Personalized Privacy-Preserving Semi-Centralized Recommendation System in a Social Network

Carson K. Leung(), Qi Wen
Department of Computer Science, *University of Manitoba*
Winnipeg, MB, Canada
Carson.Leung@UManitoba.ca

***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 numbers of choices. Personalized recommendation has gained its popularity as it provides recommendations according to the user profile, preferences and/or interests. Many existing systems make recommendation by centralizing data. However, the exposure of sensitive data raises a privacy concern as research has shown that it is possible to de-identify anonymous users. Examples include inferring sensitive information (e.g., political views, sexual orientations) from non-sensitive data (e.g., movie ratings). In this paper, we present a personalized privacy-preserving recommendation system called Trust-based Agent Network (TAN). It tackles the privacy issue by semi-decentralizing data and treating each node in the network as an agent. As such, data are distributed to each agent within each trusted network, and the recommendation service provider collects only obfuscated data from agents by adopting the differential-privacy mechanism. Consequently, data in our TAN are either protected inside local trusted networks or obfuscated outside of trusted networks. Final recommendation can then be made by aggregating the local suggestions from the trusted network and obfuscated global suggestions from the service provider. Personalized recommendations can be made by putting more emphasize on local suggestions. Evaluation results show that our TAN leads to high accuracy and highly personalized recommendations while protecting privacy.**

***Index Terms*—recommendation system, personalization, collaborative filtering, semi-centralized, trusted network**

I. Introduction

Due to the rapid development of the Internet of Things (IoT), a tremendous amount of data is generated and collected from various applications. This leads to big data [16], [22]. Nowadays, people have access to a massive amount of online information through various devices and services. However, the number of choices can 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 [37], food [28], travel [8], news [13], movies [1], etc. Especially in an e-commerce setting, a good recommendation system will not only enhance user experiences but also increase revenue [26].

Many works have been done to improve the performance of recommenders to get more accurate recommendations. 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, income) and feedback on products. For instance, in *collaborative filtering (CF)* based recommendation systems (which aim to find similar users and recommend items based on the similar users' ratings), the service provider collects as much user information as possible in order to achieve better recommendation. However, the service provider could be vulnerable and malicious, the servers can be attacked by third parties. This may also lead to the exposure of personal information, which raises serious privacy concerns [19]. For instance, in 2018, Facebook reported that there were 500 million users' data leak[1]. Moreover, the service provider could be also suspicious (e.g., it can sell user data to third parties for financial benefit).

One may argue that, as the collected data are usually insensitive information (e.g., users' feedback on a certain product or video watching histories), it is not necessary to protect this insensitive information. However, the information that is assumed to be insensitive can also vulnerable to privacy leaks. Narayanan and Shmatikov [25] demonstrated a practical de-identification attack called *linkage attack* and successfully uncovered sensitive user information from the anonymous movie ratings of subscribers. A linkage attack combines (a) datasets that are available in the public with (b) the anonymized dataset to re-identify personal information. Two common solutions are *data protection* and *distributed recommendation*. The former is to protect data by data encryption or data obfuscation, whereas the latter is to distribute the data to clients. As a preview, our solution is a non-trivial combination of these two solutions, and we also take trusted people's recommendation into account. More specifically, we design a semi-centralized network called Trusted-based Agent Network (TAN) and incorporate differential-privacy mechanism to obfuscate data.

Our *key contributions* of this paper include the design of a novel designed network architecture called trust-based Agent Network (TAN), and the adaption of differential privacy (DP) into TAN. Evaluation results show that such an incorporation of DP into the TAN protects privacy while preserving accurate recommendation.

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

The remainder of this paper is organized as follows. We begin with necessary background and related work in Section II, present our solution to the privacy issue in Section III, evaluate our proposed solution in Section IV, and finally draw conclusions in Section V.

II. Background and Related Works

In recent decades, an exponential increase in the volumes of data (i.e., Big data) has led to a dramatic increase in the number of applications and research articles regarding recommendation systems. In addition to information filtering systems, there are also machine learning-based recommendation systems [2], [17], [29], which are almost impossible to implement without the concept of big data. Especially among the popular 3 V's (Volume, Velocity and Variety) of the big data [15], the volume of data plays a vital role when applying differential privacy. In fact, the techniques (e.g., collaborative filtering, differential privacy) that are used in our privacy-preserving recommendation system are based on the assumption that there are voluminous data available in the application. This assumption holds in the current big data era.

Recall from Section I that a common solution to linkage attack is data protection. Two common data protection solutions include cryptography and obfuscation. *Cryptography solutions* are based on mathematical theory to study data encryption algorithms. For instance, Kim et al. [18] used fully homomorphic encryption to encrypt the user rating data and returned an encrypted output. By doing so, the adversaries or the service provider cannot learn from the user rating values. Shmueli and Tassa [27] used a mediator to receive the encrypted rating data and performed CF computation on them. Their work was extended by Hoorin and Tassa [3] to multiple distributed mediators. In other words, the latter two works relied on a mediator, whereas we focus on establishing a trusted local network for CF in our TAN.

In contrast, *obfuscation solutions* often impose disturbance on the user information to prevent individual users from being identified. Among related works, many [4], [14], [20] relied on *differential privacy (DP)* [11]. Its key idea is that, any individual data in the large database are small enough, and modifying (e.g., adding random noises to) any of them will not affect the overall dataset statistical information. As such, probabilities of a query returning the same result from the real dataset and from obfuscated dataset is bounded within ϵ [10]. *Randomized response* algorithm [32] satisfies ϵ-differential privacy. The DP has been applied to different problems. For example, Wang et al. [30] applied global and local DP to collaborative bandits in solving the cold-start problem. In contrast, we establish global and local DP for a more general problem of CF in this paper.

