# Personalized Privacy-Preserving Recommendation System

by

**Qi Wen**

A thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements for the degree of

**Master of Science**

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada

March 2022

Copyright © 2022 by Qi Wen

Thesis advisor

**Dr. Carson K. Leung**

Author

**Qi Wen**

**Personalized Privacy-Preserving Recommendation System**

# Abstract

In the current big data era, recommendation systems play an important role in our daily life to help us make faster and better decisions from massive amounts of choices. However, the exposure of sensitive data raises a big privacy concern. Research has shown that it is possible to de-identify anonymous users, i.e., inferring sensitive information from non-sensitive data such as movie ratings). Hence, in this M.Sc. thesis, I tackle the privacy issue by designing a semi-decentralized network called Trust-based Agent Network (TAN). It treats each node in the network as an agent, and it comprises two key concepts. First, data are distributed to each agent inside each trusted network. Second, the recommendation service provider can only collect obfuscated data from agents by adopting the differential privacy mechanism. As a result, in the TAN, data are either protected inside local trusted networks or obfuscated outside of trusted networks. A final recommendation can then be made by aggregating the local suggestions from the trusted network and obfuscated global suggestions from the service provider. Note that the resulting recommendation emphasizes the recommendations from the trusted agents. Knowing the potential limitation of the reduced search space (which can cause inaccuracy in the recommendation result), the resulting recommendation also takes into account the general suggestions from the whole network. Experimental evaluation results show that my proposed solution protects privacy while preserving good accuracy.

# Table of Contents


Abstract . . . . . ii
Table of Contents . . . . . iv
List of Figures . . . . . v
List of Tables . . . . . vi
Acknowledgements . . . . . vii
Dedication . . . . . 1

**1 Introduction** **2**
1.1 Thesis Statement . . . . . 4
1.2 Thesis Organization . . . . . 5

**2 Background** **6**
2.1 Recommendation System . . . . . 6
2.2 Research Trend . . . . . 7
2.3 Recommendation System Techniques . . . . . 8
2.3.1 Burke's Classification . . . . . 9
2.3.2 Content-Based Filtering (CBF) . . . . . 11
2.3.3 Collaborative Filtering (CF) . . . . . 12
Memory-Based Collaborative Filtering . . . . . 12
Model-Based Collaborative Filtering . . . . . 14
Privacy Issue in Collaborative Filtering . . . . . 14
2.3.4 Hybrid Filtering . . . . . 15
2.4 Recommendation System Applications . . . . . 15
2.5 Summary . . . . . 16

**3 Related Works** **18**
3.1 Challenges for Recommendation Systems in Big Data Era . . . . . 19
3.1.1 Privacy Issues . . . . . 19
3.1.2 Existing Approach 1 for Privacy Issue: Raw Data Revision . . 20
3.1.3 Existing Approach 2 for Privacy Issue: Distributed Recommendation . . . . . 23

3.2 Other Challenges . . . 23
3.2.1 Data Sparsity . . . 24
3.2.2 Cold Start . . . 24
3.2.3 Scalability . . . 24
3.3 Summary . . . 25

**4 Personalized Privacy-Preserving Recommendation System 26**
4.1 Motivation . . . 27
4.2 System Overview . . . 29
4.3 Trust-Based Agent Network (TAN) . . . 30
4.4 TAN+DP: TAN with Differential Privacy . . . 33
4.5 TAN+DP: An Additional Step when Number of Ratings is Insufficient 38
4.6 Summary . . . 39

**5 Evaluation 41**
5.1 Dataset Acquisition . . . 41
5.2 Data Exploration and Preprocessing . . . 42
5.2.1 General Filter . . . 44
5.2.2 Testing Filter . . . 44
5.3 Evaluation Metrics . . . 45
5.3.1 Root Mean Square Error (RMSE) . . . 45
5.3.2 Accuracy and F-measure . . . 45
Precision . . . 47
Recall . . . 47
Accuracy . . . 47
F-measure . . . 48
5.4 Settings on the Weights for Global View (GV) and Local View (LV) . 48
5.5 Observations . . . 49
5.5.1 Privacy Preservation . . . 50
5.5.2 Accuracy and F-measure . . . 52
5.5.3 Run Time Complexity . . . 56
5.6 When to Take the Extra Step in TAN+DP? . . . 58
5.7 Summary . . . 61

**6 Conclusions and Future Work 63**
6.1 Conclusions . . . 63
6.2 Future Work . . . 65

**Bibliography 75**

# List of Figures


2.1 Recommendation System Technique Hierarchy . . . . . 8
2.2 Content-Based Filtering (CBF) Example . . . . . 11
2.3 Collaborative Filtering (CF) Example . . . . . 13

4.1 Trust-Based Agent Network (TAN) . . . . . 32
4.2 Use Differential Privacy to Upload Obfuscated Data . . . . . 33
4.3 Randomized Response . . . . . 35
4.4 Privacy-Preserving Recommendation System . . . . . 37

5.1 Distribution of Ratings in the LibraryThing Dataset . . . . . 43
5.2 Relation between RMSE and $\alpha$ . . . . . 49
5.3 Data Representation in Traditional Server . . . . . 50
5.4 Data Representation in Agent Server in TAN . . . . . 51
5.5 Prediction Accuracy . . . . . 53
5.6 F-measure . . . . . 55
5.7 Offline CF, Online CF and TAN+DP Running Time Comparison . . 57
5.8 Offline CF and TAN+DP Running Time Comparison . . . . . 57
5.9 Number of Ratings in Descending Order . . . . . 58
5.10 Compare TAN+DP without the Extra Step to Others . . . . . 59
5.11 Predicted Distributions of 0's in Randomized Response with Different Size of Ratings . . . . . 60
5.12 MER and Number of Ratings . . . . . 61

# List of Tables

2.1 Recommendation System Articles in Association for Computing Machinery (ACM) Library . . . . . 8

5.1 A Tuple in LibraryThing . . . . . 42
5.2 Basic Statistics of LibraryThing . . . . . 42
5.3 Rating Confusion Matrix . . . . . 46
5.4 Service Provider Accessibility on Data . . . . . 51
5.5 Prediction Accuracy Comparison . . . . . 53

# Acknowledgements

I would like to express my deepest gratitude to my research supervisor Dr. Carson K. Leung. Back when I was an undergraduate student, I acquired data mining knowledge from Dr. Leung, and published my first paper. During the graduate study, he provided me an opportunity to work on a Mitacs project with the industrial partner Protegra, Inc.. Such a project inspired me with an idea in the current thesis. This thesis would not be possible without his encouragement and academic support.

I would also like to thank my examining committee members, Dr. Cüneyt Gürcan Akçora (Computer Science and Statistics) and Dr. Saman Muthukumarana (Statistics) for their time in reviewing and analyzing my thesis. Thanks Dr. Avery Miller to chair my M.Sc. thesis defence. I would also like to thank mentors and colleagues, as well as Wadood Ibrahim, Ian Scatliff and Chris McLatchie from Protegra, Inc., for their kindness and patience.

Finally, I would like to thank my parents for supporting and believing in me throughout my graduate study.

Qi Wen
B.C.Sc.(Hons.), The University of Manitoba, 2019

*The University of Manitoba*

*March 28, 2022*

*This thesis is dedicated to my parents*

# Chapter 1

# Introduction

Due to the rapid development of the Internet of Things (IoT), a tremendous amount of data is generated by various applications that lead us to enter a big data era [Mar17; KKZ20]. Now, people have abilities to access a massive amount of information online through various devices and services. However, the number of choices could be overwhelming and exceed human perception's ability to make a good decision in a reasonable amount of time.

Recommendation systems—which are also known as information filtering systems—can provide us with suggestions based on our past activities and find the items of interest more efficiently. These systems have become very popular and powerful in many domains such as videos [ZKG+16], food [TAJC20], travel [CSA16], news [FKRA20], movies [ACJB18], and so on. Especially in an e-commerce setting, a good recommendation system will not only enhances user experiences but also increase revenue [PCH11].

Many works have been done to improve the performance of the algorithms to

get more accurate recommendation results. However, most of these works require a large amount of data to train the recommendation models and strongly rely on users' personal information such as demographic information (say, age, marital status, and income) and feedback on products. Especially, in *collaborative filtering (CF)* based recommendation system—which tries to find similar users and recommends items based on the similar users' ratings—the service provider collects users' all the information as much as possible in order to achieve better recommendation, and the exposure of personal information raised serious privacy concerns [LFR06]. Research has shown that many people feel uncomfortable with being tracked by such models [BKZ17; MS16].

It is commonly accepted that recommendation service providers can fully access to these data. However, the service provider could be vulnerable and malicious, the servers can be attacked by third parties. For instance, in 2018, Facebook (changed its corporate name to Meta now) reported that there were 500 million users' data leak[1]. Moreover, the service provider could be also suspicious, it can sell users' data to third parties for financial benefit, so the service provider should be also a potential attacker in a privacy-preserving recommendation system.

One may argue that the collected data are usually insensitive information like users' feedback on a certain product or video watching histories, and it is not necessary to protect this information. However, the information that is assumed to be insensitive is also vulnerable to privacy leaks, Narayanan and Shmatikov [NS08] demonstrated a practical de-identification attack called *linkage attack* and successfully uncovered sensitive users' information from the anonymous movie ratings of

[1] `https://www.nytimes.com/2018/09/28/technology/facebook-hack-data-breach.html`

subscribers. A linkage attack combines datasets that are available in the public with the anonymized dataset and re-identify personal information. There are mainly two types of solutions in literature review: data protection and distributed recommendation. The former is to protect data by data encryption or data obfuscation, and the latter is to distribute the data to clients. Our solution is a non-trivial combination of these two types. More specifically, we design a semi-decentralized network called Trusted-based Agent Network (TAN) and apply differential privacy mechanism to obfuscate data.

## 1.1 Thesis Statement

*For this M.Sc. thesis, I aim to design a privacy-preserving recommendation system that can protect users' privacy while preserving a good accuracy.* In particular, I explore the following questions in this thesis:

Q1: How privacy is preserved? In other words, does it prevent the data from attacks like de-identification? Does it protect the data from suspicious service providers?

Q2: With the privacy-preserving approach, does the recommendation algorithm still provide an accurate result?

My *key contributions* of this thesis include the design of a novel designed network architecture called Trust-based Agent Network (TAN), and my adaption of differential privacy (DP) into TAN. Experimental results show that such an incorporation of DP into my TAN protects privacy while preserves accurate recommendation.

## 1.2 Thesis Organization

The remainder of the thesis is organized as follows. More background information on recommendation system is provided in Chapter 2. This includes the research trend on recommendation systems, recommendation system applications and the techniques, evaluation metrics used in recommendation systems. In Chapter 3, we present the challenges in the recommendation system, especially for the privacy issue. In Chapter 4, we present our solution to the privacy issue, and we design a privacy-preserving recommendation system using a novel semi-decentralized network called TAN and differential privacy. In Chapter 5, we evaluate the recommendation system by using accuracy and F-measure evaluation metrics. Finally, in Chapter 6, we conclude the research and discuss the path for future research.

# Chapter 2

# Background

In this chapter, we provide the background of the recommendation system and some techniques that are relevant to this thesis. we first illustrate the culture of the recommendation system and its research trend. Then, we introduce the common recommendation process, and explain existing recommendation techniques. Lastly, we list some recommendation system applications and explain why recommendation system is important to us and businesses.

## 2.1 Recommendation System

The invention of the first modern recommendation system [GNOT92] can be traced back to 1992. The system is called "Tapestry" developed at the Xerox Palo Alto Research Center (PARC) which aims to deal with the dramatically increasing use of the email . At that time, the number of incoming emails was overwhelming and users were having a hard time finding out the most relevant topics or tasks. Tapestry

enables users to specify a filter to scan the email list and find the most interesting ones. In addition, Tapestry also supports collaborative filtering, which means people can collaborate to help each other to filter the emails by recording the preferences to the emails. Later, Resnick and Varian [RV97] defined such a system in a more general term called "Recommender System", and conducted a functional analysis of different recommendation systems using business models. According to Burke [Bur02], a recommendation system needs to provide personalized recommendations, the main difference between recommendation systems and information retrieval systems like Tapestry is the criteria of individualization or personalization. In a recent handbook on recommendation systems [RRS15], recommendation systems are described as software tools that are designed to help the users in decision-making processes, and the development of such tools requires effort from different fields such as computer science, statistics, and marketing.

## 2.2 Research Trend

Table 2.1 shows the rapid increase of interest in the recommendation system subject. The number of articles is generated by searching the ACM library using the keyword: "recommendation system". Most of the papers focused on improving the performance (e.g., increasing the recommendation accuracy) of algorithms such as content-based filtering, collaborative filtering and hybrid. The number of studies on content-based filtering has been decreased since 2010. Collaborative filtering and hybrid techniques become more and more popular in recent years [KLPC22]. We discuss the pros and cons of each technique in Section 2.3.

Table 2.1: Recommendation System Articles in Association for Computing Machinery (ACM) Library

| Years | Number of articles |
|---|---|
| 1992-1997 | 30,134 |
| 1998-2003 | 48,650 |
| 2004-2009 | 92,980 |
| 2010-2015 | 148,249 |
| 2016-2021 | 173,774 |

## 2.3 Recommendation System Techniques

Figure 2.1 shows the hierarchy of recommendation system techniques. On the top level, the recommendation system method can be classified into two main types: non-personalized recommendation and personalized recommendation. As suggested by the name, the *non-personalized* recommendation method does not take into account the user's tastes, the method is simply based on statistical summarization thus all the users get the same recommendation result. Such system is relatively easy to

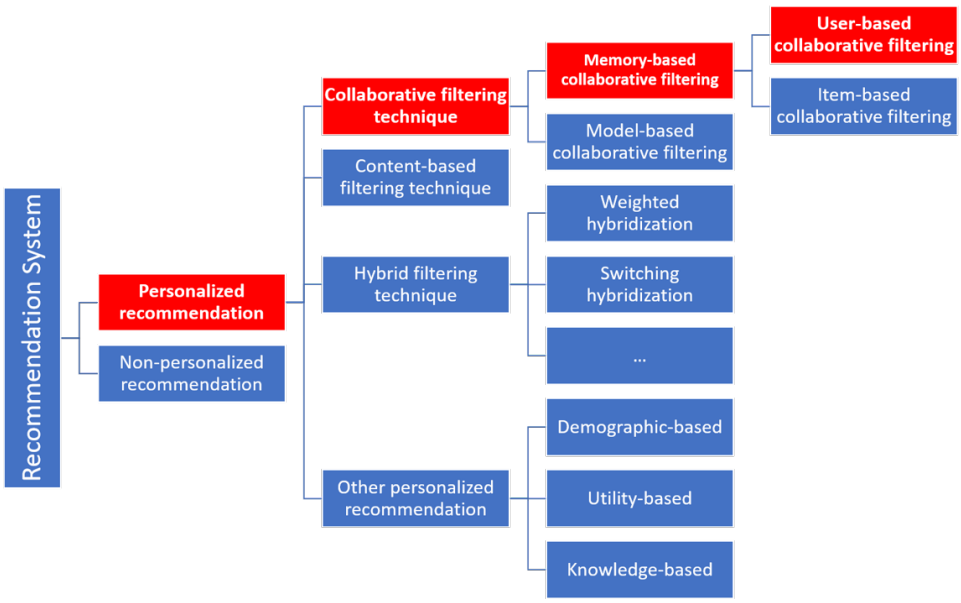


Figure 2.1: Recommendation System Technique Hierarchy

implement but might not appeal to the users, we are more interested in *personalized* recommendations in this thesis, especially user-based collaborative filtering technique that causes most privacy issue. We briefly discuss other techniques in the hierarchy and focus on the explanation of why collaborative filtering technique causes privacy leaks. Let us start with Burke's classification [Bur02] on recommendation techniques.

