DPLCF: Differentially Private Local Collaborative Filtering

Chen Gao, Chao Huang, Dongsheng Lin, Depeng Jin, Yong Li
Beijing National Research Center for Information Science and Technology
Department of Electronic Engineering, Tsinghua University
liyong07@tsinghua.edu.cn

ABSTRACT

Most existing recommender systems leverage users’ complete original behavioral logs, which are collected from mobile devices and stored by the service provider and further fed into recommendation models. This may lead to a high risk of privacy leakage since the recommendation service provider may be trustless. Despite many research efforts on privacy-aware recommendation, the problem of building an effective recommender system completely preserving user privacy is still open.

In this work, we propose a general framework named differentially private local collaborative filtering for recommendation. The designed workflow consists of three steps. First, for accumulated behavioral logs saved on users’ devices, a differentially private protection mechanism is adopted to help obfuscate the real interactions before reporting them to the server. Second, after collecting all obfuscated records from all users, the server runs an estimation model to calculate similarities between each pair of items. This step requires no user-relevant data, and thus it does not introduce any auxiliary privacy risk. Last, the server sends the estimated user-irrelevant item-similarity matrix to each user device, and the recommendation results are inferred locally based on item similarities with each user’s locally stored original behavioral data. To verify our method’s efficacy, we conduct extensive experiments on two real-world datasets, demonstrating that our proposed method achieves the best performance compared with the state-of-the-art baselines. We further demonstrate that our method still works well under various privacy budgets.

CCS CONCEPTS

• **Information systems** → **Collaborative filtering**; **Recommender systems**; • **Security and privacy** → **Privacy-preserving protocols**;

KEYWORDS

Recommender System, Collaborative Filtering, Differential Privacy

ACM Reference Format:
Chen Gao, Chao Huang, Dongsheng Lin, Depeng Jin, Yong Li. 2020. DPLCF: Differentially Private Local Collaborative Filtering. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR ’20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401053

1 INTRODUCTION

Recommender system, which is a typical personalized service nowadays, strongly relies on users’ personal behaviors, such as review, click log, etc., for feeding algorithms. Specifically, recommendation engines estimate users’ preferences from the collected personal data and generate candidates for users’ future consumption. Thus, there is a commonly accepted paradigm that these raw data is fully accessible for the service provider taking charge of the recommendation engine. On the other side, highly private user information (e.g., health condition, political inclinations, etc.) may be extracted or inferred from collected personal data [4, 28]. As a result, the current paradigm for recommendation is faced with a high risk of user privacy leakage.

To address this, researches [3, 7, 20, 21, 25, 27, 28, 30, 35, 41] proposed the task of privacy-preservation recommendation. Among them, many works [3, 7, 28, 35] regard the recommender side as trusted and only preserve user data from being attacked by the third party. This is not reasonable, especially considering the recent news that Facebook is reported to leak user data[1]. Based on this, privacy-preserving recommendation should consider the recommender as a potential attacker. There are two categories of solutions. The first category [25, 30] is based on the data protection mechanism, which adds noise to the raw user data, and only the perturbed data is reported to the server. The second category [20, 21, 41] approaches the problem with decentralized matrix factorization. That is, server and client, *i.e., user*, communicate with latent representation or gradient, and with multiple times of parameter updates, users can learn their embeddings without sending raw data outside.

Despite their effectiveness, we argue that the existing two categories models still suffer from three major limitations:

- **Cannot handle implicit data.** For data-protection based methods, they usually focus on rating data (1-5 rating of movies, for example) or data with special forms (such as trajectory data with GPS coordinates). For decentralized matrix factorization, most researches also only focus on rating data. However, these kinds of data are not universal. For example, there is prevalent implicit behavioral data in today’s online systems, which is the most widely used data for training recommendation models.
- **Privacy leakage in recommendation results.** In data protection methods, recommendation results are calculated at the server side and then sent to each user’s devices later. As the recommendation results are an estimation for future behaviors, similar to true behaviors, these estimated behaviors can also be exploited to infer sensitive information.

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

- **High communication and computation cost.** In decentralized methods, there are frequent data transferring and local computing for decentralized methods, which make it not so realistic to utilize these methods in real-world applications.

In this paper, we approach this unsolved task of designing a privacy-preserving recommendation for implicit feedbacks with high application value. We summarize the key challenges as follows.