Recall from Section I that another common solution to linkage attack is distributed recommendation. For example, Ma et al. [21] proposed a decentralized framework called ARMOR for friend recommendation from online social networks. Some other related works [24], [34] applied *federated learning* [33], which is a machine learning technique. It protects privacy by allowing multiple clients (i.e., federation) to enter the training process of a machine learning model without directly sharing data. However, in such a decentralized setting, the data communication cost can be very expensive. This explains why implementing federated learning usually costs higher than processing data centrally. Observing this drawback of distributed recommendation, we design a novel semi-centralized network called Trusted-based Agent Network (TAN) in this paper. It leverages both advantage of centralized and decentralized network, i.e., reduces data communication cost while protecting the data.

III. Our Personalized Privacy-Preserving Recommendation System

A. Motivation and System Overview

A key problem in the traditional centralized CF recommendation systems is that, as data are stored centrally in the server, the service provider can easily manipulate the data and use these data for financial benefit (i.e., privacy leaks). As a solution, a distributed recommendation can eliminate the server from the network, and its corresponding user similarity matrix can be distributed to all the clients using a peer-to-peer (P2P) network. Because any node can be malicious in the P2P network, a trusted network should be considered.

To deal with this problem, we design a semi-centralized network—called *Trust-based Agent Network (TAN)*—to build trusted and untrusted links. The resulting TAN consists of a number of small local trusted networks (each of which is a P2P network) and a server. Data are distributed via trusted links in each local trusted network.

Another key problem in the traditional CF recommendation systems is that it is not necessary to find similar users throughout the entire network. Each client could have a local trusted network that is composed of trusted people (e.g., family members, experts), and similar users could be easily found within the local trusted network. Because their opinions (from people in the local trusted network) are often more important than opinions of the public in the entire network, one should take into account—and put more emphasis on—the recommendation from these trusted people (in the local network). To avoid bias on searching from this reduced local network, one should also request general recommendations (e.g., how many people like this item?) from the entire network.

To deal with this problem, we design our TAN in such a way that it follows the privacy-preserving recommendation process below:

1) Each client has a local trusted network (i.e., a subset of the whole network), and a user similarity matrix is distributed inside the trusted network. Each client computes recommendation result based on the user similarity matrix and obtains a *trusted recommendation*.
2) Clients also send a request to the server to get a *general recommendation* throughout the whole network. Server collects obfuscated data from all the clients, and sends a general recommendation to all these clients.

3) Each client then computes the *final recommendation* based on both the local trusted recommendation and general recommendation.

Example 1: Consider Raymond, who owns a car C_1 and is considering to buy another car C_2. He first asks people he trusts (e.g., his family members and car experts in his local trusted network), they tell him C_2 is a good car. He finds that one person (say, Jane) in his trusted network has a similar interest with him, and he listens to Jane's recommendation. In addition, he also wants to know opinions of other people (who are not in his local network think of C_2), i.e., a general recommendation on how many people who owns C_1 also like C_2. Then, the service provider can provide Raymond with a general recommendation that is computed by using the obfuscated data received from all the people. He can then make a *final decision* based on the *trusted expert's recommendation* and the *general recommendation collected by the service provider*.

To elaborate, we define the trusted recommendation (provided by peers in the local trusted network by using user-based CF technique) as *local view (LV)*. As the search space is reduced from the entire network to local trusted network, there may be some concerns that the resulting trusted recommendation does not provide sufficient information. To alleviate this concern, we define the data collected by the server from the entire network as *global view (GV)*. The GV can be used to give a general broader recommendation. The final recommendation result can be then calculated by:

$$P_{u,i} = \alpha \times GV + (1 - \alpha) \times LV \tag{1}$$

where $P_{u,i}$ is user u's final recommendation (e.g., predicted rating) for item i. And, α is the weight assigned to GV such that $0 \leq \alpha \leq 1$. In the extreme case, when $\alpha = 0$, the final recommendation is purely based on the trusted recommendations; when $\alpha = 1$, the final recommendation is purely made by global recommendations. To put emphasize on local trusted network, the range for α can be adjusted to become $0 \leq \alpha < 0.5$

Furthermore, we adapt differential privacy into our TAN to form TAN+DP, which obfuscates data to protect user privacy. By doing so, data can be distributed via trusted links in each local trusted network, but obfuscated via untrusted links. As such, the obfuscated data can be used to generate a general recommendation (i.e., GV) from the entire network.

B. Trust-Based Agent Network (TAN)

In our Trust-based Agent Network (TAN), an agent (i.e., client) can be connected to another agent via a *trusted connection* to form a local trusted agent network. The agent can also be connected to an agent server (AS) via an *untrusted connection*. To elaborate:

- *Trusted connections* are links in local trusted networks, within which data are distributed via trusted connections to each agent. Consequently, attackers are prevented from collecting the real data. The trusted connections can be established by sharing a secret code or scanning a Quick Response (QR) code in person so that AS has no knowledge of the trusted connections.
- *Untrusted connections* are links between agents and AS. Data are *obfuscated* before transmitting through these untrusted connections. Consequently, the service provider or other attackers are prevented from inferring sensitive information (from the real data).

Note that, on the one hand, in a traditional *centralized network*, all data are stored on a single node (server) and it is easy to collect data throughout the network. As such, the data can be easily manipulated by the people who own the server (e.g., recommendation service providers). On the other hand, in a *decentralized network*, data are stored locally, and users (rather than server owners) can have full control of their data thus ensuring data privacy. However, data communication is very expensive and inefficient. In contrast, our TAN is a *semi-centralized network*, which is designed to take the advantages/benefits of both networks:

- All the agents are connected to the AS by untrusted connections in TAN. After collecting data from agents, the AS sends back a GV to all the agents.
- Data are distributed in each local trusted network in TAN, and all the agents can share data privately inside the local trusted network.