### 2.3.1 Burke's Classification

We assume that there are a set of users $U = \{u_1, u_2 \dots\}$ and a set of items $I = \{i_1, i_2 \dots\}$ in the system, and the goal of these personalized recommendation systems is to predict whether user $u_1$ likes item $i_1$ or not (or $u_1$'s rating on $i_1$).

1. *Content-based filtering (CBF)*

   In CBF approach, the items are recommended to $u_1$ based on $u_1$'s past activities (e.g., $u_1$'s ratings on items). CBF divides the items into several subsets (i.e., classification), then it infers $i_1$'s rating based on $u_1$'s ratings on other items that are in the same class with item $i_1$. We describe it in detail in Section 2.3.2.

2. *Collaborative filtering (CF)*

   In CF approach, the recommendation result (i.e., rating of $i_1$) is highly related to similar users' ratings on $i_1$. CF identifies a user set $S$ that contains users that are similar to user $u_1$, then it infers the $i_1$'s rating based on the ratings of $i_1$ that are given by the users in set $S$. We describe it in detail in Section 2.3.3.

3. Demographic

   Demographic-based recommendation is based on the hypothesis that people in

the different demographic class should be recommended differently. It is a simple but effective method, and a command example is that websites can display different contents based on the user's preferred language. More specifically, demographic-based recommendation identifies user set $DS$ that contains all the users that are demographically similar to user $u_1$, then infers the $i_1$'s rating based on the ratings of $i_1$ that are given by the users in set $DS$.

4. Utility-based

   Utility-based recommendation employs a utility function (linear or non-linear function) to calculate utility value for each user. For example, if $u_1$ is looking for a laptop that has a $2\times$ utility value for price and $3\times$utility value for the brand, then the system can easily calculate the total utility value for each available laptop and rank them, the item with the highest total utility value will be recommended to the user. Utility-based recommendation system is similar to CBF but the former explicitly calculates the utility values and the latter compares descriptive attributes.

5. Knowledge-based

   Knowledge-based recommendation has functional knowledge about how a certain item satisfies the user's need and usually requires knowledge engineering. To elaborate more, the system takes a description of $u_1$'s interests as an input and applies knowledge function to $I$ to get best-matched items to $u_1$. Knowledge-based recommendation is very similar to the utility-based recommendation but the expert (or service provider) defines the knowledge function in the former case, the user inputs utility function in the latter case.

In the recent state-of-the-art survey [PP20], the recommendation methods were narrowed down to three types: content-based filtering (CBF), collaborative filtering (CF) and hybrid.

### 2.3.2 Content-Based Filtering (CBF)

Content-based filtering (CBF) technique recommends items based on the user's profile, e.g., past activities or preferences. The recommendation can be only based on the user's profile, and the process can be done locally in the user's device which does not cause many privacy issues. However, one of the biggest disadvantages of CBF is that the recommendation results are highly related to the items that are already defined in their profile (e.g., the user already liked or disliked an item before), as a result, CBF only recommends similar items and barely allows users to explore new items. An example of content-based filtering movie recommendation system is shown in Figure 2.2. The example shows a set of users $U = \{User_1, User_2, User_3\}$ and

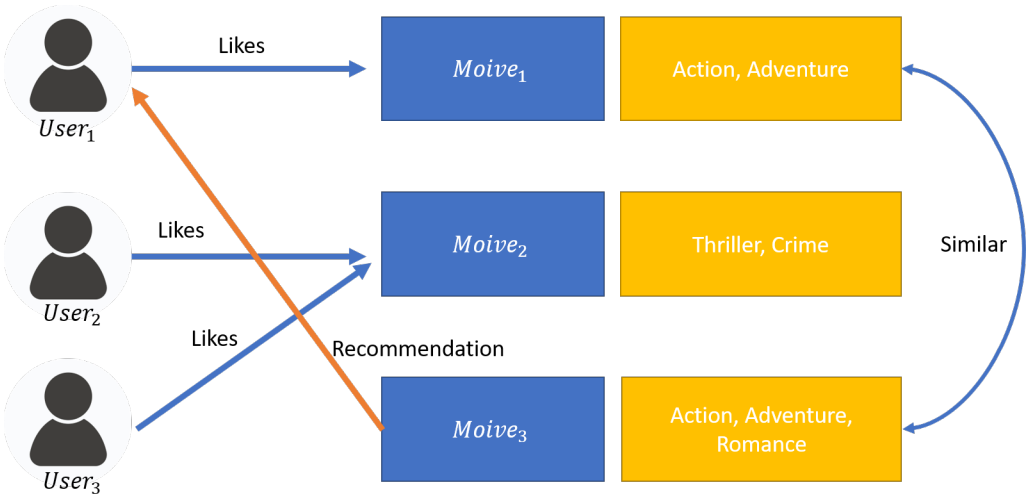


Figure 2.2: Content-Based Filtering (CBF) Example

a set of movies $M = \{Movie_1, Movie_2, Movie_3\}$. In the recommendation process, the system knows that $User_1$ likes $Movie_1$ and it searches for the movie that the genre is similar to the genre of $Movie_1$, as a result, $Movie_3$ is recommended to $User_1$. However, most of the users in the system (in this case, $User_2$ and $User_3$) like $Movie_2$. In other words, if $User_1$ never liked thriller movies before, CBF will never recommend thriller movies to $User_1$, so the user might miss potentially good movies.

### 2.3.3 Collaborative Filtering (CF)

Unlike CBF, CF is based on the concept that users with similar past preferences will have similar preferences in the future [SKKR01]. There are two main approaches in this technique: memory-based collaborative filtering and model-based collaborative filtering [PH12].

#### Memory-Based Collaborative Filtering

The memory-based collaborative filtering can be further classified into user-based collaborative filtering and item-based collaborative filtering. The user-based collaborative filtering calculates similarity among users by comparing the evaluation record (e.g., ratings) of the same items, and it then predicts rating for an item by the similar users' ratings on that item. On the other hand, the item-based collaborative filtering computes similarity among items instead of users, it calculates how similar the items in the database are to the target item, then selects top N similar items and based on the ratings on those items to predict the target item's rating. Figure 2.3 shows an example of how user-based collaborative filtering works. In the example, the system

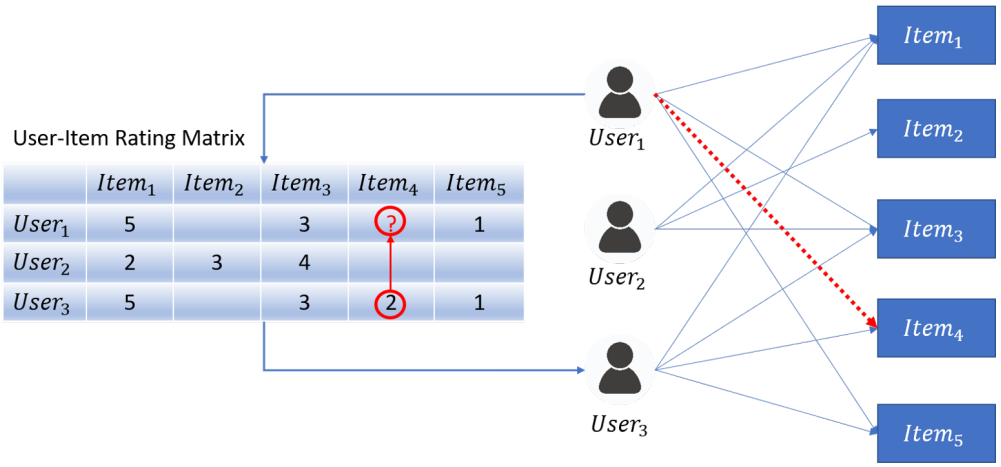


Figure 2.3: Collaborative Filtering (CF) Example

tries to predict the rating of $Item_4$ for $User_1$, the system first identifies that $User_1$ and $User_3$ made similar ratings for $Item_1$, $Item_3$ and $Item_5$ using user-item rating matrix. Then the system computes $User_3$ as the most similar user to $User_1$ and makes a prediction based on the rating of $Item_4$ that is provided by $User_3$.

There are a lot of similarity measurement techniques [Fki21], we describe the two most popular ones: Vector similarity (cosine), Pearson correlation coefficient (PCC).

1. Vector similarity (Cosine similarity)

   we assume there is an item set $I$ and a user set $U$ in the system, for each user, the ratings on items can be indicated as an n-dimensional vector, and Vector similarity (Cosine) is calculated as follow [CHC+18]:

$$sim_{xy} = \cos(x, y) = \frac{x \cdot y}{\|x\|\|y\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} (x_i)^2}\sqrt{\sum_{i=1}^{n} (y_i)^2}} \tag{2.1}$$

   where $\text{sim}_{xy}$ indicates the similarity value between user $x$ and $y$ and $x, y \in U$. $x \cdot y$ represents a dot product of two rating vectors of user $x$ and $y$. $x_i$ and $y_i$

represent the ratings of item $i$ by user $x$ and $y$ respectively.

2. Pearson Correlation Coefficient (PCC)

   Pearson correlation coefficient (PCC) is calculated as follow [KK17]:

$$sim_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2.2}$$

   where $\bar{x}$ and $\bar{y}$ indicate the average ratings by user $x$ and $y$.

### Model-Based Collaborative Filtering

In model-based collaborative filtering approach, the model is built using data mining or machine learning algorithms, then trained using existing data and to be used to predict unknown data. Some examples are (1) using association rule mining to find the patterns in the data set and make personalized marketing decisions [PL10], (2) Rathore et al. [RK16] used Decision Tree technique to implement a recommendation system that can predict software fault and (3) Ananda et al. [ANAB20] used one of the most famous clustering technique: K-means algorithm in course recommendation system to help students with the selection of the courses. Moreover, using machine learning algorithms [BP15; AKK+17; HAM21] to develop the recommendation model becomes popular in the past decade.

### Privacy Issue in Collaborative Filtering

Both memory-based collaborative filtering and model-based collaborative filtering can lead to privacy leaks. As shown in the previous user-based collaborative filtering example, the user-item rating matrix plays a vital role in the process, and recom-

mendation systems need to collect as much information as possible from the users to generate such matrix and provide good recommendations. Moreover, for machine learning and data mining approaches in model-based collaborative filtering, the data are usually stored in the server to reduce the cost of data communication and train the model efficiently which also raises privacy concern.

### 2.3.4 Hybrid Filtering

Comparing content-based filtering (CBF) with collaborative filtering (CF), they both have advantages and disadvantages. CBF does not raise many privacy issues as the prediction relies on the user's past experience, i.e., users in the system do not need to share much information. As mentioned above, however, it is hard for the users to explore new items. On the other hand, CF is able to utilize other users' information to recommend new items, but there are also limitations [IFO15]: data sparsity problem, cold-start problem. A hybrid recommendation system is designed to solve the limitations of both techniques by integrating information of CBF and CB [BH04]. According to Burke's survey [Bur02], there are seven types of hybrid recommendation model: Weighted hybridization, switching hybridization, cascade hybridization, mixed hybridization, feature-combination, feature-augmentation, and meta-level.

## 2.4 Recommendation System Applications

Recommendation systems are mainly applied into seven fields: streaming service, social network service, tourism service, e-commerce service, healthcare service, edu-

cation service, and academic information service [KLPC22]. On a daily basis, we are using the recommendation system applications all the time and we can easily come up with some well-known recommendation applications in those fields like Netflix, Facebook (renamed to Meta in 2021), and Amazon. Especially in an e-commerce setting, we can rarely find an e-commerce service provider which does not have purchasing recommendations. Recommendation systems are particularly helpful for indecisive people, and a good recommendation system not only enables users to quickly and correctly obtain the contents they need but also enables the service provider to increase customer loyalty and further improve their revenues.

## 2.5 Summary

In this chapter, we introduced the first information filtering system called "Tapestry" which was used to scan the email list and filter useful emails, and modern recommendation systems are motivated by it which enables people to quickly find the most useful information from a large number of choices. Moreover, we also showed a dramatic increase in research on recommendation systems in recent decades and the domains that recommendation systems are applied. We also introduced existing recommendation system techniques with Burke's classification that include collaborative filtering (CF), content-based filtering (CBF), demographic-based, utility-based and knowledge-based and provided the overall hierarchy of recommendation system techniques. As a preview, we use *user-based CF technique* in our recommendation system and propose a privacy-preserving approach of user-based CF technique. User-based CF recommends items based on the concept that users who have similar behavior in

the past will have a similar preference in the future, thus the service provider needs to collect the data from all the users which leads to privacy leaks.

# Chapter 3

# Related Works

In this chapter, we discuss issues in recommendation systems and related works on tackling privacy issues. Most of the recommendation system literature focused on improving the accuracy of the recommendations however, we argue that the privacy issue cannot be ignored, and our related works are mainly focusing on the literature that deals with privacy concerns. Firstly, we explain how big data accelerates the improvement recommendation system. Secondly, we list all the challenges in the recommendation systems. Then, we emphasize the privacy issue and introduce two main approaches to solve such issues. The first approach is data revision which includes cryptography and obfuscation, and the second approach is distributed recommendation. we mainly explain one of the data obfuscation techniques called differential privacy.

## 3.1 Challenges for Recommendation Systems in Big Data Era

In recent decades, the number of applications and research articles regarding recommendation systems has increased dramatically. The driving force behind that is an exponential increase in the volumes of data—Big data. Besides the aforementioned types of recommendation systems, there are also machine learning-based recommendation systems [KPAM19; AK20; TS22] which are impossible to implement without the concept of big data. Especially in popular 3 V's (Volume, Velocity and Variety) in big data [Hof17]—the volume of data plays a vital role when applying differential privacy. In fact, the techniques (e.g., collaborative filtering (CF), differential privacy) that are used in our privacy-preserving recommendation system (Trust-based Agent Network (TAN)) are based on the assumption that there are voluminous data available in the application. This assumption holds in the current big data era.

