 감사드립니다. 먼저 석사 과정 동안 지도해주시고 저의 여러 고민을 성심성의껏 상담해주신 이윤준 교수님께 깊은 감사의 말씀을 드립니다. 교수님의 제자로서 부끄럽지 않은 사람이 되도록 항상 노력하겠습니다. 아울러 귀한 시간을 내시어 논문을 심사해주시고 조언을 아끼지 않아 주신 김명호 교수님, 유신 교수님께도 깊은 감사 드립니다.

즐겁게 생활한 데이터베이스 연구실 사람들에게도 감사의 뜻을 전합니다. 연구실 학생들을 친자식처럼 대해주시고 저희가 잘 모르는 업무를 열심히 도와주신 박경희 선생님, 랩장으로서 책임감있게 연구실 사람들을 신경써주시고 웃음이 많은 후영누나, 제 말에 잘 웃어주시고 주말에도 연구실에 나와서 열심히 연구하던 수형이형, 컴퓨터 장비 관련 문제를 도와주신 똑똑한 우람이형, 프로젝트도 같이 하고 삶의 여러 지혜를 전수하시고 저의 디펜스도 열심히 도와주신 민호형, 백과사전같이 다방면으로 아는 것이 많고 공감대 형성이 잘 되어 재밌던 창욱이형, 연구실의 분위기를 담당한 화진누나, 연구실 일을 항상 묵묵히 잘 해내고 여러모로 도와주는 대인배 남윤이, 더 같이 지내지 못해 아쉬운 원영이형에게도 감사의 마음을 전합니다. 데이터베이스 연구실 가족이 되어 정말 행복한 대학원 생활을 할 수 있었습니다.

그리고 기쁜 일과 슬픈 일을 함께한 석사 동기들과 친구들에게도 감사의 인사를 드립니다. 앞으로도 함께하는 시간을 많이 가질 수 있길 희망하고 하고자 하는 일 모두 잘 되길 응원하겠습니다.

마지막으로 항상 절 지지해주시는 사랑하는 가족들에게도 감사드립니다. 지금의 제가 있을 수 있도록 길러주시고 베풀어주셔서 감사합니다. 앞으로도 멋진 모습을 보여드리며 은혜에 보답하도록 하겠습니다.

## Curriculum Vitae in Korean

이　　　름: 심 동 진

생 년 월 일: 1992년 8월 4일

출　생　지: 경기도 송탄시

전 자 주 소: djsim@kaist.ac.kr

### 학　　력

2008. 3. – 2011. 2.　거창고등학교

2011. 3. – 2014. 8.　성균관대학교 소프트웨어학과 (학사)

2015. 3. – 2017. 2.　한국과학기술원 전산학부 (석사)

### 경　　력

2015. 9. – 2016. 8.　한국과학기술원 전산학부 조교

2015. 7. – 2015. 8.　한국과학기술정보연구원 인턴