- **Protection mechanism for binary data with privacy guarantee.** Different from rating or attributes, interaction data is in the binary form and there is no explicit meaning. With only two kinds of data form, it is challenging to develop a method to protect the binary data with privacy guarantee.
- **Learning from protected binary data is difficult.** It is hard to extract useful predictive signals from a protected data with binary form, since 0 and 1 have totally opposite meaning.
- **Recommendation with low computation and communication cost.** As user device's computation and communication resources are limited, it is challenging to design a recommendation model with few data or parameter change between server and device.

To address the above-mentioned challenges, we design a novel general framework of differential private collaborative filtering for implicit feedback. We adopt a widely used privacy criterion, differential privacy [10] in data collection, and each user only reports protected data to the server. We design a novel estimation method to infer the item similarity from the protected data, serving as the parameter of our item-based recommendation model. Finally, item similarity is sent back to the device and combined with locally stored data to generate recommendation results based on item-based collaborative filtering. This step only requires few computation resources, addressing another key challenge.

To summarize, the main contributions of this work are as follows.

- We propose a novel and general framework for different private collaborative filtering for implicit feedback. Under differential privacy, privacy leakage is bounded with a strict guarantee. In addition, this framework only relies on the interaction data and can be applied to various recommendation scenarios.
- We developed an effective method for extracting item similarity as the predictive signal from the differentially privately protected data. Then we combine this signal with locally stored user raw data to do local collaborative filtering for recommendation.
- We conduct extensive experiments on two real-world datasets in typical recommendation scenarios where exist high risks of privacy leakage. Experimental results demonstrate our proposed DPLCF can effectively outperform the state-of-the-art methods while protecting user privacy. Further studies show that our method can still work well for different noise levels.

The remainder of this paper is as follows. We first formulate the research problem in Section 2. We then elaborate on our proposed method in Section 3.2. We conduct experiments in Section 4, before reviewing related work in Section 5. Lastly, we conclude this paper in Section 6.

2 PRELIMINARIES

We first formulate the problem to solve in this paper and then recapitulate differential privacy, a widely used privacy protection mechanism.

2.1 Problem Formulation

Collaborative filtering based recommendation is based on the basic assumption that users with similar past behaviors will have similar behaviors in the future [36]. Implicit feedbacks [33] refer to the interactions between user and item, which are implicit, such as a click or not. Let M and N denote the number of users and items, respectively, we denote the user-item matrix $\mathbf{R} \in \mathbb{R}^{M \times N}$ with a binary value at each entry defined as follows,

$$r_{ui} = \begin{cases} 1, & \text{if } u \text{ has interacted with } i; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The task of collaborative filtering from implicit feedbacks is to obtain a model estimating the likelihood that a user u will interact with an item i with the input $\mathbf{R} \in \mathbb{R}^{M \times N}$.

Existing methods [19, 24, 33] follow a paradigm that all feedbacks are collected and uploaded to the server and then a model-based or memory-based CF model is deployed at the server to perform recommendation. We argue it is not reasonable since the service provider is not reliable. Thus, the problem of privacy-preserving collaborative filtering for implicit feedbacks requires that the user's true data can only be stored at the user's own device. Furthermore, the recommendation results reflect the user's future behaviors to some extent, and therefore the recommendation procedure (generating results) can also only be conducted at the user side. Last, massive computation, such as mining and extracting CF signal, must be centrally performed at the server only, as the computation resources at the user side are quite limited, and distributed computation has high communication requirements between server and clients.

In brief, the objective of our task is to obtain a model estimating the likelihood that a user u will interact with an item i with following rules:

- **Rule 1**: Users' true historical interactions can only be stored at users' devices locally.
- **Rule 2**: Users' uploaded data should be protected under the differential privacy guarantee.
- **Rule 3**: Users' recommendation results can only be inferred at users' devices locally.