### 3.1.1 Privacy Issues

In this thesis, our focus is mainly on the issue of privacy leaks in recommendation systems. We mainly discuss privacy issue and existing solutions to the issue and briefly discuss some challenges to improve the accuracy of collaborative filtering recommendation system.

Recommendation systems predict users' preferences from the personal data that are uploaded from the users. In general, a tremendous number of data are required to achieve accuracy, and the data include both insensitive data (ratings, click logs) and

sensitive data (health condition, marital status), and these data are usually stored in one or more servers that are owned by service provider. Thus, there is a commonly accepted paradigm that the service providers are trusted or service providers only collect insensitive data from the users. However, the information that is assumed to be insensitive is also vulnerable to the leak of private data, Narayanan and Shmatikov [NS08] demonstrated a practical de-identification attack and successfully uncovered sensitive user information from the anonymous movie ratings of subscribers. They applied a de-anonymization methodology to the Netflix Prize dataset and successfully inferred the users' political preferences and other sensitive information by linking other datasets that are available online in the public. Besides the de-anonymization attacks from adversaries, the recommendation system provider itself is also a potential attacker, there is no guarantee that the provider will not use the private data for financial benefit. This brings great threats to user privacy.

Many studies have been done to address the privacy issue, and the solutions in existing research can be categorized into two major groups: raw data revision and distributed recommendation. Generally speaking, the former ensures privacy by applying some data protection techniques to the raw data and the latter ensures privacy by storing and computing the data locally.

### 3.1.2 Existing Approach 1 for Privacy Issue: Raw Data Revision

For the raw data revision approach, there are two main techniques to protect the data: cryptography and obfuscation. The *cryptography method* is based on mathe-

matical theory to study data encryption algorithms, Kim et al. [KKK+18] used fully homomorphic encryption to encrypt the users' rating data and return an encrypted output so that the adversaries or the RS cannot learn on the user's rating values.

The *obfuscation method* often imposes disturbance on the user's information to prevent individual users from being identified. Some existing works [BFK+15; FBK16; LYC20] applied differential privacy [DMNS06] to add random noises to the prediction results to prevent the attacks. As one of the well-known data obfuscation mechanisms, differential privacy (DP) applies a randomized algorithm to the raw dataset to prevent the adversaries from inferring individual information from the obfuscated dataset [Dwo08]. The idea behind DP is that any individual data in the large database are small enough, thus modifying any of them will not affect the overall dataset statistical information. The process of individual data modification is called adding noises to the data. By adding noises to the dataset, the probability of a query returns a same result from the real dataset and obfuscated dataset is within a bound of $\epsilon$ [Dwo06]. Formally, a random algorithm $M$ is differentially private if and only if it satisfies the following equation:

$$\frac{Pr[M(D) = O]}{Pr[M(D') = O]} \leq e^{\epsilon} \tag{3.1}$$

for any adjacent datasets $D$ and $D'$ and output $O$ where adjacent datasets mean that two datasets differ in at most one data entry. To elaborate, the difference between the probability of using randomized algorithm $M$ on a dataset $D$ to get output $O$ and the probability of using randomized algorithm $M$ on a dataset $D'$ to get output $O$ is less than or equal to $e^{\epsilon}$. In other words, modifying an individual data in dataset $D$ (which becomes $D'$) does not change the distribution of the output in a range of

$e^{\epsilon}$. Thus, from the output $O$, the adversaries cannot infer any individual information because (1) they do not know the output is from $D$ or $D'$, and (2) any individual information could have been changed when we change original dataset $D$ to $D'$. Using a randomized algorithm—bit-flipping as an example, Stanojevic et al. [SNY17] proved that random bit-flipping scheme satisfies $\epsilon$-differential privacy with probability $P = 1/(1 + e^{\epsilon})$, and assume the clients want to upload a set of data $S = (D_1, ..., D_n)$ where $D_n = 0$ or 1, clients can apply the bit-flipping algorithm with the value of P before uploading the raw set to the server and the distribution of 0's and 1's does not change significantly.

Further, randomized response algorithm [War65] also satisfies $\epsilon$-differential privacy. A typical example of randomized response is conducting a survey by asking sensitive questions, and most people do not want to reveal their answers to the public. For example, "Do you smoke?" is a sensitive question, and we can apply randomized response to this question to protect users' privacy, the process is as follow:

1. Toss a coin

2. if heads, submit the real answer

3. if tails, toss coin again and if heads, submit "yes", otherwise submit "no"

After collecting all the answers, we can infer the approximate distribution of the real answers because we know the probability in each step of the process. However, it is impossible to infer which answer is provided by which individual.

Due to its benefits on preserving user identity, We use randomized response as our data obfuscation method in our privacy-preserving system. We will elaborate more

in Section 4.4.

### 3.1.3 Existing Approach 2 for Privacy Issue: Distributed Recommendation

Regarding distributed recommendation solutions, Ma et al. [MML+18] proposed a decentralized framework called ARMOR for the friend recommendation system in online social networks. Recently, Yi et al. [YWW+21] proposed an efficient federated learning framework for news recommendation systems. *Federated learning* is a machine learning technique first proposed by Google [YLCT19], it protects privacy by allowing multiple clients (i.e., federation) to enter the model training process without sharing data directly. However, in a decentralized manner, the cost for data communication is very expensive and the cost for implementing federated learning is usually higher than processing it centrally.

Observing this drawback of distributed recommendation, we design a novel semi-decentralized network called Trusted-based Agent Network (TAN). It leverages both advantage of centralized and decentralized network, i.e., reduces data communication cost while protecting the data. We will discuss TAN in detail in Section 4.3

## 3.2 Other Challenges

This thesis aims to solve the privacy issue in collaborative filtering technique. However, there are also common issues in modern recommendation system applications. We will discuss potential solutions to these issues in Section 6.2.

### 3.2.1 Data Sparsity

Data sparsity is a common problem in most collaborative filtering based recommendation systems. Generally speaking, there are usually a lot of available items in recommendation systems, but most users only rate a few items, and this causes difficulty in finding similar users or items in the recommendation process [RRS15]. So, when rating data are sparse, the system can only use a very limited number of similar users' ratings to predict the current user's rating. Especially, in an e-commerce setting, only most of the popular items are rated, the rest of the items will be never recommended to the users resulting in biased recommendations.

### 3.2.2 Cold Start

Cold start refers to that recommendation systems are unable to make an accurate recommendation to new users, or new items are never recommended to the users due to insufficient data on them. One of the common solutions to this issue is that recommendation systems provide a survey on newly arrived users to have a basic idea of user's preference. For example, when a new user is registered in a movie recommendation system, the recommendation system can ask the user to choose from different genres of movies, create a basic profile for this user based on the user's choice.

### 3.2.3 Scalability

Due to big data, there are a tremendous number of users and items in the recommendation applications, thus a large amount of memory and computation power is

required to generate the recommendations.

## 3.3 Summary

In this chapter, we introduced two existing solutions to privacy issue: raw data revision and distributed recommendation. As a data revision technique, differential privacy is useful in preserving privacy. In particular, a differential privacy mechanism called *randomized response* ensures data privacy by adding noises to the real data so that the service providers cannot infer sensitive information from the data and the distribution of response does not change a lot. As such, we will adapt randomized response in our solution.

Moreover, observing disadvantage of the distributed recommendation solutions (i.e., the expensive cost of data communication), we plan to design a novel network called TAN such that it can benefit from both centralized and decentralized architecture. As a preview, our solution is a non-trivial combination of those two approaches.

For completeness, we also described several challenges in modern recommendation systems including data sparsity, cold start and scalability. As a preview, potential solutions are briefly discussed in Section 6.2.

# Chapter 4

# Personalized Privacy-Preserving Recommendation System

In this chapter, we propose our solution to tackle the aforementioned privacy issue. Firstly, we start with a common recommendation process to figure out the major steps that cause the privacy leaks, and propose a novel privacy-preserving recommendation process. Secondly, we provide a system overview of our proposed solution to indicate a general idea of our approach. Then, to implement the privacy-preserving solution, we introduce a novel semi-decentralized network called Trust-based Agent Network (TAN) and explain how TAN ensures data privacy. Furthermore, We show how randomized response, i.e., differential privacy, helps the system further protect users' privacy. Lastly, we discuss an edge case in our approach and propose a solution to it.

## 4.1 Motivation

Before designing the privacy-preserving recommendation system, we investigate the modern centralized collaborative filtering algorithm process to find which steps are vulnerable (i.e., privacy leaks):

1. Clients send recommendation requests to the server.
2. Server collects data from all the clients.
3. Server generates a user similarity matrix and computes the recommendation result based on the similar users' data.
4. Server sends back the recommendation result to all the clients.

The main problem is in Step 2 because we consider the server is suspicious, e.g., the service provider can use data for financial benefit. As a solution, a distributed recommendation can eliminate the server from the network, i.e., the user similarity matrix is distributed to all the clients using a peer-to-peer network. However, a trusted network should be considered because any node can be malicious in the peer-to-peer network.

Moreover, it is not necessary to find similar users throughout the whole network. Each client could have a local trusted network which is composed of trusted people or experts and we compute similar users inside a local trusted network, then only request a general recommendation from the whole network (e.g., how many people like this item?) to avoid the limitation of reduced search space. With this idea, we redesign a privacy-preserving recommendation process:

1. Each client has a local trusted network (a subset of the whole network), and a user similarity matrix is distributed inside each trusted network.

2. Each client computes recommendation result based on the user similarity matrix and obtains trusted recommendation.

3. Clients also send a request to the server to get a general recommendation throughout the whole network.

4. Server collects obfuscated data from all the clients.

5. Server sends a general recommendation result to all the clients.

6. Each client computes the final recommendation result based on both the local trusted recommendation and general recommendation.

As an example, I like a data mining book $B_1$, and I would like to buy a data mining book $B_2$. I will first ask people I trust and some experts in my trusted network. Then, my family members and my academic supervisor (Dr. Leung) tell me $B_2$ is a good book. I find that my Dr. Leung has similar interests with me and I listen to his recommendation. However, I also want to know how the other people that are not in my trusted network think of the book $B_2$, i.e., how many people like $B_1$ also like $B_2$? Then the service provider can provide me the result that is computed by using the obfuscated data received from all the people. Finally, I can make a final decision based on Dr. Leung's recommendation and the recommendation from the service provider.

## 4.2 System Overview

There are two main issues in traditional centralized CF: (1) users' real data are stored on the server, so the service provider can easily manipulate the data. (2) the recommendation from trusted people (e.g., family members, experts) is not taken into account and their opinions are often more important than the opinions of the public. The above privacy-preserving recommendation process resolves these two issues and the tasks we need to accomplish are:

T1: Designing a network that can consist of a server and a number of small local trusted networks.

T2: Applying data obfuscation technique to protect users' privacy, the obfuscated data still allow us to generate the whole network's recommendation.

For task T1, we design a semi-decentralized network called Trust-based Agent Network (TAN) to build trust and untrusted links. For task T2, we apply differential privacy to obfuscate the data. In TAN, data are distributed via trusted links in each local trusted network but obfuscated via untrusted links using randomized response in the global network. Each trusted network is a peer-to-peer network. We define the trusted recommendation—i.e., the recommendations provided by peers in the local trusted network—as *local view (LV)*. The LV is generated using user-based CF technique. As the recommendations provided by peers in a local trusted network may not provide sufficient information due to reduced search space, the *global view (GV)* is defined—the obfuscated data that are collected by the server. The GV is used to get a broader recommendation that is generated from the whole network, and the

final recommendation result can be calculated using following equation:

$$P_{ui} = \alpha \times GV + \beta \times LV \tag{4.1}$$

where

- $P_{ui}$ is user $u$'s recommendation (e.g., predicted rating) for item $i$; and
- $\alpha$ and $\beta$ are weights such that $0 \leq \alpha \leq \beta \leq 1$ because we care more about the recommendations inside local trusted network, both $\alpha$ and $\beta$ are non-negative numbers and $\alpha + \beta = 1$. Note that $\alpha \leq \beta$ and $\alpha + \beta = 1$ imply $0 \leq \alpha \leq 0.5$ and $0.5 \leq \beta \leq 1$. In extreme cases, when $\alpha = 0$, the final recommendation is purely based the trusted recommendations; when $\alpha = 1$, the final recommendation is purely made by global recommendations.

## 4.3 Trust-Based Agent Network (TAN)

In Trust-based Agent Network (TAN), we refer to each client as "agent", and the server as "Agent Server" (AS). An agent can be connected to another agent which forms a local trusted agent network, agent can also be connected to the AS via an untrusted connection. To elaborate, there are two types of connections in TAN: trusted connection and untrusted connection. The *trusted connection* is the connection in a trusted network, and the data are distributed via trusted connections to each agent which prevents the attackers to collect the real data. The generation of trusted connections can be done by sharing a secret code or scanning a Quick Response (QR) code in person and AS has no knowledge of the trusted connections.

The *untrusted connections* are the connections between agents and AS. In which, only obfuscated data are sent and received using untrusted connections, which prevent the service provider or other attackers to infer sensitive information from the real data.

In a traditional *centralized network*, all data are stored on a single node (server) and it is easy to collect data throughout the network. However, the data can be easily manipulated by the people who own the server (usually the recommendation service provider). In contrast, in a *decentralized network*, data are stored locally, users rather than server owners can have full control of their data thus ensuring data privacy. However, data communication is very expensive and inefficient. Hence, we propose TAN, which is a *semi-decentralized network* that is designed to incorporate both networks and benefit from both advantages:

- TAN can be considered as a *centralized network* in the sense that many clients are connected to a server using client-server-client architecture, and data are stored in the centralized server. In TAN, all the agents are connected to AS by untrusted connections, and AS collects data from agents then send back a GV to all the agents,

- TAN can be considered as a *decentralized network* in the sense that each client is directly connected to each other. Each client serves as a centralized server and communicates directly to another client in the local trusted network. In TAN, data are distributed in each local trusted network and all the agents can share data privately inside the local trusted network.