Example 2: Let us use Fig. 1(a) to illustrate how to generate TAN and how Agent3 gets the final recommendation from TAN:

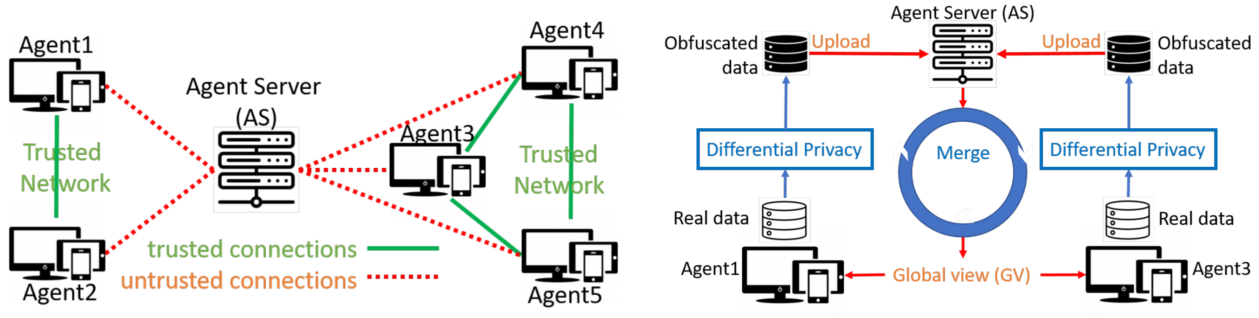
1) After creating an agent server (AS), Agent3 enters TAN. It establishes an untrusted connection (as indicated by a red dashed line) to the AS, and two trusted connections (as indicated by two green solid lines) to Agents4 and 5.
2) As data are distributed inside each trusted network, Agent3 computes the LV from its trusted network (i.e., trusted recommendation from trusted peers Agents4 and 5) based on a user similarity matrix.
3) Agent3 then requests for a general recommendation from AS, which collects data from all the agents and sends the GV to Agent3.
4) Agent3 makes final recommendation based on the LV (received from its local trusted network) and GV (received from the AS).

C. TAN+DP: TAN with Differential Privacy

In TAN, data are distributed via trusted links in each local trusted network, but via untrusted links in the global network. To protect privacy (on data distributed via untrusted links), we adapt differential privacy (DP) into our TAN to form *TAN+DP*. By doing so, data are obfuscated—using randomized response—in the untrusted connections in the resulting TAN+DP.

Example 3: Continue with Example 2. With TAN+DP, Step (3) becomes:

(3) Agent3 then requests for a general recommendation from the AS, which collects *obfuscated* data from all



(a) TAN (b) DP in TAN+DP

Fig. 1. (a) Trust-Based Agent Network (TAN), (b) Use of differential privacy (DP) for uploading obfuscated data in TAN+DP

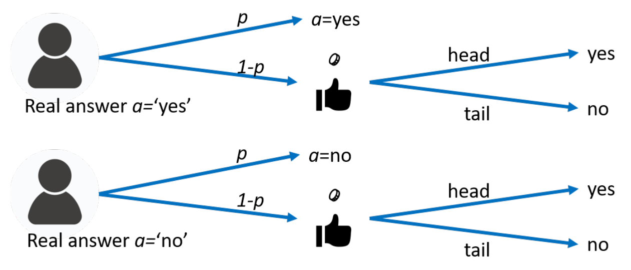
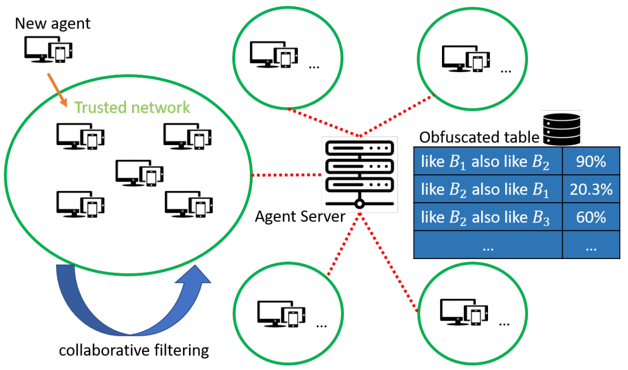


Fig. 2. Randomized Response



like B_1 also like B_2	90%
like B_2 also like B_1	20.3%
like B_2 also like B_3	60%
...	...

Fig. 3. Privacy-Preserving Recommendation System

the agents. Before uploading these obfuscated data, a DP mechanism is adopted to the real data (e.g., ratings on cars) from all agents in the global network. Then, the AS sends the GV to Agent3. See Fig. 1(b).

Recall from Section II that randomized response satisfies ϵ-differential privacy. Hence, we incorporate the randomized response to obfuscate the real data. The key idea is that the $Randomization$ function takes the real answer a and a user-defined probability p as inputs, and returns an obfuscated answer. Specifically, it first compares a randomly-generated probability r with p. If $r \leq p$, then it returns the real answer a. Otherwise (when $r > p$), it flips a fair coin, and returns "yes" if head and "no" if tail. See Fig. 2.

Then, a logical question is: Can adversaries infer from the randomized response to de-identify individuals? In other words, can adversaries infer which response belongs to which individual? To answer these questions, let us consider the following. If an obfuscated answer is "yes", there is no guarantee that the real answer is also "yes". It is because there is a case that, although a user's real answer was "no", the randomly-generated probability $r > p$ and the tossed coin showed a head. Hence, because of the randomization, adversaries *cannot* infer from the randomized response to de-identify individuals.

A follow-up question is: While the obfuscated data preserve privacy, can we obtain a GV from the obfuscated data? The answer is yes. Let us illustrate the idea in Example 4.

Example 4: Let set the user-defined probability p = 25%, i.e., there is a 25% probability that the real data are not obfuscated. Suppose there are 100,000 responses with 45,000 "yes" and 55,000 "no". Then, 75,000 responses are expected to use fair-coin tossing, and let us call them *fake responses*:

- Among these fake responses, approximately 50% (when using a fair coin) of these 75,000 = 37,500 are fake "yes" and 37,500 are fake "no". Then, we can infer that there are approximately 37,500 fake responses in 45,000 of "yes" responses. 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, 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". This helps to obtain a GV from the obfuscated data.