2.2 Differential Privacy

Differential privacy [10] is originally designed to protect each individual's data when releasing aggregated information of a database. A fundamental requirement of differential privacy is that the query of the aggregated information is negligibly affected when a single user's data is modified. Specifically, give two *adjacent* databases D and D', among which there is only one different value of a single individual, the probability that a query returns the same amount for these two databases should be within a bound of ϵ. To achieve differential privacy, a typical solution is to add controlled random noise, such as a noise drawn from Laplace distribution to the query output. The most significant advantage of differential privacy is

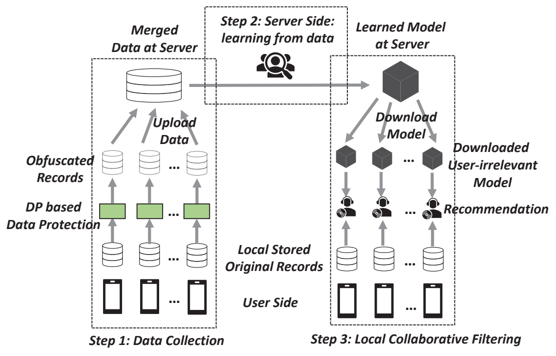


Figure 1: Our proposed framework of privacy-preserving collaborative recommendation for implicit feedbacks

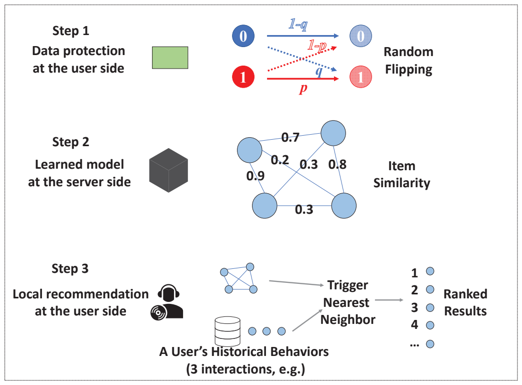

Step 1
Data protection at the user side
1-q
1-p
q
p
Random Flipping
Step 2
Learned model at the server side
0.7
0.2
0.3
0.8
0.9
0.3
Item Similarity
Step 3
Local recommendation at the user side
Trigger Nearest Neighbor
Ranked Results
A User's Historical Behaviors (3 interactions, e.g.)

Figure 2: Our proposed DPLCF method under the privacy-preserving framework

that a mechanism can be proven to be differentially private independently from any side information that the adversary attacker might possess.

There is an equivalent criterion for differential privacy [12]. Given D and D', if a mapping function M satisfies that the output of the M, with the database as the input, belongs to a certain subset of the range of M, formulated as follows,

$$\frac{Pr[M(D) \in S]}{Pr[M(D') \in S]} \leq e^{\epsilon}, \forall S \in Range(M), \tag{2}$$

then the mapping function M satisfies differential privacy.

In the scenario where the recommendation service provider is not reliable, or the user does not agree with providing own historical behavior for penalization, a data collection method based on differential privacy can decouple data collection and recommendation. To be specific, although the service provider (adversary attacker) has some prior knowledge, the privacy leakage is strictly bounded by differential privacy.

3 SYSTEM OVERVIEW AND METHOD

3.1 System Overview

To preserve user data fundamentally to build a recommender system, we propose a novel design of the framework of differentially private collaborative filtering for implicit feedbacks when the recommender is not trusted. In our framework, we integrate the advantages of two mainstream privacy-preserving solutions, data protection [25, 30] and decentralized recommendation [20, 21, 27, 41]. Specifically, it can be divided into three steps, as illustrated in Figure 1.

3.1.1 Data Collection. At the *first step* shown in Figure 1, for each user, we adopt a random perturbation step as the data protection mechanism. Take user u as an example, we apply a random function f on each user's historical behaviors $\mathbf{y}_u = \{j|y_{uj} = 1\}$ to obtain obfuscated images $\hat{\mathbf{y}}_\mathbf{u} = f(\mathbf{y}_u)$. Here we use differential privacy as the guarantee to ensure privacy is protected when an attacker wants to infer whether an uploaded interaction is real or fake.

Input: Original user-item historical behaviors $\mathbf{y}_u$

Output: Obfuscated user-item historical behaviors $\hat{\mathbf{y}}_u$

3.1.2 Data Exploitation. At the *second step*, the server collects obfuscated interactions from each user and merges them into an obfuscated interaction matrix $\hat{R}$[2]. With the collected matrix, we develop an effective estimation model to exploit the noisy data and extract useful predictive signals. Shortly, in this step, we obtain model parameters that can be used to generate recommendation results.

Input: Obfuscated user-item interaction matrix $\hat{R}$.

Output: Parameters of a recommendation model.

3.1.3 Recommendation. At the *third step*, as mentioned above, recommendation results, to some extent, is the estimation for users' future behaviors. Thus, we abandon the traditional manner that the server generates recommendation results centrally, and we send the learned model parameters to each user. Then each user's device uses the locally stored raw behavioral data as the input of the recommendation model, and further, obtain the predicted score for each item candidate.