Using an example of TAN that is shown in Figure 4.1, we explain how TAN is generated, and how Agent1 gets the final recommendation result in TAN:

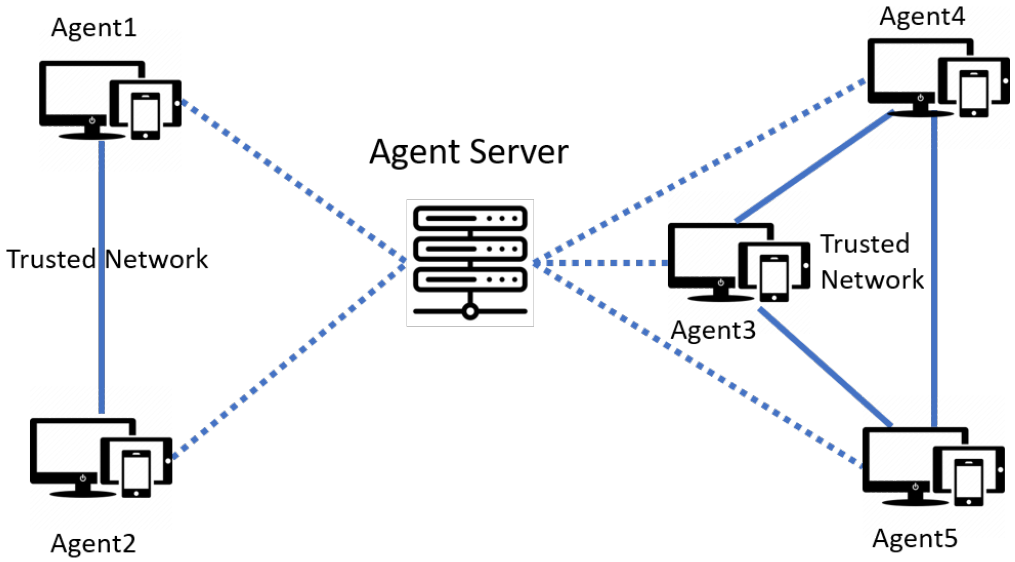


Figure 4.1: Trust-Based Agent Network (TAN)

1. Agent Server (AS) is created.

2. Agent1 enters TAN, then it creates an untrusted connection (a dashed line) between AS and Agent1, and it also creates a trusted connection (a solid line) to Agent2. For simplicity, there is only one trusted agent (Agent2) in this example but there can be many trusted peers.

3. Data are distributed inside each trusted network.

4. Agent1 computes LV from its trusted network. In this example, only Agent2 provides trusted recommendation but if there are many trusted agents, Agent1 compute LV based on a user similarity matrix.

5. Agent1 requests for a general recommendation from AS.

6. AS collects obfuscated data from all the agents, i.e., before all the agents upload

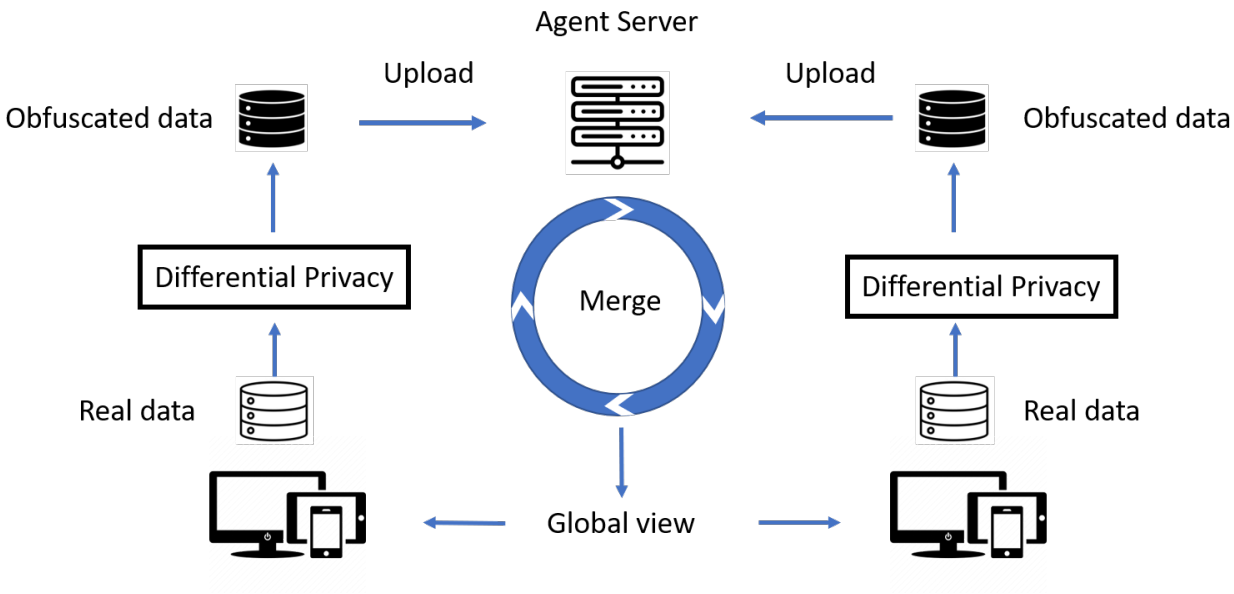


Figure 4.2: Use Differential Privacy to Upload Obfuscated Data

data (e.g., a set that contains ratings on movies), a differential privacy mechanism is adopted to the data. Figure 4.2 shows the process and we explain it in Section 4.4.

7. AS computes GV.

8. AS sends GV to Agent1.

9. Agent1 computes final recommendation result locally based on GV (received from AS) and LV (received from the local trusted network).

## 4.4 TAN+DP: TAN with Differential Privacy

Now, we are able to obtain LV using TAN, and next step is to use differential privacy to get GV, i.e., T2 mentioned earlier. Recall that GV is a global recommen-

dation from the whole network, e.g., how many people in the network like data mining book $B_1$ also like $B_2$. In common centralized recommendation systems, it is very easy to get the answer because the server can easily calculate the percentage using real data stored in it. However, the question is how to extract useful information from the obfuscated data uploaded by the agents in TAN.

**Algorithm 1:** Randomized Response

```
1  Function Randomization(a, p):
2      if rand() ≤ p then
3          return a
4      else
5          Toss a coin
6          if head then
7              return "yes"
8          else
9              return "no"
10         end
11     end
```

From Section 3.1.2, we know that randomized response satisfies $\epsilon$-differential privacy, we use the randomized response to obfuscate the real data and a pseudocode (Algorithm 1) is provided. The *Randomization* function takes $a$ and $p$ as inputs that represent a real answer and a user-defined probability respectively, and returns an obfuscated answer. Note that the input value of $a$ is "yes" or "no", and the return value is also "yes" or "no". When pick a random number from 0 to 1, if the number is less than or equal to $p$, then the function returns the real value $a$ as it. Otherwise, tosses a coin, and if the coin shows head, it returns "yes" otherwise it returns "no".

Let us use the example in Figure 4.3 to explain how this *Randomization* algorithm (Algorithm 1) obfuscates the responses and how we can infer GV from the obfuscated

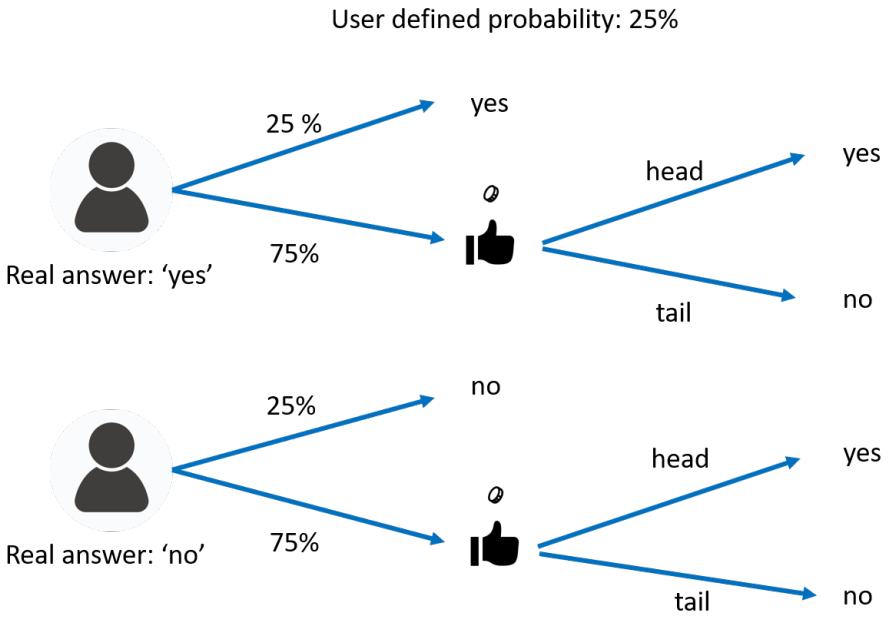


Figure 4.3: Randomized Response

data. The user-defined probability is set to $p = 25\%$ in the example, i.e., there is a 25% probability that the real data are not obfuscated:

Q3: How randomized responses obfuscate the real data? Can adversaries infer which response belongs to which individual?

A3: Because of the randomization, the adversaries can not infer the real answer from the obfuscated responses. For example, if the response is a "yes", there is no guarantee that the real answer is also a "yes" because there is a case that user's real answer was a "no", and it did not satisfy $p$ and the tossed coin showed head as well.

Q4: How to infer GV from the obfuscated data?

A4: Let us assume there are 100,000 responses with 45,000 "yes" and 55,000 "no".

We know that there are around 75% of 100,000 = 75,000 responses used coin-tossing. So, let us call them *fake responses*.

- Among those fake responses, approximately 50% (when using a fair coin) of these 75,000 = 37,500 are fake "yes" and fake "no". Then, we can infer that there are approximately 37,500 fake responses in 45,000 of "yes". Thus, the number of real "yes" can be computed by the difference between the number of "yes" responses and $\frac{1-p}{2}$ of all responses, i.e., around $45,000 - 37,500 = 7,500$ are real "yes".
- Similarly, using the same strategy, we can also infer the number of real "no" by the difference between the number of "no" responses and $\frac{1-p}{2}$ of all responses, i.e., around $55,000 - 37,500 = 17,500$ are real "no".
- Finally, we can calculate that there are approximately $\frac{7,500}{7,500+17,500} = 30\%$ of people answered "yes" and 70% of people answered "no". Note that even though we find out there are 7,500 real "yes" answers, we still cannot infer which 7,500 individuals provided a real "yes". Thus, the user's privacy is protected.

In Section 4.3, we listed the steps of how Agent1 gets LV inside its trusted network. Now, with randomized response, the process of how Agent1 gets GV is as follow:

1. Agent1 sends GV question to AS: How many people in the network like $B_1$ also like $B_2$?
2. AS sends the question to all the agents: Do you like $B_1$ and also like $B2$?

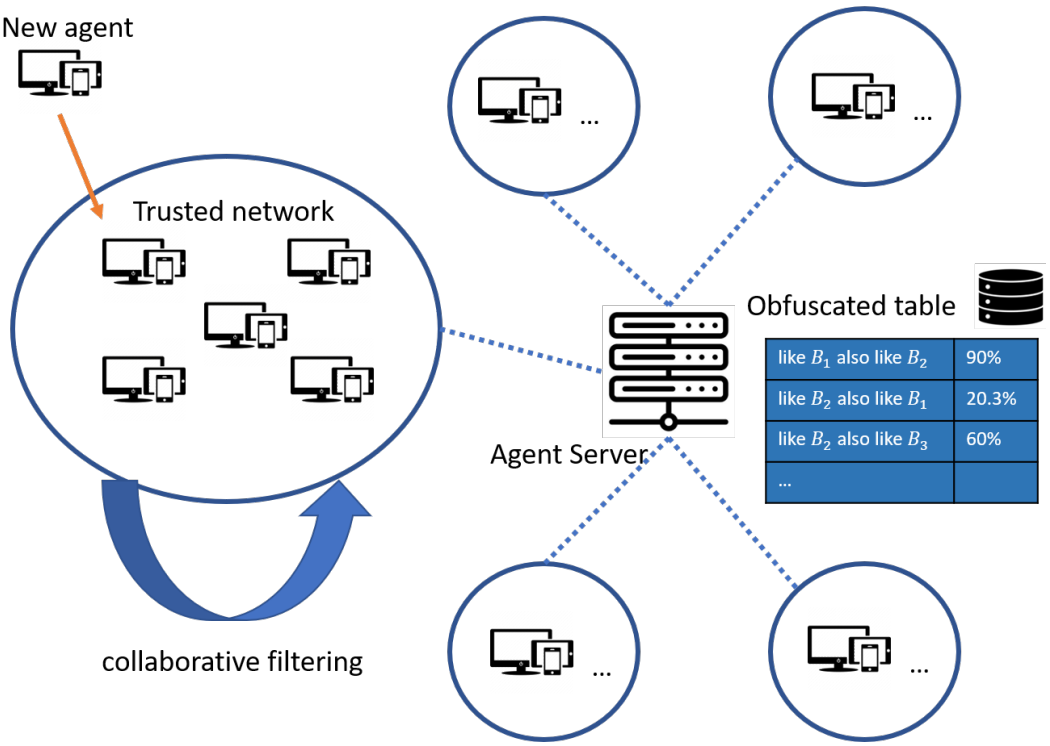


Figure 4.4: Privacy-Preserving Recommendation System

3. When an agent (e.g., Agent2) receives the question, if Agent2 does not have a record of the user's rating on $B_1$ or it does not like $B_1$ (e.g., low rating score), then Agent2 does not respond. Otherwise, Agent2 answers "yes" or "no" to the question based on the $B_2$'s rating given by the user.

4. Agent2 invokes *Randomization* function to the real answer and sends obfuscated data to AS.

5. AS merges data from all the agents and sends GV to all the agents. Note that there is an *obfuscated table* created to store GV, and it updates periodically. (Figure 4.4)

6. Agent1 receives GV from AS.

Finally, Agent1 can compute the recommendation result using Equation (4.1). As an example, assume that we set $\alpha = 0.3$ and $\beta = 0.7$, and the rating range in the system is set to 0 to 100. Recall that we set $0\% \leq \alpha \leq \beta \leq 1$ because we care more about the recommendations from the users inside the trusted network. Then, say the trusted recommendation LV is 100 (i.e., generated in local trusted network using user-based CF) and the GV from obfuscated table is 90, then the predicted Agent1's rating on $B_1$ is $\alpha \times GV + \beta \times LV = 0.3 \times 90 + 0.7 \times 100 = 97$.

Now, we consider an edge case in the above solution. As mentioned in Section 3.1, the differential privacy only works with big data, if there are not many ratings in the GV, then the distribution (e.g., the percentage of likes and dislikes) in obfuscated data will be not accurate. To deal with this issue, we add an additional step in TAN+DP.

## 4.5 TAN+DP: An Additional Step when Number of Ratings is Insufficient

Recall that when AS asks a question to all the agents, and if an agent has not rated the item, then the agent will not respond to the question. This could result in only small number of users reply to AS. In fact, it is common that there are not many ratings on each item in real-life applications except for those very popular items. As a solution, when the number of ratings in the GV is very small, AS asks multiple responses from agents who have rated the item. To elaborate, the process is as follow:

- AS sends the question to all the agents: Do you like $B_1$ and also like $B2$?

- AS only receives a few answers (say $n$ answers).
- AS sends the same question again, and asks those $n$ agents to reply multiple times (say $m$ times) using randomized response.