Note that, even though we find out there are 7,500 real "yes" answers (to help obtain the GV), the user privacy is protected as one cannot infer which 7,500 individuals provided a real "yes".

Example 5: Revisit Example 3. With TAN+DP, let us elaborate its Step (3). Agent3 requests for a general recommendation (e.g., How many people in the global network who owns car C_1 also like C_2?) sends from the AS. The AS sends this request to all the agents.

- If an agent (e.g., Agent1) does not have a record satisfying the request (e.g., low or no rating on C_2), it does not respond.
- Otherwise, the agent (e.g., Agent2) has an answer "yes" or "no" to the request based on its rating (e.g., Agent2's rating on C_2). It invokes $Randomization$ function to obfuscate its real answer and sends the obfuscated data to the AS.

The AS then merges obfuscated data from all the agents, and sends the GV to all the agents (including the requester Agent3). Agent3 receives GV from the AS. As shown in Fig. 3, there is an *obfuscated table* created to store GV and it updates periodically.

Let us also elaborate Step (4). Finally, Agent3 makes final recommendation by using Eq. (1). Suppose $\alpha = 0.3$, the rating range in the system set to 0 to 100, the trusted recommendation LV (generated in local trusted network using user-based CF) is 100, and the GV from obfuscated table is 90. Then, the predicted rating p_{Agent3,C_2} for Agent3 on car C_2 would be $\alpha \times GV + (1-\alpha) \times LV = 0.3 \times 90 + 0.7 \times 100 = 97$.

D. An Additional Step for TAN+DP when the Number of Ratings is Insufficient

Now, we consider an edge case for TAN+DP. Recall from Section II, differential privacy only works with big data. If there is insufficient number of ratings for the GV, then the distribution (e.g., percentage of likes and dislikes) in obfuscated data may not be accurate. To deal with this edge case, we add an additional step to TAN+DP.

Recall that, when the AS sends a request from a general recommendation to all the agents, 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 agent reply to the AS. In fact, it is not uncommon that there is insufficient number of ratings on many items in real-life applications—except for very popular items. Our solution is that, when the number of ratings in the GV is insufficient, the AS asks multiple responses from agents who have rated the item. To elaborate, the process is as follow:

1) The AS sends the requests (e.g., Do you like car C_2 when you own C_1?) to all the agents.
2) When the AS receives only a few answers (say, n answers), it sends the same requests again and asks those n responded agents to reply the same request multiple times (say, m times). Note that, although the request is the same, the answers may be different because of the 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 $n = 1,00$ and 70 people in the global network likes C_2 (i.e., GV equals 70). With $m = 10$, the number of people likes C_2 becomes 700 and GV still equals 70. This solution comes with a price of extra data communication cost.

TABLE I
A TUPLE IN LIBRARYTHING

userId	item ID	stars	social relation	comment
van_stef	3206242	5.0	Mishalak	This a great book for young readers ...

TABLE II
BASIC STATISTICS OF LIBRARYTHING

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

IV. EVALUATION

A. Data Acquisition

To evaluate our TAN and TAN+DP systems, we used an existing dataset [6], [36], which were generated from LibraryThing[2] that includes users' ratings on books and social connections between users. Table I shows an example row in the dataset. During the data pre-processing step, some redundant and irrelevant columns are trimmed. The resulting dataset is available at [URL hidden for anonymous review]. Table II shows basic statistics: There are around 83,000 users who rated on 500,000 items. The total number of ratings is 1.7 million, and the total number of social connections is 220,000.

B. Data Exploration

Figure 4 shows the distribution of ratings given by users (in the center of the figure). The histogram on the top indicates the distribution of ratings, and the histogram on the right indicates the distribution of the number of ratings on each item. More than 80% of ratings are in the range of [2.5, 4.0] and the average of the ratings is 3.11. The largest rated item has 2,439 ratings. As the average number of ratings on each item is 20.5, which is not big enough to support the randomized response. Hence, we take the additional step (i.e., multiple responses from the agents) in TAN+DP (as described in Section III-D) in the evaluation.

C. Evaluation Metrics

Root mean square error (RMSE) [9] is a commonly used statistical accuracy metric and it is computed as follow:

$$RMSE = \sqrt{\frac{1}{n}\sum(P_{u,i} - A_{u,i})^2} \tag{2}$$

where n is the total number of ratings, $P_{u,i}$ is user u's predicted rating on item i and $A_{u,i}$ 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, a low RMSE indicates a more accurate recommendation. We use RMSE to define the weights on LV and GV is in Section IV-D.

We also use *confusion matrix* to calculate accuracy of the prediction. The confusion matrix includes True Positive (TP), False Positive (FP), False Negative (FN), True Negative (TN):

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

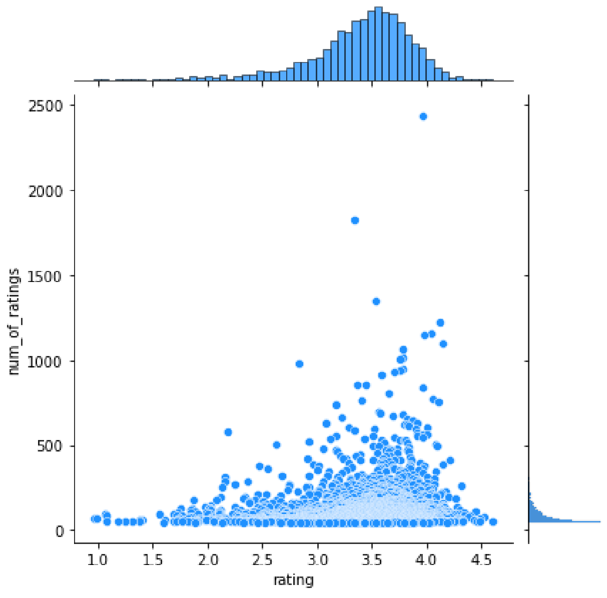


Fig. 4. Distribution of Ratings in the LibraryThing Dataset

- True Positive (TP) means the model correctly predicted positive values, i.e., correctly predicted that the user u likes the item i.
- False Positive (FP) means the model incorrectly predicted positive values, i.e., fail to predict that the user u likes the item i.
- False Negative (FN) means the model incorrectly predicted negative values, i.e., fail to predict that the user u dislikes the item i.
- True Negative (TN) means the model correctly predicted negative values, i.e., correctly predicted that the user u dislikes the item i.