Input: Parameters of a recommendation model and the original user-item historical behaviors $\mathbf{y}_u$.

Output: Locally generated recommendation results for each user $RecList(u)$.

3.2 The Proposed Method

Under the proposed general framework, we propose a method DPLCF, composed of three phases, namely obfuscated reports uploading, item-based neighborhood finding, and local recommendation computing.

3.2.1 Obfuscated Reports Uploading (Client Side). Recommender systems rely on users' historical behavior for interest estimation. In traditional settings, users' devices upload users' true behaviors without any processing [34]. Here we replace the direct uploading with an obfuscated version. Since implicit feedbacks only have binary values, existing methods for adding continuous noise on rating data [30] cannot be used in our task. To protect the discrete

[2]Note that $\hat{\mathbf{y}}_u$ is the transpose of the u-th row of $\hat{R}$

behavioral data, we adopt a random bit flipping technique under the differential privacy guarantee.

Given the historical interaction record of user u, denoted as $R_{[u,:]}$, which is a N-dimension bit vector, each user needs to upload an obfuscated image $\hat{R}_{[u,:]}$. For each bit R_{ui} in the bit vector, if it's 1, we maintain its original value with probability p and flip it with probability $\overline{p} = 1 - p$; for 0, we flip it with probability q, and with probability $\overline{q} = 1 - q$ we maintain its original value as 0. Namely, given a random variable r sampled from the uniform distribution $\mathbf{U}[0, 1]$, we have

$$\widehat{R}_{ij} = \begin{cases} 1, & \text{if } r \leq p \text{ and } R_{ij} = 1; \\ 0, & \text{if } r > p \text{ and } R_{ij} = 1; \\ 0, & \text{if } r > q \text{ and } R_{ij} = 0; \\ 1, & \text{if } r \leq q \text{ and } R_{ij} = 0. \end{cases} \tag{3}$$

This flipping mechanism is presented at the top of Figure 2.

From **Theorem 3.1** of [38], this random bit flipping mechanism satisfies ϵ-differential privacy for the criteria in (2) with $\frac{p}{q} \leq e^{\epsilon}$. Note that here the privacy is defined for the existence of every interaction record. According to the definition of differential privacy, the presence of the absence of a single interaction record would not have a huge impact on the output of this mechanism [10]. Every user adopted this technique to derive an obfuscated bit vector. Then the obfuscated report is uploaded to the server.

3.2.2 Item-based Neighborhood Finding (Server Side). With each user reporting the obfuscated historical behaviors, these *noisy* vectors can be merged at the server to build a *noisy* interaction matrix. Traditional latent factor models, such as probabilistic matrix factorization [24] , Bayesian personalized ranking [33], or neural matrix factorization [19], use a vector in latent space to represent each user's interests. However, the learning of user vectors is highly sensitive to each data sample (*i.e.* interaction) [45]. Since the collected matrix at the server has been processed by the random flipping, there is a lot of false negative or false positive sample, which largely worsens the learning procedure of latent factor models. On the other hand, the locally stored user vector cannot be further utilized after the user side receives the user embedding sent from the server side. In short, the traditional factorization model for implicit data cannot address the challenge in learning from the obfuscated binary interaction matrix.

To learn from the obfuscated interaction matrix, we address the challenges from another perspective. Since the recommendation is performed in each user's device locally, we utilize item-based collaborative filtering [36] to serve as the model for the server side. To be more precise, the key part of the item-based CF model is to get the similarity score for each item pair; then, the item-similarity can be combined with user history to trigger the recommendation list (historical interacted items can serve as the item-to-item trigger). We propose an estimation-based method to distill useful similarity signals as much as possible.