Note that the distribution in the real data does change with multiple responses, only the number of responses is increased to support randomized response. For example, if $n = 100$ and 70 people in the global network likes $B_1$, i.e., GV equals 70. With $m = 10$, the number of people likes $B_1$ is 700 and GV still equals 70. However, the downside of this additional step is the extra data communication cost, and we want to avoid this as much as possible.

## 4.6 Summary

In this chapter, we proposed our privacy-preserving solution. We first carefully went through all the steps in the common recommendation system process and found the privacy leakage, then we designed a novel recommendation process to protect users' data. The idea is that users can send obfuscated data to the service provider but the obfuscated data can still be used to get a general idea of the whole network's thoughts. We also argued that the recommendation should focus more on trusted people and less on untrusted people. So, we proposed Equation (4.1) that used rating strategy to assign more weights on LV.

We also explained two major techniques, i.e., Trust-based Agent Network (TAN) and randomized response, that are used in our solution. TAN is a semi-decentralized network. The centralization property of TAN allows the network to contain a central

server (Agent Server) which can easily collect from all the agents. Moreover, the decentralization property of TAN allows agents to have their own local trusted network, this ensures the data can be distributed securely inside the trusted network. When considering AS is suspicious in the sense that it can gain financial benefit by selling some private information to third parties, the randomized response enables users to add noise to the data so that the server cannot infer any individual information from received data. While this is sufficient to protect the users' privacy in big data applications, we also considered the case with insufficient number of ratings. We proposed an additional step to our solution that is collecting multiple responses from users who have rated the item to ensure the volume of ratings in GV.

# Chapter 5

# Evaluation

In this chapter, we evaluate the proposed privacy-preserving recommendation system using a real-world dataset. Firstly, we explain the dataset that we have obtained and the process of data exploration and data preprocessing Secondly, we introduce the evaluation methods that will be used in the experiments. Then, we observe the evaluation results to answer a few questions including how data privacy is preserved? how accurate the prediction is? and how efficient is the system? Lastly, considering the data sparsity issue in real-life applications, we conduct an experiment to see how the number of ratings impact the randomized response algorithm.

## 5.1 Dataset Acquisition

While an ideal approach to fully evaluate the proposed privacy-preserving recommendation system would be to implement and scale such system, and conduct user experience surveys, an alternative and practical approach is to collect a real-world

Table 5.1: A Tuple in LibraryThing

| userId | itemId | stars | social relation | comment |
|---|---|---|---|---|
| van_stef | 3206242 | 5.0 | Mishalak | This a great book for young readers... |

Table 5.2: Basic Statistics of LibraryThing

| | users | items | ratings | connections |
|---|---|---|---|---|
| LibraryThing | 83,083 | 504,230 | 1,707,070 | 219,790 |

dataset that contains (1) massive amount of users' reviews and (2) trusted connections between users, then run a simulation of our system. There are many existing datasets contain a good number of users and reviews but there are very few datasets that contain the measurement of trustworthiness regarding connections. To the best of our knowledge, from literature reviews [ZMK15; CHM17] and online search, we obtain a relatively large volume of a dataset that contains both connections and ratings. The data are generated from LibraryThing[1] that includes users' ratings on books and social connections between users. Although the connections are not necessarily trusted, other existing available datasets are either too small or have no connections. Table 5.1 shows an example row in the dataset, there are other columns in the original dataset but we eliminate them because of redundancy and irrelevance. Table 5.2 shows basic statistics. There are around 83,000 users rated on 500,000 items. The total number of ratings is 1.7 million and the total number of social connections is 220,000.

## 5.2 Data Exploration and Preprocessing

Figure 5.1 shows the distribution of ratings given by users. The histogram on the top of the graph indicates the distribution of ratings, and the histogram on the right

[1] `https://www.librarything.com/grouplens.org/datasets/movielens/`

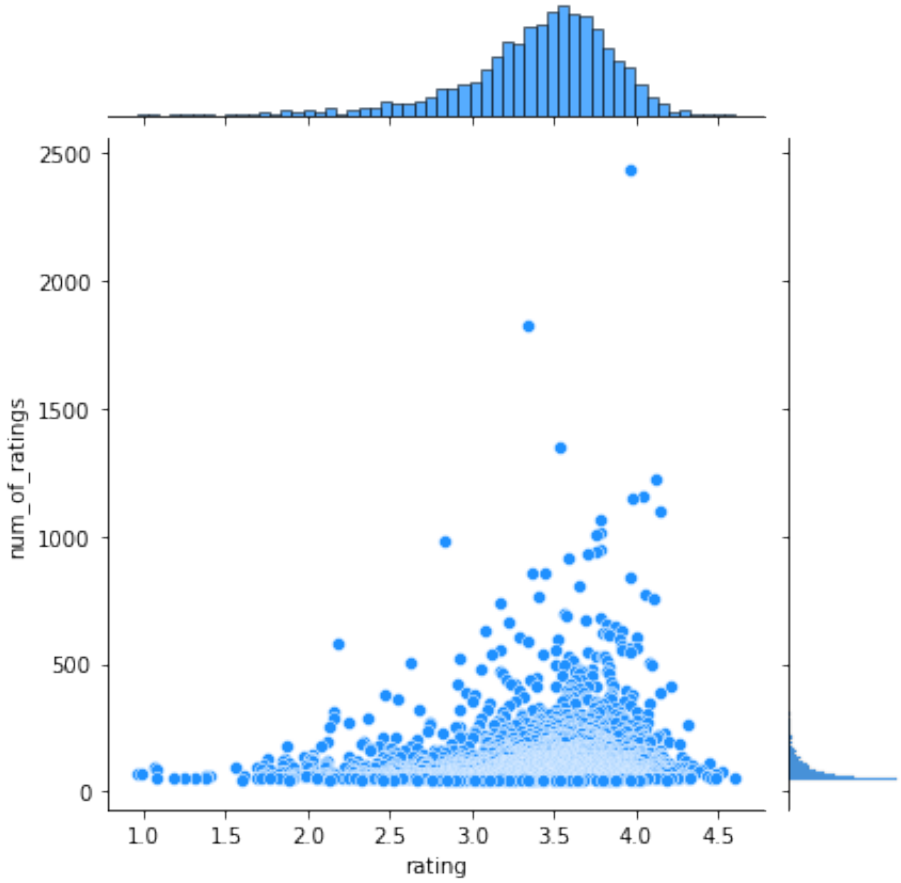


Figure 5.1: Distribution of Ratings in the LibraryThing Dataset

side indicates the distribution of the number of ratings on each item. More than 80% of ratings are in the range of 2.5 to 4.0 and the average of the ratings is 3.11. The largest rated item has 2,439 ratings, and the average number of ratings on each item is 20.5 which is not big enough to support the randomized response, so we will take an additional step in TAN+DP for the rest of the evaluation, i.e., multiple responses from the agents.

In the data preprocessing phase, we have two major goals: (1) eliminate unnecessary records and only keep useful information to increase the efficiency of the

evaluation process, (2) based on the filtered dataset, select users that have sufficient information to avoid data sparsity issue. In general, we only keep the users that satisfy minimum testing requirements, e.g., users have at least two ratings and at least one connection with others. We set two filters to achieve the major goals: general filter and testing filter: the general filter is used to clean up the dataset and the testing filter is used to generate the users who have sufficient ratings.

### 5.2.1 General Filter

The general filter is set up to reduce the memory usage and save evaluation time. We first delete the columns that will not be used in our evaluation process. Also, we eliminate rows that contain outliers, for example, the range of rating should be 0 to 5 and rows that contain negative ratings should be removed.

### 5.2.2 Testing Filter

We also use the testing filter to generate the user-item rating matrix that is used in collaborative filtering (Figure 2.3). The size of the user-item rating matrix is restricted because of the memory limitation, so we define our testing filter to fit it into the memory but still keep all useful information:

- Keep users that have at least one similar user with a similarity score above 0.6. The similarity score is calculated using Equation (2.1), i.e., cosine similarity.
- Keep users that have more than 10 ratings.
- Keep users that have at least one connection with other users, and the connected

user has rated at least one same item as the user.

Using the general filter and the testing filter, we obtained 3992 matching users in total, and we randomly select 100 users for each test.

## 5.3 Evaluation Metrics

### 5.3.1 Root Mean Square Error (RMSE)

Root Mean Square Error (RMSE) is a commonly used statistical accuracy metric and it is computed as follow [CS00]:

$$RMSE = \sqrt{\frac{1}{n}\sum(P_{ui} - A_{ui})^2} \tag{5.1}$$

where $n$ is the total number of ratings, $P_{ui}$ is user $u$'s predicted rating on item $i$ and $A_{ui}$ is actual rating on item $i$ given by user $u$. In general, RMSE can be thought of the distance between the predicted value and actual value, the lower RMSE indicate a more accurate recommendation. We will use RMSE to define the weights on LV and GV is in Section 5.4.

### 5.3.2 Accuracy and F-measure

F-measure is one of the most popularly used decision support accuracy metrics, it views accuracy prediction as a binary operation, and summarizes a binary classification table called confusion matrix. The confusion matrix often includes True Positive (TP), False Positive (FP), False Negative (FN), True Negative (TN) and we defined

Table 5.3: Rating Confusion Matrix

| | Actual rating$\geq$ 3.11 | Actual rating $<$ 3.11 |
|---|---|---|
| Predicted rating $\geq$ 3.11 | True Positive (TP) | False Positive (FP) |
| Predicted rating $<$ 3.11 | False Negative (FN) | True Negative (TN) |

them in Table 5.3. In our experiment, the preference of a user on an item is a binary variable. As the average rating in the dataset is 3.11, we define the positive class as "rating is greater than or equal to 3.11" and the negative class as "rating less than 3.11", to elaborate our rating confusion matrix:

- True Positive (TP) means the model correctly predicted positive values. In our experiment, both the predicted rating and the actual rating are greater than or equal to 3.11, i.e., we correctly predicted the user likes the item.

- False Positive (FP) means the model incorrectly predicted positive values. In our experiment, the predicted rating is greater than or equal to 3.11 but the actual rating is less than 3.11, i.e., fail to predict the user's preference.

- False Negative (FN) means the model incorrectly predicted negative values. In our experiment, the predicted rating is less than 3.11 but the actual rating is greater than or equal to 3.11, i.e., fail to predict the user's preference.

- True Negative (TN) means the model correctly predicted negative values. In our experiment, both predicted rating and actual rating are less than 3.11, i.e., we correctly predicted the user does not like the item.

**Precision**

The formula to calculate precision is given in Equation (5.2), The denominator is the sum of TP and FP which tells us the total predicted positive. So, precision is the ratio of the number of TP to the total number of predicted positive.

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

**Recall**

The formula to calculate recall is given in Equation (5.3), We can notice that the sum of TP and FN is equal to the total number of actual positives, so recall is the ratio of the number of TP to the total number of actual positive.

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

**Accuracy**

As the name indicates, accuracy represents how accurately the model predict users' preference. More specifically, Accuracy can be defined as the total number of correct predictions divided by the total number of all predictions and the formula is as follow:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{5.4}$$

**F-measure**

Finally, F-measure is commonly used when we want to find a balance between precision and recall, it tells how precise and robust the classifier is in the prediction. There are many ways to get the F-measure depending on how many weights are assigned to precision and recall in the calculation. A traditional F-measure is the harmonic mean of precision and recall as follow:

$$F\text{-}measure = \frac{2PR}{P+R} \tag{5.5}$$

where $P$ indicates precision and $R$ indicates recall.

## 5.4 Settings on the Weights for Global View (GV) and Local View (LV)

In Section 4.2, we mentioned that our recommendation is a combination of GV and LV, and the recommendation result is calculated using Equation (4.1) where $0\% \leq \alpha \leq \beta \leq 1$ and $\alpha + \beta = 1$. As we know that the connections in the dataset are not necessarily trusted, so we want to see how bad or good the LVs in the dataset are. Figure 5.2 shows the relation between RMSE and $\alpha$. We observe that, when $0 \leq \alpha < 0.5$, the RMSE does not show much difference, it stays in the range of 0.92 to 0.95. So, theoretically, we can define $\alpha$ as any number between 0 and 0.5. For the rest of the evaluation, we will define $\alpha$ as 0.25 which is the median number and $\beta$ as 0.75.

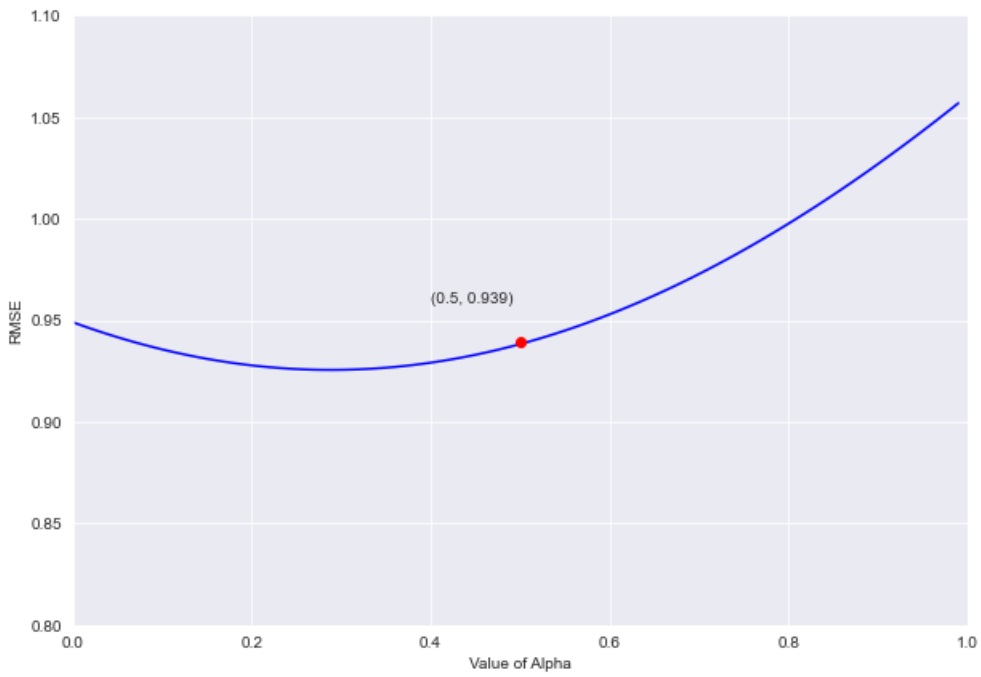


Figure 5.2: Relation between RMSE and $\alpha$

## 5.5 Observations

We conduct our experiments to answer the following questions:

Q5: How is privacy preserved?

Q6: How accurate is the privacy-preserving approach compared with non-privacy-preserving approach?

Q7: How efficient is the privacy-preserving approach compared with non-privacy-preserving approach?