With the average rating in the dataset is 3.11, we interpret the positive class as "user i likes item i if its rating $\geq$ 3.11" and the negative class as "user i dislikes item i if its rating $<$ 3.11". From the confusion matrix, we compute accuracy as the total number of correct predictions divided by the total number of all predictions:

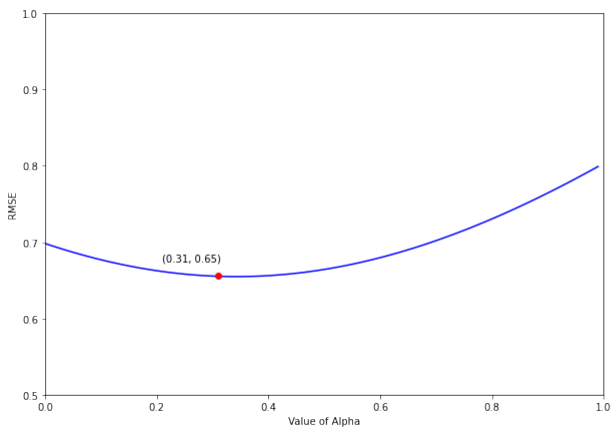
$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3}$$

D. Determine the Weights for Global View (GV) and Local View (LV)

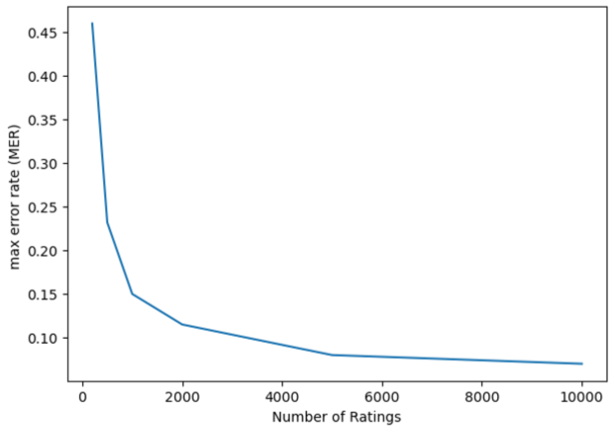
Recall from Section III-A that the final recommendation is a weighted sum of GV and LV (as shown in Eq. (1)). Figure 5(a) show evaluation results that $\alpha = 0.31$ (i.e., putting 31% weights on GV and 69% on LV) led to the smallest RMSE.

E. Comparisons with Related Works

We used the centralized user-based CF [] as our baseline, and compared it with our system.



(a) α vs. RMSE



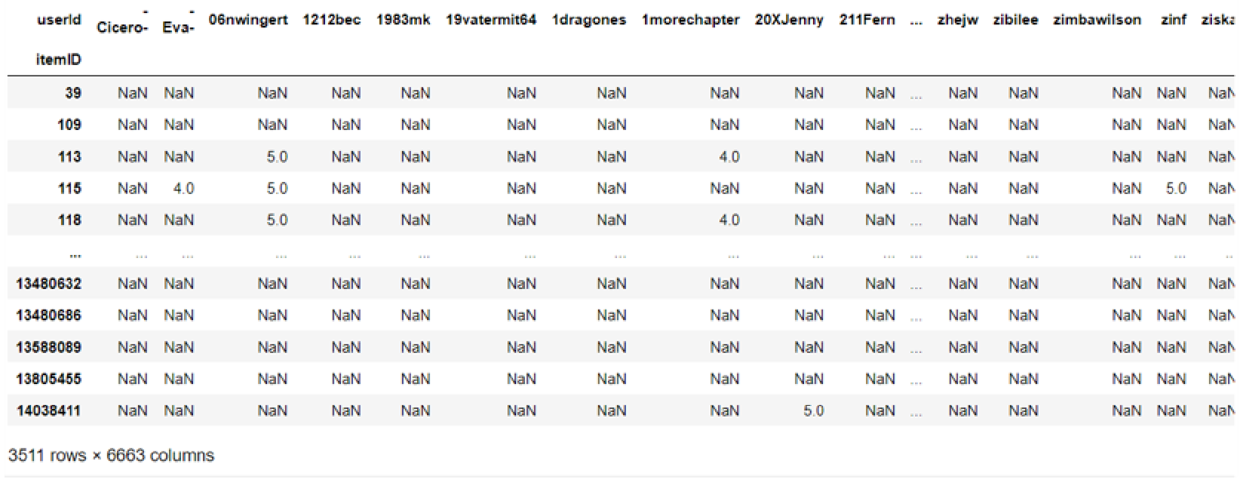
(b) #ratings vs. MER

Fig. 5. (a) Optimal α; (b) #ratings vs. MER

1) Privacy Preservation: As service providers can be suspicious, we represent how data are stored in traditional centralized server when compared with those stored in Agent Server (AS) in our TAN. As shown in Fig. 6, the server has full access to all the user information in traditional approach. This may lead to linkage attack, in which attackers from third parties could easily infer sensitive information of these users by combining user ratings with other existing datasets that are available in the public).

In contrast, in TAN (Fig. 7), 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 (e.g., "1212bec") does not reply as they do not have a record on item 113 or 115. As a result, neither the service provider (the AS) nor third party attackers can infer individual information from existing data received and stored in the AS.