Given $V_i = R_{[:,i]}$, i.e. the i-th column of the real interaction matrix, recall that the Jaccard similarity between item i and j can be calculated as follows,

$$sim(i, j) = \frac{|V_i \cap V_j|}{|V_i \cup V_j|}. \tag{4}$$

Therefore, the key of estimating item similarities is about providing an accurate estimation model for the cardinality of set $V_i \cap V_j$ and $V_i \cup V_j$. Take two obfuscated images $\widehat{V_1}$ and $\widehat{V_2}$ as examples, which are derived from raw bit vectors V_1 and V_2 using the random flipping mechanism, we denote the number of position s where $\hat{V}_1[s] = i$ and $\hat{V}_2[s] = j$ for $i, j \in \{0, 1\}$ by n_{ij}, and similarly m_{ij} for its counterpart of obfuscated records $\widehat{V_1}$ and $\widehat{V_2}$. Following the manner in [38], the cardinality of set union and intersection can be estimated using a mean-field model as follows,

$$\begin{bmatrix} E(m_{00}) \\ E(m_{01}) \\ E(m_{10}) \\ E(m_{11}) \end{bmatrix} = A \begin{bmatrix} n_{00} \\ n_{01} \\ n_{10} \\ n_{11} \end{bmatrix}, \tag{5}$$

where

$$A = \begin{bmatrix} \overline{q}^2 & \overline{pq} & \overline{pq} & \overline{p}^2 \\ \overline{qp} & \overline{q}p & \overline{p}q & \overline{p}p \\ \overline{qp} & \overline{p}q & \overline{q}p & \overline{p}p \\ q^2 & \overline{p}p & qp & p^2 \end{bmatrix}. \tag{6}$$

By taking the inversion of the matrix, we can derive four unbiased estimators for n_{ij} as follows,

$$\begin{bmatrix} \widehat{n_{00}} \\ \widehat{n_{01}} \\ \widehat{n_{10}} \\ \widehat{n_{11}} \end{bmatrix} = A^{-1} \begin{bmatrix} m_{00} \\ m_{01} \\ m_{10} \\ m_{11} \end{bmatrix}. \tag{7}$$

Recall that the length of every bit vector is M (*i.e., number of users*), n_{00} has the following relationship with $|V_1 \cup V_2|$,

$$n_{00} = M - |V_1 \cup V_2|. \tag{8}$$

Then $|V_1 \cup V_2|$ can be estimated as,

$$\widehat{|V_1 \cup V_2|} = M - \widehat{n_{00}}. \tag{9}$$

Meanwhile, $\widehat{n_{11}}$ directly serves as an estimator of $|V_1 \cap V_2|$ as follows,

$$\widehat{|V_1 \cap V_2|} = \widehat{n_{11}}. \tag{10}$$

After obtaining the similarity matrix, the neighborhood can be easily computed by finding items with top-K highest similarity values. Then the server broadcasts the neighborhood of each item (with corresponding similarity values) to every user.

Recall that given a privacy budget ϵ, only the ratio $\frac{p}{q}$ is decided as e^{ϵ}. To find the exact values of p and q, another condition is still required. One solution taken by most related works is using symmetric flipping, where $p = 1 - q$. The word "symmetric" indicates that the probability of flipping an "1" to a "0" is the same as that in the reverse way. In this way, the inversion of matrix A is also symmetric and easy to calculate.

Another solution is about using asymmetric flipping, where $p \neq 1 - q$. Although p can take any arbitrary value among $[\frac{1}{2}, 1]$ theoretically, it is not always easy to derive a neat analytical solution of A^{-1}. Therefore, here we consider an extreme case where $p = 1$, and consequently, $q = \frac{p}{e^{\epsilon}}$. It still satisfy ϵ-differential privacy for the criteria in (2). Then A^{-1} can be easily calculated as follows,

$$A^{-1} = \frac{1}{1-q} \begin{bmatrix} \frac{1}{1-q} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\frac{q^2}{1-q} & 0 & -q & 1-q \end{bmatrix}. \quad (11)$$

Both two kinds of flipping manners satisfy the differential privacy and the choice can be considered as a hyper-parameter, of which the optimal setting may be different for different datasets. We name the method with asymmetric flipping as DPLCF-AP and the method with symmetric flipping as DPLCF-SP.

3.2.3 Local Recommendation Computing. After the server obtains the item-item similarity matrix at the second stage, then at the third stage of our proposed framework, recommendation results are calculated locally to protect the privacy further. For each user, the true interaction history is stored in the user device, and receive an item-item similarity matrix.

We first find the neighborhood $KNN(j)$ for every item j by filtering out *number_of_neighbour* items with top highest similarity $sim(j, \cdot)$. Then the score of a candidate i is derived as a weighted sum of interaction records within the neighborhood,

$$s(u, i) = \sum_{j \in KNN(i)} sim(i, j) \cdot R_{uj}, \quad (12)$$

where $sim(i, j)$ is the similarity measure between item i and j, which can be cosine similarity, Jaccard similarity, or others. In this paper, we use Jaccard similarity as the similarity metric, so that the estimation model in the second stage can be directly utilized.