We use the centralized user-based CF as our baseline. For Q5, we observe the data on the server side. For Q6 and Q7, we compare the accuracy, F-measure and run time complexity of our solution with CF.

### 5.5.1 Privacy Preservation

As we consider the service provider is suspicious, we represent the data stored in traditional centralized server and the data stored in Agent Server in TAN. As shown in Figure 5.3, the server has full access to all the users' information in traditional approach. Also as mentioned in introduction, with the linkage attack [NS08], the attackers from third parties could easily infer sensitive information of these users by combining users' ratings with other existing datasets that are available in the public.

| userId<br>itemID | Cicero- | Eva- | 06nwingert | 1212bec | 1983mk | 19vatermit64 | 1dragones | 1morechapter | 20XJenny | 211Fern | ... | zhejw | zibilee | zimbawilson | zinf | ziska |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **39** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN |
| **109** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN |
| **113** | NaN | NaN | 5.0 | NaN | NaN | NaN | NaN | 4.0 | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN |
| **115** | NaN | 4.0 | 5.0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | 5.0 | NaN |
| **118** | NaN | NaN | 5.0 | NaN | NaN | NaN | NaN | 4.0 | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **13480632** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN |
| **13480686** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN |
| **13588089** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN |
| **13805455** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN |
| **14038411** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 5.0 | NaN | ... | NaN | NaN | NaN | NaN | NaN |

3511 rows × 6663 columns

Figure 5.3: Data Representation in Traditional Server

However, in TAN (Figure 5.4), the Agent Server only receives the likelihood of each users in binary representation (e.g., 1 represents the user likes the item and vice versa), and stores the record in obfuscated table. Note that the users "mashley" and "ovistine" do not like item "115", and they responded a "yes" to the AS. Moreover, some users like "1212bec" does not reply as she or he does not have a record on item "113" or "115". As a result, either the service provider (in this case, AS) and third party attackers cannot infer individual information from existing data received and stored in AS.

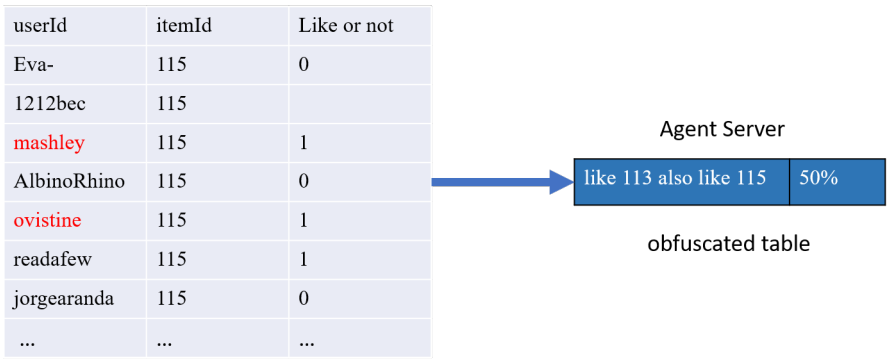


Figure 5.4: Data Representation in Agent Server in TAN

Table 5.4 shows service providers' accessibility on different data in different solutions. The traditional server stores user-item rating matrix locally thus the service provider has ability to access users' entire information, i.e., users' privacy is not protected at all. However, in TAN, the user-item rating matrix is only distributed in local trusted networks, and users' ratings are also protected because the Agent Server (AS) only collects preferences from all the agents, but AS still can track user ids and preferences. Finally, adding differential privacy in data communication in TAN, i.e., TAN+DP, data privacy is fully preserved as all the data AS have received are obfuscated, and AS can not access or infer any individual information from the data.

Table 5.4: Service Provider Accessibility on Data

| prevent from ... | Traditional server | TAN AS | TAN+DP AS |
|---|---|---|---|
| ... accessing user-item rating matrix | × | √ | √ |
| ... tracking user ratings | × | √ | √ |
| ... tracking user ids | × | × | √ |
| ... tracking user preferences | × | × | √ |

### 5.5.2 Accuracy and F-measure

We separate the privacy-preserving solutions to two cases: with DP and without DP. In other words, we first observe if TAN can provide a good prediction accuracy and F-measure, then we add DP to TAN and observe again. To elaborate, the following cases are tested:

- Privacy is not preserved: Use traditional centralized recommendation system, i.e., pure collaborative filtering to predict users' ratings.
- Privacy is preserved:
  - Use TAN to predict ratings, i.e., a combination of local trusted network recommendations (LV) and global recommendations (GV).
  - Use both TAN and randomized response (i.e., DP) to predict users' ratings.

The prediction accuracy and F-measure scores are shown in Figure 5.5 and Figure 5.6 respectively. In each graph, y-axis shows the scores for each approach, and the x-axis is the test number. Each colored bar indicates that:

- CF: privacy is not preserved.
- TAN: privacy is preserved with TAN.
- TAN+DP: privacy is preserved with TAN and DP.

The detailed accuracy numbers are represented in Table 5.5, in general, the accuracy scores for three cases are above 85% in all the tests. Let us first compare the cases without differential privacy, i.e., CF and TAN. The average prediction accuracy

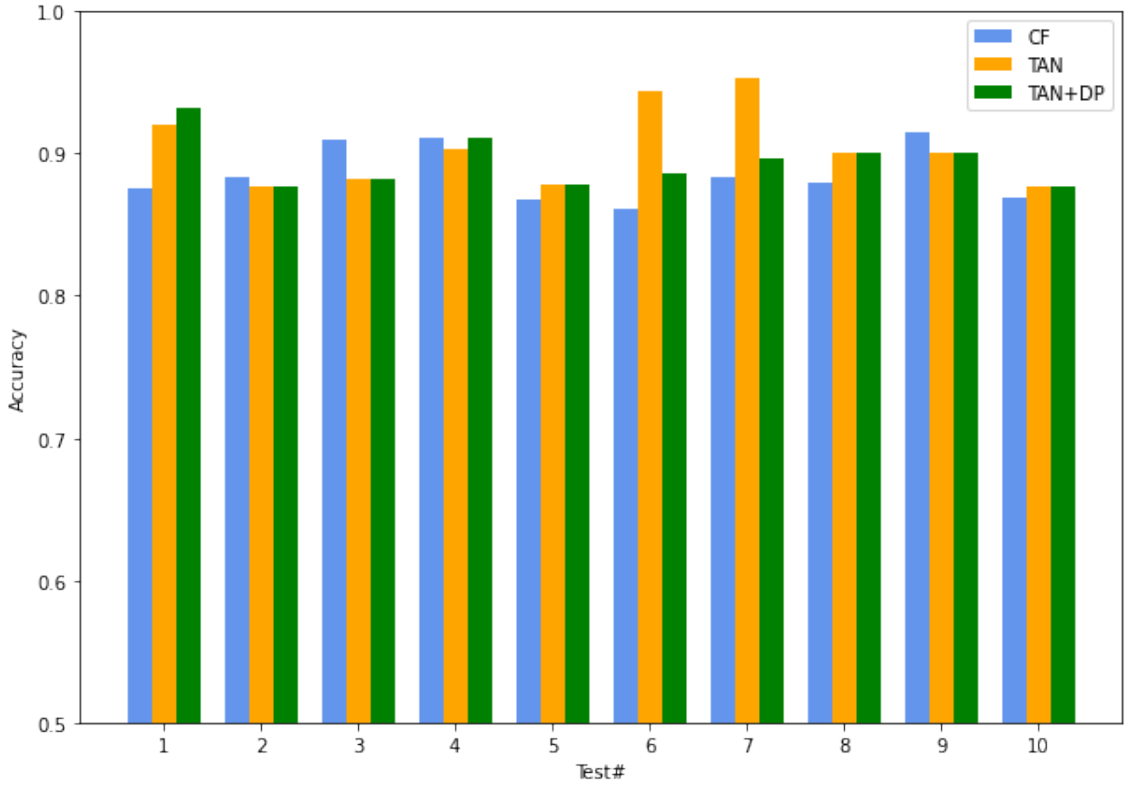


Figure 5.5: Prediction Accuracy

Table 5.5: Prediction Accuracy Comparison

| | CF | TAN | TAN+DP |
|---|---|---|---|
| 1 | 0.8750 | 0.9198 | **0.9218** |
| 2 | **0.8830** | 0.8768 | 0.8768 |
| 3 | **0.9098** | 0.8816 | 0.8816 |
| 4 | **0.9115** | 0.9026 | **0.9115** |
| 5 | 0.8677 | **0.8783** | **0.8783** |
| 6 | 0.8610 | **0.8867** | 0.8854 |
| 7 | 0.8828 | **0.8960** | **0.8960** |
| 8 | 0.8791 | **0.9003** | **0.9003** |
| 9 | **0.9144** | 0.9000 | 0.9000 |
| 10 | 0.8695 | **0.8763** | **0.8763** |
| average | 0.8854 | **0.9033** | 0.8928 |

of TAN is 90.33% and the average prediction accuracy of CF is 88.54% which indicates that TAN outperforms CF in general. There are four cases (Test2 ,Test3, Test4 and Test9) that CF shows higher accurate results, recall that the recommendation result using TAN is a combination of GV and LV, and we set $\beta$ as 0.75 so the result of prediction is highly related to the ratings given in local trusted network. As the connections in LibraryThing dataset is not necessarily trusted, the connections' ratings vary. So, we expect that, if we have highly related ratings from trusted connections, TAN will outperform CF in almost all the cases. That is because the users in trusted connections are experts or trusted users, and they will usually provide more useful recommendation than random similar users in the global network.

Now, we add DP to TAN and do the comparison, i.e., CF and TAN+DP. The average prediction accuracy of TAN+DP is 89.28% which also outperforms CF, and we only have a slight decrease comparing with TAN. In other words, applying randomized response does not affect the accuracy much, and we can achieve a good level of prediction accuracy with our privacy-preserving approach comparing with non-privacy-preserving approach.

One interesting observation is that, although we add DP to TAN, TAN+DP outperforms TAN in some cases (Test1 and Test4). Recall that randomized response adds noises to the real data, and those noises sometimes benefits the prediction result. For example, if the actual rating is 3.2, and we predicts a rating of 3.1 using TAN which is a FN case. With randomized response, the prediction result could fluctuate in some range (say 2.95 to 3.25), if it results in 3.25, then TAN+DP will have a TP case. In fact, in Figure 5.6 which indicates the scores of F-measure, we

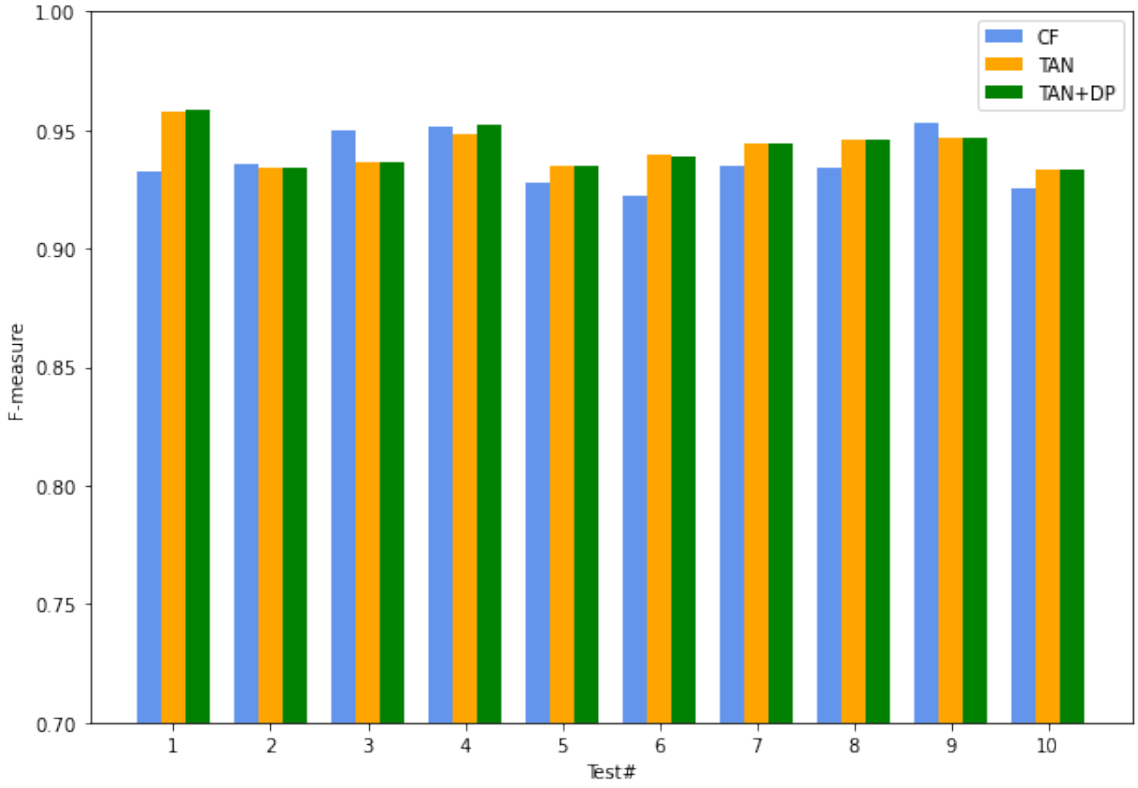


Figure 5.6: F-measure

can clearly observe the above example. That are in Test1 and Test4, TAN+DP has a higher F-measure scores compared with TAN which means TAN+DP predicts a higher number of TP or TN than TAN. Theoretically, with infinite numbers of test, we expect that TAN+DP will be more accurate than TAN in approximately half of the test cases, in other words, adding noises can make the prediction worse but it is also possible that it can benefit the prediction. The result of F-measure scores for each case is not surprisingly interesting, TAN+DP and TAN still indicate higher F-measure scores than CF in most of the cases while CF shows a better F-measure scores in some cases as well.

### 5.5.3 Run Time Complexity

We compare the run time complexity using two baselines: online and offline CF. The difference between them is whether the user-item rating matrix is computed in a real-time manner or not. Generally speaking, the user-item rating matrix should be generated every time CF finds similar users because users' preferences can be updated frequently. However, the computation cost of generating global user-item rating matrix is very expensive, and service provider usually computes and stores the global user-item rating matrix locally and updates it periodically. In TAN, CF is only used within local trusted network with much smaller space and the computation time is much less than the traditional one. However, the additional cost of getting GV should be considered. So, we do run time comparison in following two cases:

- Compare offline CF with TAN+DP
- Compare online CF with TAN+DP

The overall running time comparison of online CF, offline CF and TAN+DP is shown in Figure 5.7. To elaborate, if the user-item rating matrix is computed every time when calculating the users' similarities, i.e., the online CF, the run time complexity is $\mathcal{O}(n^2)$ where $n$ is the number of users, because the algorithm needs to go through the user list to find the top similar users once for each user. In contract, if the user-item rating matrix is computed once and updated periodically, i.e., the offline CF, the run time complexity is $\mathcal{O}(n)$ which equals the number of users. In privacy-preserving approach, i.e., TAN+DP, the run time complexity is $\mathcal{O}(tn)$ where $t$ is the number of trusted people in the LV which is usually a constant number.