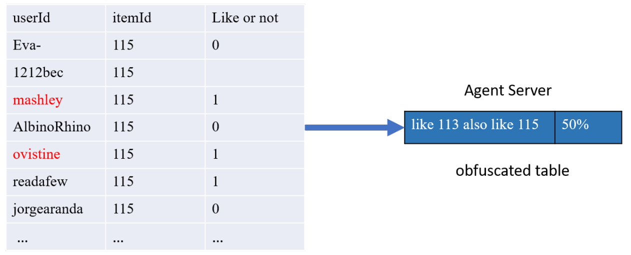
Table III shows service providers' accessibility on data among different servers. The traditional server stores user-item rating matrix locally, and thus the service provider has ability to access users' entire information, i.e., users' privacy is not



userId / 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

Fig. 6. Data Representation in the Traditional Server



userId	itemId	Like or not
Eva-	115	0
1212bec	115	
mashley	115	1
AlbinoRhino	115	0
ovistine	115	1
readafew	115	1
jorgearanda	115	0
...	...	...

Fig. 7. Data Representation in Agent Server in our TAN

TABLE III
SERVICE PROVIDER ACCESSIBILITY ON DATA

	Traditional server	TAN AS AS	TAN+DP AS
Prevention from accessing the user-item rating matrix & from tracking user ratings	×	√	√
Prevention from tracking user IDs & user preferences	×	×	√

protected at all. In contrast, in our 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. However, AS still can track user IDs and preferences. By adding differential privacy (DP) in data communication to TAN (to form TAN+DP), data privacy is fully preserved in our TAN+DP because all the data AS have received are obfuscated and the AS cannot access or infer any individual information from the data.

2) *Accuracy:* In order to predict a user's rating and compare with the actual user's rating, we filter users by following steps:

1) Select the users who have at least one connection (25,610 users)
2) Among those users, select the users who have at least one connection and have rated at least one common item. (7,809 users)

Then, we tested with *20-fold cross-validation*. We separate our privacy-preserving solutions to two cases: without DP (i.e., TAN) and with DP (i.e., TAN+DP). In other words, we first observe if TAN can provide a good prediction accuracy, then we add DP to TAN and observe again. To elaborate, the following cases are tested:

- Privacy is not preserved in the traditional centralized CF recommendation system (e.g., pure CF to predict users? ratings). The observed average prediction accuracy is 83.32%.
- Final recommendation in TAN relies on a combination of local trusted network recommendations (LV) and global recommendations (GV). However, data are not obfuscated in the global network. The observed average prediction accuracy is 84.93%.
- Privacy is preserved in TAN+DP, in which data are not obfuscated in the global network by randomized response. The observed average prediction accuracy is 84.95%.

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 can be expensive. Hence, 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 thus much less than the traditional one. However, the additional cost of getting GV should be considered. So, we compare run times in following two cases:

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

The overall run time comparison of online CF, offline CF and TAN+DP is shown in Fig. 8(a). Here, if the user-item

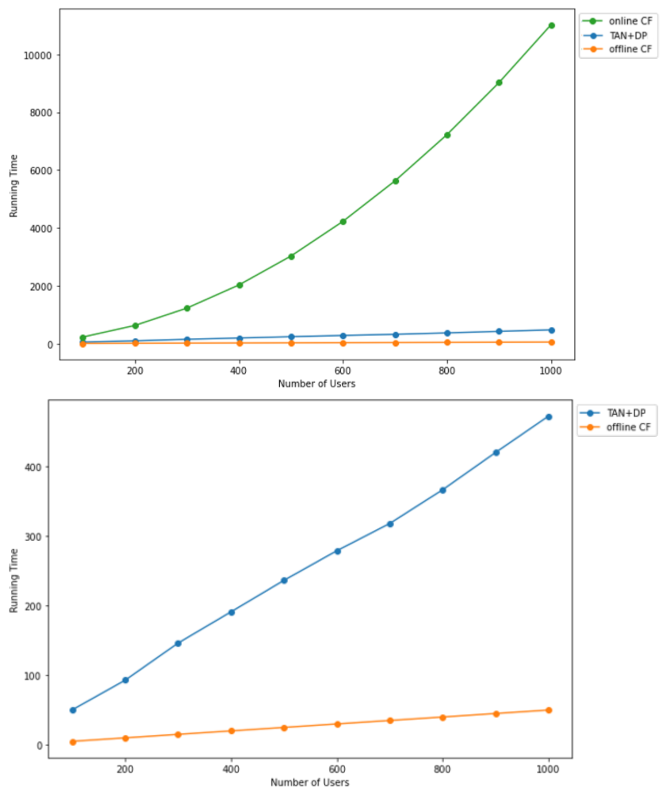


Fig. 8. (a) Runtime comparisons on offline CF, Online CF and TAN+DP, and (b) its zoomed-in view

rating matrix is computed every time when calculating the users' similarities (i.e., online CF), the run time complexity is 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 O(n) which is proportional to the number of users. In our privacy-preserving network (i.e., TAN+DP), the run time complexity is O(tn) where t is the number of trusted people in the LV which is usually a small constant number. In the worst case, the run time complexity of TAN+DP can be the same as online CF (i.e., exponential 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). Hence, the run time complexity of TAN+DP become linear, i.e., O(tn) $\approx$ O(n) when for constant t. Figure 8(a) shows comparisons, and Fig. 8(b) shows a zoomed-in view.

F. Determine When to Take the Additional Step in TAN+DP?

Recall that we took an additional step in the above evaluation because of the insufficient number of ratings on each item. This extra step requires an addition request sent by the AS to all the agents, which increase the data communication cost. Hence, we want to avoid it as much as possible. In order to find the relation between number of ratings and prediction accuracy, we created an integer array of a size of N and assigned 70% of 0's and 30% of 1's. We simulated this array as the users' real responses, i.e., 0's real distribution is 0.7. Then, we applied randomized response and backtracking, and run the test 500 times for different number of ratings. Figure 9 shows predicted 0's distribution frequencies for N number of ratings. We consider *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(b) shows the relation between MER and number of ratings. So, we observe 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 not uncommon that there are insufficient number of users and ratings when the application is just implemented, which is called *data sparsity problem*. The proposed extra step in TAN+DP can solve this issue. As a trade-off, the data communication cost will be increased.