3.2.4 Complexity Analysis. For the user side, the local device is involved in Step 1 and Step 3. For the first step, the computation complexity and communication cost of data protection are linear with the interaction number of each user; for the third step, the computation complexity is $O(log(N) * number\_of\_neighbour)$. The server side is involved in Step 2. At this step, the computation complexity is $O(|R|)$, linear with the number of all interactions.

For decentralized algorithms, one primary concern is the communication efficiency across clients or between clients and servers, since high time delay will cause bad user experience and algorithms' low performance. In our proposed solution, there is one uploading and one downloading procedure for the whole recommendation, which is far more efficient than some existing decentralized methods which require multiple times' data or model update. Another concern is the computation consumption at local devices since the computational ability of users' devices is always limited. In our proposed DPLCF, this key limitation is well addressed.

4 EXPERIMENTS

We conduct experiments on two real-world datasets to answer the following research questions.

- **RQ1:** How does our proposed method perform compared with the state-of-the-art recommendation methods?
- **RQ2:** How does the noise level in data protection affect our proposed method and other methods?
- **RQ3:** How do the hyper-parameters affect recommendation performance for our proposed method?

Table 1: Statistics of the datasets

dataset	#Users	#Items	#Interactions	Sparsity
AppUsage	871	1,682	51,935	96.455%
MovieLens	74,529	9,953	18,848,812	97.459%

In what follows, we first describe the experimental settings and then answer the above three research questions.

4.1 Experimental Settings

4.1.1 Datasets and Evaluation Protocol. We experimented with two real-world datasets from the most mainstream recommendation services.

- **AppUsage Dataset.** This dataset is the record of users' mobile App usage, released in [44]. Each entry in this record contains an anonymized user identification, timestamps, and an ID of the used mobile App. This dataset is quite suitable for our task since App usage data is private for users, and a lot of App service providers collect such data for personalized recommendations.
- **MovieLens Dataset.** This is a famous public movie rating dataset[3]. Although ratings of movies are typically considered insensitive, they may be exploited to infer some sensitive information such as health condition and political inclinations [4]. Since the original MovieLens-20M dataset is quite large, to fit it into the memory, we only retain users and items with at least 60 interaction records. We convert each rating into an interaction to build an implicit feedback dataset.

The statistics of final evaluation datasets are summarized in Table 1. In the evaluation stage, given a user in the testing set, each algorithm ranks all items that the user has not interacted with before. We applied the widely used leave-one-out technique [18, 19] to obtain the training set and test set. Specifically, for each user, choose the last-interacted item as the test item. We then adopted two popular metrics, HR and NDCG, to judge the performance of the ranking list:

- **HR@K:** *Hit Ratio* (HR) measures whether the test item is contained by the top-K item ranking list (1 for yes and 0 for no).
- **NDCG@K:** *Normalized Discounted Cumulative Gain* (NDCG) complements HR by assigning higher scores to the hits at higher positions of the ranking list.

4.1.2 Baselines. We compared our method with four baselines, which can be divided into two groups according to whether the recommendation manner is centralized or decentralized.

The first group consists of decentralized methods, for which prediction scores are calculated in the user side when obtaining recommendation results.

- **LCF-AP** This is a noise-unaware method, that removes the estimation part in our DPLCF-AP model. In other words, it directly calculates the item similarity with obfuscated interaction data as [36].
- **LCF-SP** This is a noise-unaware method when we set $p = 1 - q$ in Eqn (3). The 0-to-1 and 1-to-0 flipping is symmetric, and we name it as LCF-SP (local collaborative filtering with symmetric

[3]https://grouplens.org/datasets/movielens/

flipping). That is, it removes the estimation part in our DPLCF-SP model.

For the second group, we use state-of-the-art matrix factorization models, BPR [33] and GMF [19], as the centralized recommendation methods to compare with. For these two methods, data is merged in the server, and recommendation results are generated centrally. Thus, they cannot completely protect user privacy.

- **BPR-MF** [33] Bayesian personalized ranking for matrix factorization (BPR-MF) is the state-of-the-art model matrix factorization method for implicit feedback. It utilizes a pairwise loss function to maximize the margin between positive samples and negative samples.
- **GMF** This is one of three variants proposed in [19], which extends traditional matrix factorization by utilizing a vector to assign various weights to different dimensions before adopting the inner product.