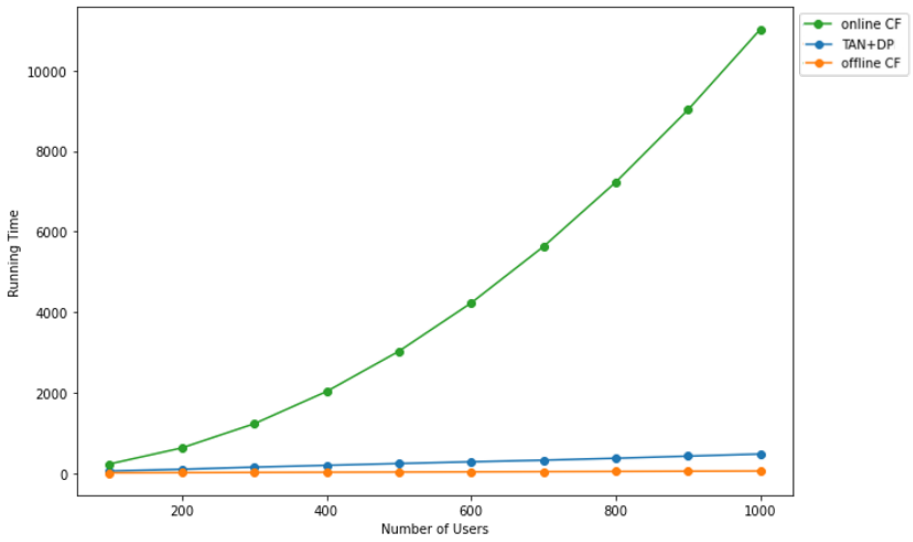


Figure 5.7: Offline CF, Online CF and TAN+DP Running Time Comparison

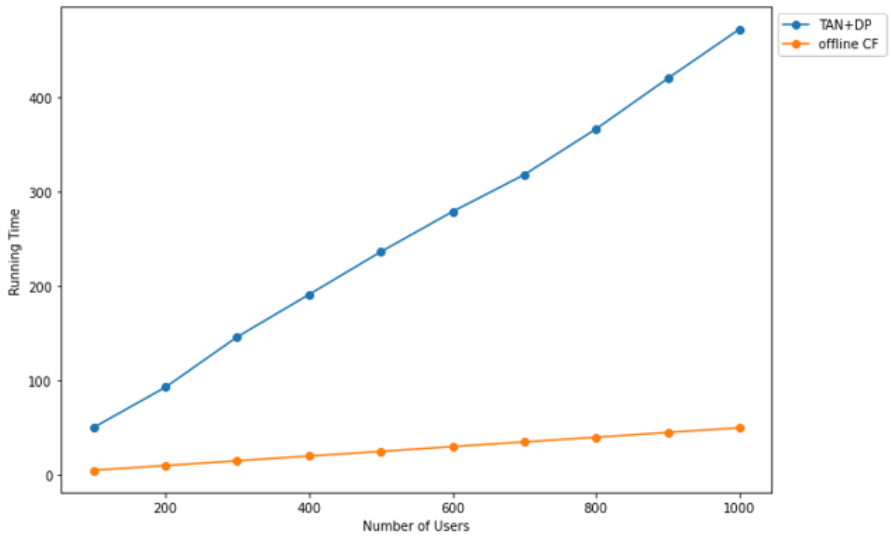


Figure 5.8: Offline CF and TAN+DP Running Time Comparison

In the worst case, the run time complexity of TAN+DP is as same as online CF, i.e., exponential, that is when all the user are connected to each other with trusted connections. However, more realistic case is that users will only connect to a few number of trusted people, i.e., $t$ is a constant number. So we can say that the run

time complexity of TAN+DP is also linear, i.e., $\mathcal{O}(n)$. Figure 5.8 shows the a clear comparison of the offline CF and TAN+DP which zooms in on Figure 5.7.

## 5.6 When to Take the Extra Step in TAN+DP?

Recall that we took an extra step in the above evaluation because of the insufficient number of ratings on each item, and this extra step requires an addition request sent by AS to all the agents which increase the data communication cost, so we want to avoid it as much as possible. It will be a good idea to see how bad TAN+DP without the additional step performs with small number of ratings. Figure 5.9 shows the top-rated items, most of them are rated less than 1,000 times. With this information in mind, we compare TAN+DP without the additional step to others which is shown

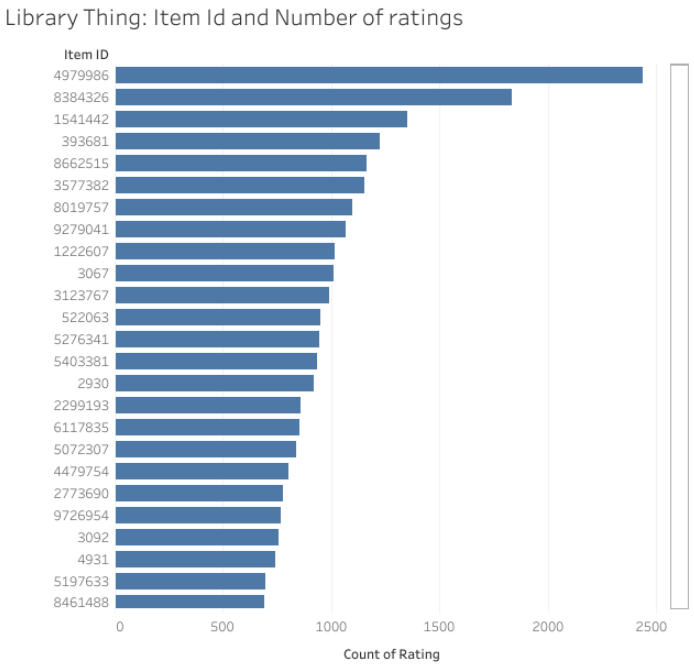


Figure 5.9: Number of Ratings in Descending Order

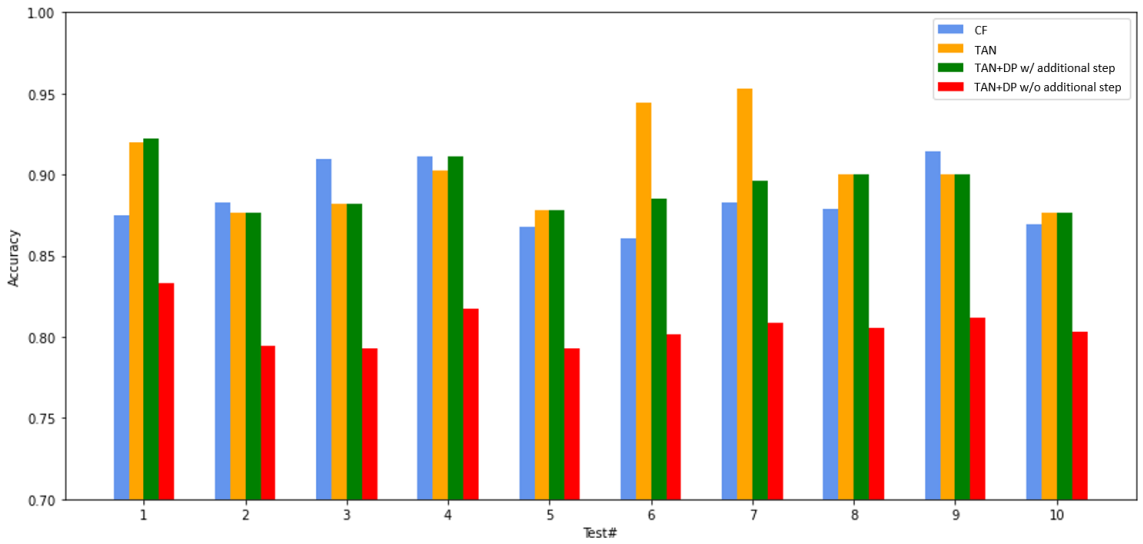


Figure 5.10: Compare TAN+DP without the Extra Step to Others

in 5.10. We can observe that there is a dramatic decrease in accuracy, so we want to find how many number of ratings on an item is sufficient to support the randomized response so that we can avoid the additional step.

For simplicity, we created an integer array of a size of N, and assigned 70% of 0's and 30% of 1's, and consider this array as the users' real responses, i.e., 0's real distribution is 0.7. Then we apply randomized response and backtracking, and run the test 500 times for different number of ratings. Figure 5.11 shows predicted 0's distribution frequencies for $N$ number of ratings. We define max error rate (MER) as $max$(predicted – real) which is the maximum gap that the prediction can have comparing with the real data distribution, in other words, MER shows that in the worst case, how bad is the predicted data distribution. For example, when $N = 500$, then worst prediction is 0.932 and $MER = 0.932 - 0.7 = 0.232$. We can observe that when $N$ grows, MER decreases, and when $N = 5000$, the prediction is very close to the actual 0's distribution.

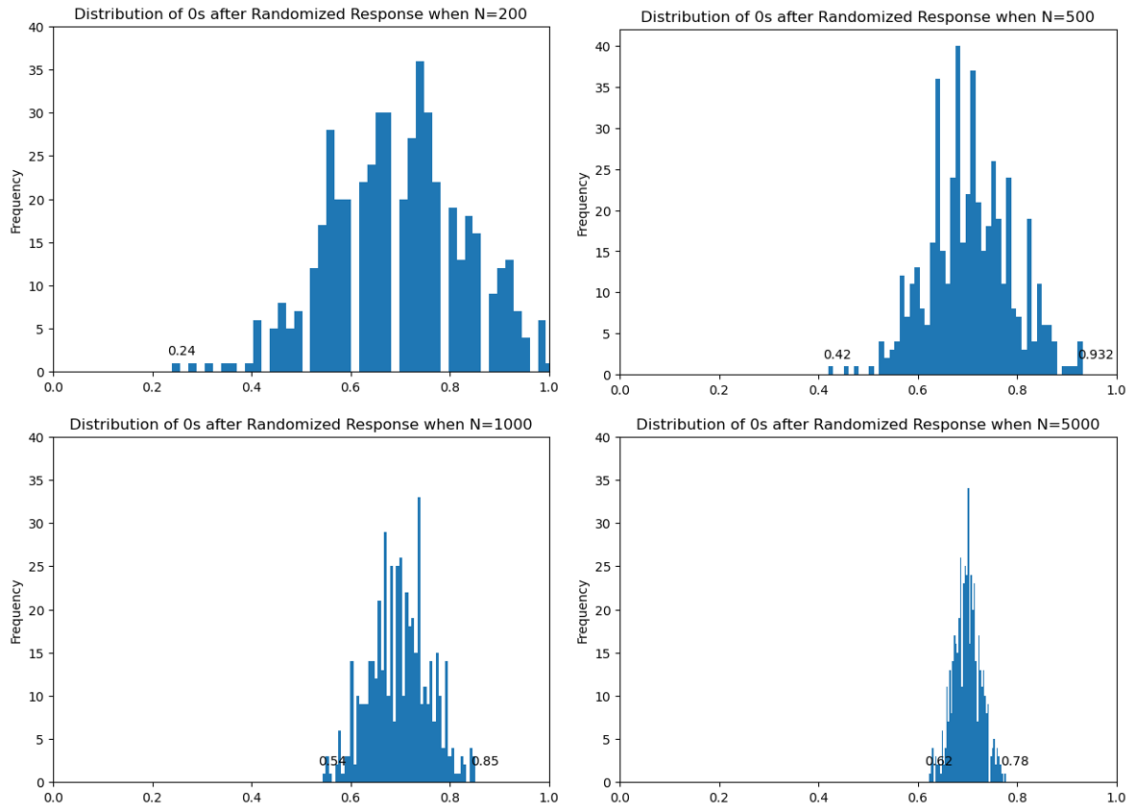


Figure 5.11: Predicted Distributions of 0's in Randomized Response with Different Size of Ratings

Figure 5.12 shows the relation between MER and number of ratings. So, We can conclude that, if the number of ratings is infinitely big, the predicted distribution will be infinitely close to the actual distribution. And taking the extra step or not, it depends on the system requirement, for example, to achieve less than 0.15 MER, there must be at least 1,000 ratings on an item. In real-life applications, it is a common case that there are not sufficient number of users and ratings when the application is just implemented which is called data sparsity problem we mentioned in Section 3.2, and the proposed extra step in TAN+DP can solve this issue, however as a trade-off, the data communication cost will be increased.

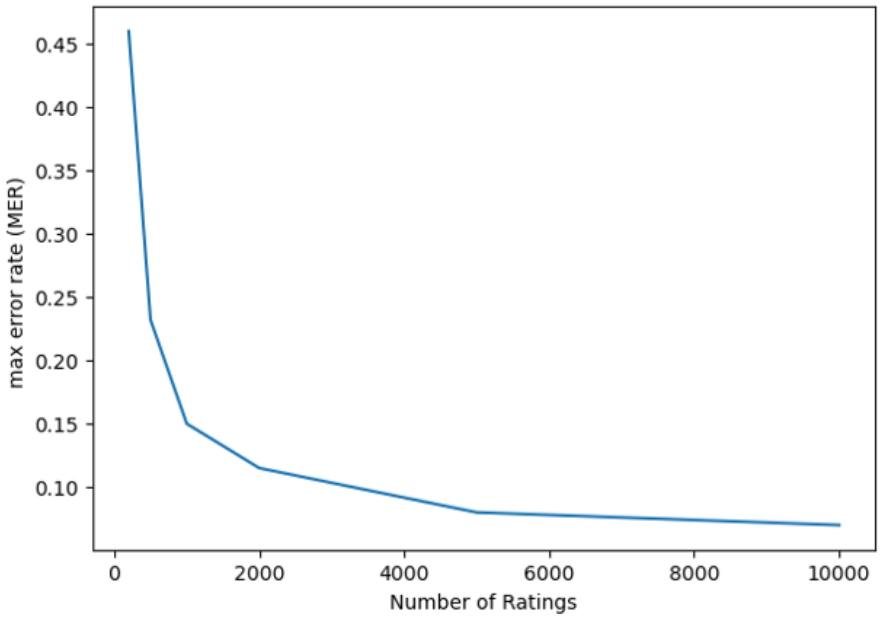


Figure 5.12: MER and Number of Ratings

## 5.7 Summary

In this chapter, we evaluated the proposed privacy-preserving recommendation system using a real-world dataset called LibraryThing. We fully considered all the scenarios and observed the limitation of the dataset: the connection is not necessarily trusted and insufficient number of ratings. For the problem of the untrusted connections, one potential solution could be making dummy trusted connections in the dataset and assigning a reasonable rating to them then retesting it, but we do not think it is worth doing an extra evaluation because of the improved accuracy on LV can definitely benefit the evaluation result. For the problem of insufficient number of ratings, we conducted a separate experiment to find how randomized response is affected by the number of ratings, and our conclusion is that to get a reliable GV, i.e., MER $\leq 0.15$, at least 1,000 ratings on each item are required.