To conclude our evaluation, we use Table IV to summarize an overall comparison for all the cases. In traditional CF, the network is centralized thus service provider can easily maintain data. However, user privacy is not preserved, and the computation time for generating a user similarity matrix globally is also costly. In our proposed solution, user's privacy is persevered while the prediction accuracy is also guaranteed. As a trade-off, our approach requires trusted connections in the network and an additional data communication cost should be also considered.

V. Conclusion

Recommendation systems are ubiquitous in our daily life. Whether we are watching movies or shopping online, we spend a lot of time in making decisions without recommendation systems. Especially after entering the era of big data, recommendation systems become 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.

In paper, our goal is to protect users' privacy while preserving accuracy. As a solution, we proposed a personalized privacy-preserving recommendation system, which provides a novel semi-centralized network called Trusted-based Agent Network (TAN) and applies 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 (who gives local view, aka trusted recommendation). As the service provider can be suspicious, we avoid the servers interacting with the real data by applying the differential

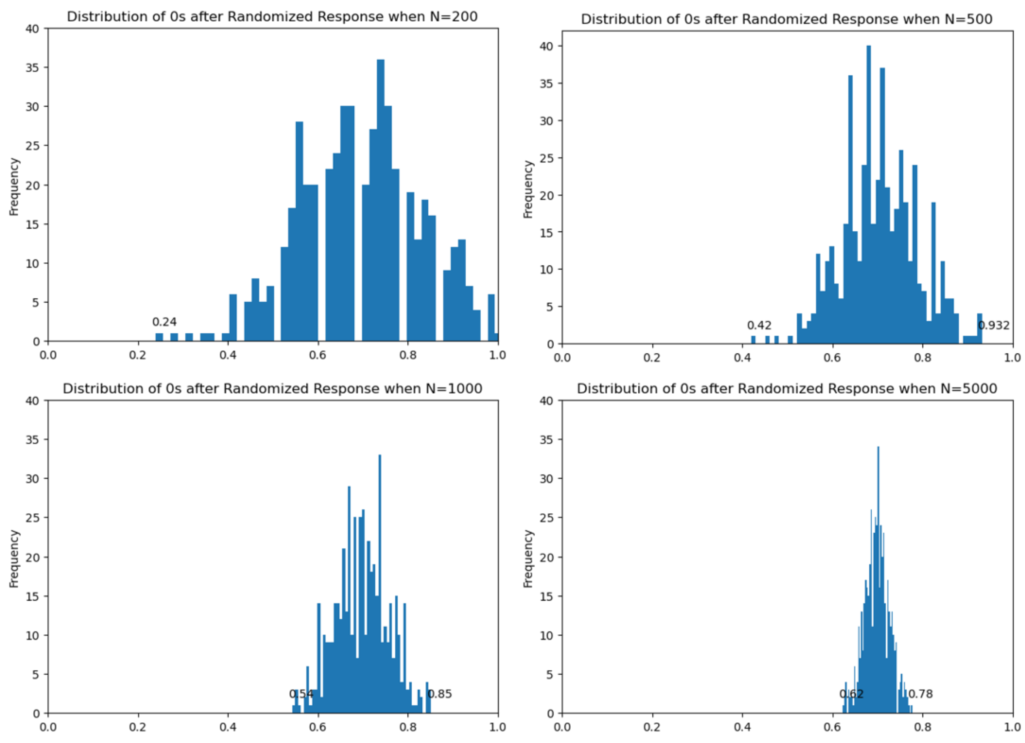


Fig. 9. Predicted Distributions of 0's in Randomized Response with Different Size of Ratings

TABLE IV
OVERALL COMPARISON

	traditional CF	TAN	TAN+DP with the extra step
privacy preserved	×	×	√
trusted recommendation	not considered	need trusted comm. info	
network	centralized	semi-centralized	
	data are easy to maintain, easy to send & receive	add'l data comm. cost for GV	2× add'l data comm. cost for GV
compute big data	slow	fast	fast
prediction accuracy	83.32%	84.93%	84.95%

privacy mechanism (i.e., servers cannot directly be involved in the users' real data) and only the obfuscated data are transmitted to the servers (which leads to global view, aka general recommendation). Finally, the system makes final recommendation result based on a weighted sum of local and global views.

Moreover, we showed that TAN+DP preserves privacy because of both decentralized and centralized properties of TAN. 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) can only collect obfuscated data from the Agents by applying randomized response before data transmission, thus protecting data privacy from suspicious service providers. Our evaluation also showed that our privacy-preserving approach can predict the user's preference better than the traditional CF based recommendation system. With a more suitable and larger dataset, we could be more confident in our result, and we expect to see the privacy-preserving recommendation system can outperform the non-privacy-preserving CF 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. As *ongoing and future work*, we explore ways to further enhance accuracy of our prediction.

REFERENCES

[1] F. Amat, A. Chandrashekar, T. Jebara, and J. Basilico. 2018. Artwork personalization at Netflix. In ACM RecSys 2018, pp. 487-488.
[2] J. Anitha and M. Kalaiarasu. 2020. Optimized machine learning based collaborative filtering (OMLCF) recommendation system in e-commerce. JAIHC 12(6), 6387?6398.
[3] A.B. Horin and T. Tamir. 2021. Privacy preserving collaborative filtering by distributed mediation. In ACM RecSys 2021, pp. 332-341.
[4] A. Berlioz, A. Friedman, M. A. Kaafar, R. Boreli, and S. Berkovsky. 2015. Applying differential privacy to matrix factorization. In ACM RecSys 2015, pp. 107?114.

[5] S.C. Boerman, S. Kruikemeier, and F.J. Zuiderveen Borgesius. 2017. Online behavioral advertising: a literature review and research agenda. Journal of Advertising 46(3), 363?376

[6] C. Cai, R. He, and J. McAuley. 2017. SPMC: socially-aware personalized Markov chains for sparse sequential recommendation. In IJCAI 2017, pp. 1476-1482.