We also report the recommendation performance of ItemCF [36] without protection mechanism in data collection, which can achieve the upper-bound performance of item-based collaborative filtering. Thus, we name it **LCF-UB**.

It is worth mentioning that existing recommendation methods with privacy-preserving data collection [25, 30] cannot be compared in this paper, as these works are only suitable for rating data or data in some special forms.

4.1.3 Parameter Settings. We choose the size of the neighborhood as 10 for the AppUsage dataset and 20 for MovieLens dataset considering the scale of these two datasets. Although further fine-tuning of this parameter can improve these models' overall performances, the optimal choice varies with different amounts of privacy budget and diverse application scenarios.

For two traditional centralized models, BPR-MF and GMF, we tune the hyper-parameters, including learning rate, regularization term, and number of latent factors through cross-validations. The grid search range for the optimal learning rate is $[0.001, 0.0005, 0.0001]$, together with $[0.001, 0.0001, 0]$ for regularization term and $[4, 8, 16]$ for the number of latent factors. The batch size and the negative sampling ratio are fixed as 1024 and 4, respectively, which are common settings in existing researches [19, 33].

4.2 Performance Comparison (RQ1)

We first compare the top-K recommendation performance with the baselines. We investigate the top-K performance with setting K from 2 to 10, as we found results are not so stable when setting K to 1. Note that we sampled 99 unobserved items for each interaction to build the ranking list, which is a widely accepted setting [19]. This operation makes the values of the performance metric of different datasets more friendly.

We present the performance in Table 2, 3, 4 and 5 for our DPLCF methods and baseline methods. All results are calculated as the averaged values of repetitive experiments. From the results, we have the following observations:

- **DPLCF achieves the best performance.** Our proposed DPLCF method obtains the best performance in terms of HR@K and NDCG@K as compared to all baselines. The one-sample paired t-tests indicate the all improvements are statistically for $p < 0.05$, which makes sure these results are reliable. For the AppUsage dataset, with asymmetric flipping, our DPLCF-AP can achieve the best performance compared with baselines. Specifically, DPLCF-AP can outperform the best baseline LCF-AP by 2.74% to 6.18% in HR and 1.61% to 3.56% in NDCG. For the MovieLens dataset, with symmetric flipping, our DPLCF-SP can achieve the best performance compared with baselines. Specifically, DPLCF-SP can outperform the best baseline LCF-AP by 1.04% to 14.77% in HR and 6.82% to 15.49% in NDCG. In short, for the utilized datasets, choosing a proper flipping manner can obtain the best performance.
- **Choosing asymmetric or symmetric noise is determined by the dataset** As we can observe, DPLCF-AP performs best on the AppUsage dataset, and DPLCF-SP performs best on the MovieLens dataset. This can be explained that datasets' characteristics play significant roles, including scale, sparsity, etc. For the AppUsage dataset, the interaction is sparser, which means users' positive records are fewer. Thus, an asymmetric noise that does less *damage* to positive records is a better flipping manner.
- **Traditional centralized method is not suitable for our task.** For BPR-MF and GMF, which are two widely-used competitive matrix factorization models, the recommendation performances are very poor at the AppUsage dataset. For MovieLens datasets, as there are too many samples in the merged protection interaction matrix, we do not report the results after struggling with the tunning for a long time. Note that this is also a severe shortcoming since the complexity of these centralized methods is highly related to the number of positive records. Thus, when we use privacy-protection mechanisms to generate many fake positive records, these methods have very low application value due to the efficiency limit. On the other side, as mentioned before, the centralized recommendation manner for BPR-MF and GMF still suffers from privacy concerns, even if the data for training BPR-MF or GMF has been protected with differential privacy. This is because the recommendation results reflecting user interests are accessible for the server while our proposed DPLCF perform recommendation locally. The server side only possesses a user-irrelevant item-similarity matrix.
- **Privacy has a various influence on utility for different datasets** As for experimental results in Table 2, 3, 4 and 5, we set the noise level ϵ to 1.0, which is a widely used value in researches on differential privacy. However, with different characteristics of datasets, such as various long-tail effects among different datasets, the utility with $\epsilon = 1.0$ is diverse. We can observe that for the Moive-Lens datasets, the performance gap between the optimal method without any privacy protection and our DPLCF is small. For the AppUsage dataset, the gap is relatively large, and other privacy-preserving methods are the same. The relatively large gap *does not mean a failure* of our DPLCF since users' data is collected privately, and recommendation results are inferred locally. This interesting phenomenon shows how data characteristics influence the trade-off between privacy and utility, and we leave it as future work.