Regarding the questions that we raised at the beginning of the evaluation in Section 5.5. For Q5, the privacy preservation evaluation (Section 5.5.1) showed that the centralized server can view and access all the data from users which considered as privacy leaks. However, in TAN+DP, the AS was unable to infer any individual information from the existing data. For Q6, in Section 5.5.2, TAN+DP outperformed the traditional CF in terms of accuracy, and showed same level of F-measure score as CF. We expect a better result if the LV is made by trusted people or experts. For Q7, in Section 5.5.3, we showed that the time complexity of our solution is linear which definitely outperformed the exponential run time complexity of online CF, and we also take into account the offline CF which the existing recommendation systems generally use. Although it is less efficient than offline CF, the linear time complexity is generally acceptable for most applications.

# Chapter 6

# Conclusions and Future Work

## 6.1 Conclusions

Recommendation systems are ubiquitous in our daily life. Whether we are watching movies or shopping online, we will waste a lot of time in making decisions without recommendation systems. Especially after entering the era of big data, the algorithms of recommendation systems become more and more accurate, and many enterprises have obtained financial benefits by providing better recommendations. However, these data are contributed from our personal information, which is likely to reveal our privacy and cause trouble in our lives.

In this thesis, the goal is to protect users' privacy while preserving accuracy. As a solution, we proposed a personalized privacy-preserving recommendation system which used a novel semi-decentralized network called Trusted-based Agent Network (TAN) and applied differential privacy in data communication. In TAN, the recommendation results mainly rely on a small portion of the network—the local trusted

network which only contains the trusted people or some experts that we know (we defined it as local view). Because the size of these networks is reduced, the recommendation results are limited. As we consider the service provider is suspicious, we avoid the servers interacting with the real data by applying the differential privacy mechanism, i.e., servers cannot directly be involved in the users' real data, and only the obfuscated data are transmitted to the servers (we defined it as global view). Finally, the system generates overall recommendation result based on the local view and global view.

Moreover, our evaluation showed that the privacy-preserving approach can predict the user's preference better than the traditional collaborative filtering (CF) based recommendation system while preserving privacy. With a more suitable and larger dataset, we could be more confident in our result, we expect to see the privacy-preserving recommendation system can outperform the non-privacy-preserving collaborative filtering recommendation system, especially when the ratings from local view are highly related to the user's ratings. Regarding running time complexity evaluation, the run time complexity of our solution is linear which is a bit higher than the offline version of CF but largely lower than the online version of CF.

In Section 1.1, we stated the goal of this thesis and raised two questions, the answers are provided in Chapters 4 and 5, and we elaborate them here:

Q1: How privacy is preserved? In other words, does it prevent the data from attacks like de-identification, does it protect the data from suspicious service providers?

A1: Recall that TAN has the property of both decentralized network and centralized network. The decentralization of TAN ensures the data is distributed in a local

trusted network to prevent the attacks like de-identification from third parties. In terms of the centralization property of TAN, the Agent Server (AS or the service provider) can only collect obfuscated data from the Agents by applying randomized response before data transmission, thus protecting data privacy from suspicious service providers.

Q2: With the privacy-preserving approach, does the recommendation algorithm still provide an accurate result?

A2: In Section 5.5.2, the evaluation results indicated that the proposed privacy-preserving system led to more accurate recommendations than its non-privacy-preserving counterpart.

## 6.2 Future Work

We believe we have fully answered the questions we set at the beginning with the proposed solution, but there are some works could be done with this solution in the future. It would be interesting to evaluate such system by implementing and scaling it to the public, and conduct a survey to the users to measure the accuracy. also, if there is a dataset which contains a large number of ratings and measure of trustworthiness on connections available in the future, it would be fun to evaluate the system with such dataset too.

We have considered the problem regarding insufficient number of ratings in Section 4.5. However, as mentioned in Section 3.2, there are some other challenges when implementing our privacy-preserving system from the business perspective. One of

the major problems could be the cold start problem that is when there are not many users in the system, it would be impossible to use randomized response. This issue could be potentially solved by two approaches: (1) provide a survey when the user registered, and based on the survey we do a content-based filtering instead of collaborative filtering, (2) the user may use our application because some people suggested to her or him, and the user is very likely to make a trusted connection with the referrer, so this is the extreme case that we can set $\alpha$ as 0 in Equation (4.1) and provide recommendation purely based on local view. The other problem could be scalability, as the number of users becomes larger, the computation time to generate a global view becomes longer, this will slow down our recommendation process. We have mentioned that the obfuscated table in the AS could be updated periodically to avoid large data communication cost, however, when the system gains popularity and scales significantly, the limitation of memory usage and increase of communication cost should be considered.

# Bibliography

[ACJB18] Fernando Amat, Ashok Chandrashekar, Tony Jebara, and Justin Basilico. Artwork personalization at Netflix. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 487–488, New York, NY, USA, 2018. Association for Computing Machinery.

[AK20] J Anitha and M Kalaiarasu. Optimized machine learning based collaborative filtering (OMLCF) recommendation system in e-commerce. *Journal of Ambient Intelligence and Humanized Computing*, 12(6):6387–6398, 2020.

[AKK+17] Farman Ali, Daehan Kwak, Pervez Khan, Shaker Hassan A Ei-Sappagh, S. M Riazul Islam, Daeyoung Park, and Kyung-Sup Kwak. Merged ontology and SVM-based information extraction and recommendation system for social robots. *IEEE Access*, 5:12364–12379, 2017.

[ANAB20] Ridho Ananda, Muhammad Zidny Naf'an, Amalia Beladinna Arifa, and Auliya Burhanuddin. Recommendation system for specialization selection using k-means density canopy. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, 4(1):172–179, 2020.

[BFK+15] Arnaud Berlioz, Arik Friedman, Mohamed Ali Kaafar, Roksana Boreli, and Shlomo Berkovsky. Applying differential privacy to matrix factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 107–114. ACM, 2015.

[BH04] Justin Basilico and Thomas Hofmann. Unifying collaborative and content-based filtering. In *Proceedings of the 21st International Conference on Machine Learning*, ICML '04, page 9. ACM, 2004.

[BKZ17] Sophie C Boerman, Sanne Kruikemeier, and Frederik J Zuiderveen Borgesius. Online behavioral advertising: A literature review and research agenda. *Journal of Advertising*, 46(3):363–376, 2017.

[BP15] Sonia Bergamaschi and Laura Po. Comparing lda and lsa topic models for content-based movie recommendation systems. In *Web Information Systems and Technologies*, LNBIP, pages 247–263. Springer International Publishing, Cham, 2015.

[Bur02] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[CHC+18] Rui Chen, Qingyi Hua, Yan-Shuo Chang, Bo Wang, Lei Zhang, and Xiangjie Kong. A survey of collaborative filtering-based recommender systems: From traditional methods to hybrid methods based on social networks. *IEEE Access*, 6:64301–64320, 2018.

[CHM17] Chenwei Cai, Ruining He, and Julian McAuley. SPMC: Socially-aware

personalized markov chains for sparse sequential recommendation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1476–1482, 2017.

[CS00] Paul Cotter and Barry Smyth. PTV: Intelligent personalised TV guides. In *Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence*, page 957–964. AAAI Press, 2000.

[CSA16] Ingrid Christensen, Silvia Schiaffino, and Marcelo Armentano. Social group recommendation in the tourism domain. *Journal of Intelligent Information Systems (JIIS)*, 47(2):209–231, 2016.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284. Springer Berlin Heidelberg, 2006.

[Dwo06] Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming (ICALP 2006)*, LNCS 4052, pages 1–12. Springer Berlin Heidelberg, 2006.

[Dwo08] Cynthia Dwork. Differential privacy: A survey of results. In *Theory and Applications of Models of Computation (TAMC 2008)*, LNCS 4978, pages 1–19. Springer Berlin Heidelberg, 2008.

[FBK16] Arik Friedman, Shlomo Berkovsky, and Mohamed Ali Kâafar. A differ-

ential privacy framework for matrix factorization recommender systems. *User Model. User Adapt. Interact.*, 26(5):425–458, 2016.

[Fki21] Fethi Fkih. Similarity measures for Collaborative Filtering-based Recommender Systems: Review and experimental comparison. *Journal of King Saud University. Computer and information sciences*, 2021.

[FKRA20] Chong Feng, Muzammil Khan, Arif Ur Rahman, and Arshad Ahmad. News recommendation systems - accomplishments, challenges & future directions. *IEEE Access*, 8:16702–16725, 2020.

[GNOT92] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM (CACM)*, 35(12):61–70, December 1992.

[HAM21] Mohsin Hasan Hussein, Akeel Abdulkareem Alsakaa, and Haydar A Marhoon. Adopting explicit and implicit social relations by SVD++ for recommendation system improvement. *Telkomnika*, 19(2):471–478, 2021.

[Hof17] Erik Hofmann. Big data and supply chain decisions: the impact of volume, variety and velocity properties on the bullwhip effect. *International Journal of Production Research*, 55(17):5108–5126, 2017.

[IFO15] F.O Isinkaye, Y.O Folajimi, and B.A Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273, 2015.

[KK17] Hamidreza Koohi and Kourosh Kiani. A new method to find neighbor users that improves the performance of collaborative filtering. *Expert systems with applications*, 83:30–39, 2017.

[KKK+18] Jinsu Kim, Dongyoung Koo, Yuna Kim, Hyunsoo Yoon, Junbum Shin, and Sungwook Kim. Efficient privacy-preserving matrix factorization for recommendation via fully homomorphic encryption. *ACM Trans. Priv. Secur.*, 21(4), June 2018.

[KKZ20] Muhammad Usman Shahid Khan, Samee U Khan, and Albert Y Zomaya. *Big Data-Enabled Internet of Things*. Computing and networks. Institution of Engineering & Technology, Stevenage, 2020.

[KLPC22] Hyeyoung Ko, Suyeon Lee, Yoonseo Park, and Anna Choi. A survey of recommendation systems: Recommendation models, techniques, and application fields. *Electronics*, 11(1):141, 2022.

[KPAM19] Shristi Shakya Khanal, P.W.C Prasad, Abeer Alsadoon, and Angelika Maag. A systematic review: machine learning based recommendation systems for e-learning. *Education and Information Technologies*, 25(4):2635–2664, 2019.

[LFR06] Shyong K. "Tony" Lam, Dan Frankowski, and John Riedl. Do you trust your recommendations? an exploration of security and privacy issues in recommender systems. In *Emerging Trends in Information and Communication Security*, pages 14–29. Springer Berlin Heidelberg, 2006.

[LYC20] Ao Liu, Yanqing Yao, and Xianfu Cheng. Recommender systems with condensed local differential privacy. In *Third International Conference on Machine Learning for Cyber Security - ML4CS 2020, Guangzhou, China, October 8-10, 2020, Part I*, volume 12486 of *LNCS*, pages 355–365. Springer, 2020.

[Mar17] Bernard Marr. *Data Strategy: How to Profit from a World of Big Data, Analytics and the Internet of Things*. Kogan Page, Limited, London, 2017.

[MML+18] XinDi Ma, Jianfeng Ma, Hui Li, Qi Jiang, and Sheng Gao. ARMOR: A trust-based privacy-preserving framework for decentralized friend recommendation in online social networks. *Future Gener. Comput. Syst.*, 79:82–94, 2018.

[MS16] Kirsten Martin and Katie Shilton. Putting mobile application privacy in context: An empirical study of user privacy expectations for mobile devices. *The Information Society*, 32(3):200–216, 2016.

[NS08] A Narayanan and V Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (SP 2008)*, pages 111–125. IEEE, 2008.

[PCH11] Pearl Pu, Li Chen, and Rong Hu. A user-centric evaluation framework for recommender systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 157–164. ACM, 2011.

[PH12] Sung-Hyuk Park and Sang Han. Empirical analysis of the impact of product diversity on long-term performance of recommender systems. In *Proceedings of the 14th Annual International Conference on Electronic Commerce*, ICEC '12, pages 280–281. ACM, 2012.

[PL10] Chenguang Pan and Wenxin Li. Research paper recommendation with topic analysis. In *2010 International Conference on Computer Design and Applications*, pages V4–264. IEEE, 2010.

[PP20] Krupa Patel and Hiren B Patel. A state-of-the-art survey on recommendation system and prospective extensions. *Computers and Electronics in Agriculture*, 178, 2020.

[RK16] Santosh S Rathore and Sandeep Kumar. A decision tree logic based recommendation system to select software fault prediction techniques. *Computing*, 99(3):255–285, 2016.

[RRS15] Francesco Ricci, Lior Rokach, and Bracha Shapira, editors. *Recommender Systems Handbook*. Springer US, New York, NY, 2nd ed. 2015. edition, 2015.

[RV97] Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM (CACM)*, 40(3):56–58, March 1997.

[SKKR01] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01,

pages 285–295, New York, NY, USA, 2001. Association for Computing Machinery.

[SNY17] Rade Stanojevic, Mohamed Nabeel, and Ting Yu. Distributed cardinality estimation of set operations with differential privacy. In *2017 IEEE Symposium on Privacy-Aware Computing (PAC)*, pages 37–48, 2017.

[TAJC20] Phupat Thongyoo, Phuttipong Anantapanya, Pornsuree Jamsri, and Supannada Chotipant. A personalized food recommendation chatbot system for diabetes patients. In *Cooperative Design, Visualization, and Engineering*, LNCS 12341, pages 19–28. Springer International Publishing, Cham, 2020.

[TS22] Kretika Tiwari and Dileep Kumar Singh. Machine learning-based recommendation system for disease-drug material and adverse drug reaction: Comparative review. *Materials Today: Proceedings*, 51:304–313, 2022.

[War65] Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[YLCT19] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[YWW+21] Jingwei Yi, Fangzhao Wu, Chuhan Wu, Ruixuan Liu, Guangzhong Sun, and Xing Xie. Efficient-FedRec: Efficient federated learning framework

for privacy-preserving news recommendation. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, page 2814–2824, 2021.

[ZKG+16] Renjie Zhou, Samamon Khemmarat, Lixin Gao, Jian Wan, Jilin Zhang, Yuyu Yin, and Jun Yu. Boosting video popularity through keyword suggestion and recommendation systems. *Neurocomputing*, 205:529–541, 2016.

[ZMK15] Tong Zhao, Julian McAuley, and Irwin King. Improving latent factor models via personalized feature projection for one class recommendation. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 821–830. ACM, 2015.