[7] Q. Chen, X. Yu, N. Liu, X. Yuan, Z. Wang. 2020. Personalized course recommendation based on eye-tracking technology and deep learning. In IEEE DSAA 2020, pp. 692-698.

[8] I. Christensen, S. Schiaffino, and M. Armentano. 2016. Social group recommendation in the tourism domain. JIIS 47(2), 209?231.

[9] P. Cotter and B. Smyth. 2000. PTV: intelligent personalised TV guides. In AAAI/IAAI 2000, pp. 957-964.

[10] C. Dwork. 2006. Differential privacy. In ICALP 2006, pp. 1?12.

[11] C. Dwork, F. McSherry, K. Nissim, and A. Smith. 2006. Calibrating noise to sensitivity in private data analysis. In Theory of Cryptography, pp. 265?284.

[12] W. Fan, Y. Ma, D. Yin, J. Wang, J. Tang, and Q. Li. 2019. Deep social collaborative filtering. In: ACM RecSys 2019, pp. 305-313.

[13] C. Feng, M. Khan, A.U. Rahman, and A. Ahmad. 2020. News recommendation systems - accomplishments, challenges & future directions. IEEE Access 8, 16702?16725.

[14] A. Friedman, S. Berkovsky, and Mohamed Ali Kâafar. 2016. A differential privacy framework for matrix factorization recommender systems. User Model. User Adapt. Interact 26(5), 425?458.

[15] E. Hofmann. 2017. Big data and supply chain decisions: the impact of volume, variety and velocity properties on the bullwhip effect. IJPR 55(17), 5108?5126.

[16] M.U.S. Khan, S.U. Khan, and A.Y. Zomaya. 2020. Big Data-Enabled Internet of Things. IET.

[17] S.S. Khanal, P.W.C Prasad, A. Alsadoon, and A. Maag. 2019. A systematic review: machine learning based recommendation systems for e-learning. Education and Information Technologies 25(4), 2635?2664.

[18] J. Kim, D. Koo, Y. Kim, H. Yoon, J. Shin, and S. Kim. 2018. Efficient privacy-preserving matrix factorization for recommendation via fully homomorphic encryption. ACM TOPS 21(4), 17:1-17:30.

[19] S.K. Lam, D. Frankowski, and J. Riedl. 2006. Do you trust your recommendations? an exploration of security and privacy issues in recommender systems. In ETRICS 2006, pp, 14?29.

[20] A. Liu, Y. Yao, and X. Cheng. 2020. Recommender systems with condensed local differential privacy. In ML4CS 2020, pp. 355?365.

[21] X. Ma, J. Ma, H. Li, Q. Jiang, and S. Gao. 2018. ARMOR: a trust-based privacy-preserving framework for decentralized friend recommendation in online social networks. FGCS 79, 82?94.

[22] B. Marr. 2017. Data Strategy: How to Profit from a World of Big Data, Analytics and the Internet of Things. Kogan Page.

[23] K. Martin and K. Shilton. 2016. Putting mobile application privacy in context: an empirical study of user privacy expectations for mobile devices. The Information Society 32(3), 200?216.

[24] L. Minto, M. Haller, B. Livshits, and H. Haddadi. 2021. Stronger privacy for federated collaborative filtering with implicit feedback. In ACM RecSys 2021, pp. 342-350.

[25] A Narayanan and V Shmatikov. 2008. Robust de-anonymization of large sparse datasets. In IEEE SP 2008, pp. 111?125.

[26] P. Pu, L. Chen, and R. Hu. 2011. A user-centric evaluation framework for recommender systems. In ACM RecSys 2011, pp. 157?164.

[27] E. Shmueli and T. Tassa. 2020. Mediated secure multi-party protocols for collaborative filtering. ACM TIST 11(2), 1?25.

[28] P. Thongyoo, P. Anantapanya, P. Jamsri, and S. Chotipant. 2020. A personalized food recommendation chatbot system for diabetes patients. In CDVE 2020, pp. 19?28.

[29] K. Tiwari and D. K. Singh. 2022. Machine learning-based recommendation system for disease-drug material and adverse drug reaction: comparative review. Materials Today: Proceedings 51, 304?313.

[30] H. Wang, Q. Zhao, Q. Wu, S. Chopra, A. Khaitan, and H. Wang. 2020. Global and local differential privacy for collaborative bandits. In ACM RecSys 2020, pp. 150?159.

[31] F. Wang, W. Zhong, X. Xu, W. Rafique, Z. Zhou, L. Qi. 2020. Privacy-aware cold-start recommendation based on collaborative filtering and enhanced trust. In IEEE DSAA 2020, pp. 655-662.

[32] S.L. Warner. 1965. Randomized response: a survey technique for eliminating evasive answer bias. J. Amer. Statist. Assoc. 60(309), 63?69.

[33] Q. Yang, Y. Liu, T. Chen, and Y. Tong. 2019. Federated machine learning: concept and applications. ACM TIST 10(2), 1?19.

[34] J. Yi, F. Wu, ChuhanWu, R. Liu, G. Sun, and X. Xie. 2021. Efficient-FedRec: efficient federated learning framework for privacy-preserving news recommendation. In EMNLP 2021, pp. 2814-2824.

[35] Z. Zhang, G. Xu, P. Zhang, and Y. Wang. 2017. Personalized recommendation algorithm for social networks based on comprehensive trust. Applied Intelligence 47, 659-669.

[36] T. Zhao, J. McAuley, and I. King. 2015. Improving latent factor models via personalized feature projection for one class recommendation. In ACM CIKM 2015, pp. 821?830.

[37] R. Zhou, S. Khemmarat, L. Gao, J. Wan, J. Zhang, Y. Yin, and J. Yu. 2016. Boosting video popularity through keyword suggestion and recommendation systems. Neurocomputing 205, 529?541.