Table 2: HR Performance comparison among methods on AppUsage dataset

TopK	2	3	4	5	6	7	8	9	10
LCF-UB	0.5885	0.6989	0.7609	0.7977	0.8195	0.8414	0.8586	0.8655	0.8713
BPR-MF	0.2241	0.2621	0.2966	0.3356	0.3552	0.3839	0.4138	0.4483	0.4770
GMF	0.2218	0.2667	0.3092	0.3368	0.3644	0.3966	0.4230	0.4414	0.4678
LCF-AP	0.3345	0.3897	0.4368	0.4678	0.4977	0.5207	0.5471	0.5609	0.5839
LCF-SP	0.3080	0.3655	0.4080	0.4368	0.4747	0.4977	0.5207	0.5333	0.5517
DPLCF-AP	**0.3448**	**0.4138**	**0.4517**	**0.4874**	**0.5184**	**0.5437**	**0.5621**	**0.5770**	**0.5943**
DPLCF-SP	0.2264	0.2609	0.2851	0.2977	0.3149	0.3299	0.3425	0.3575	0.3747

Table 3: NDCG Performance comparison among methods on AppUsage dataset

TopK	2	3	4	5	6	7	8	9	10
LCF-UB	0.5172	0.5724	0.5991	0.6134	0.6211	0.6284	0.6339	0.6359	0.6376
BPR-MF	0.1949	0.2138	0.2287	0.2438	0.2508	0.2603	0.2698	0.2801	0.2885
GMF	0.1930	0.2154	0.2337	0.2444	0.2542	0.2649	0.2733	0.2788	0.2865
LCF-AP	0.3039	0.3315	0.3518	0.3638	0.3745	0.3821	0.3905	0.3946	0.4013
LCF-SP	0.2792	0.3079	0.3263	0.3374	0.3509	0.3585	0.3658	0.3696	0.3749
DPLCF-AP	**0.3088**	**0.3433**	**0.3596**	**0.3734**	**0.3844**	**0.3929**	**0.3987**	**0.4032**	**0.4081**
DPLCF-SP	0.2103	0.2276	0.2380	0.2428	0.2490	0.2540	0.2580	0.2625	0.2674

Table 4: HR Performance comparison among methods on MovieLens dataset

TopK	2	3	4	5	6	7	8	9	10
LCF-UB	0.4675	0.5695	0.6431	0.7006	0.7446	0.7804	0.8081	0.8318	0.8505
LCF-AP	0.3662	0.4454	0.5060	0.5541	0.5935	0.6280	0.6563	0.6806	0.7030
LCF-SP	0.3661	0.4430	0.4999	0.5440	0.5802	0.6096	0.6354	0.6577	0.6763
DPLCF-AP	0.3513	0.4274	0.4880	0.5351	0.5757	0.6126	0.6424	0.6711	0.6962
DPLCF-SP	**0.4203**	**0.5003**	**0.5559**	**0.5969**	**0.6271**	**0.6520**	**0.6716**	**0.6877**	0.7000

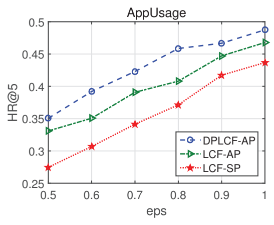
Table 5: NDCG Performance comparison among methods on MovieLens dataset

TopK	2	3	4	5	6	7	8	9	10
LCF-UB	0.4122	0.4633	0.4950	0.5172	0.5329	0.5448	0.5536	0.5607	0.5661
LCF-AP	0.3234	0.3629	0.3891	0.4077	0.4217	0.4332	0.4421	0.4494	0.4559
LCF-SP	0.3241	0.3625	0.3871	0.4041	0.4170	0.4268	0.4349	0.4417	0.4470
DPLCF-AP	0.3110	0.3491	0.3752	0.3934	0.4079	0.4201	0.4296	0.4382	0.4454
DPLCF-SP	**0.3735**	**0.4135**	**0.4374**	**0.4533**	**0.4641**	**0.4724**	**0.4786**	**0.4834**	**0.4870**

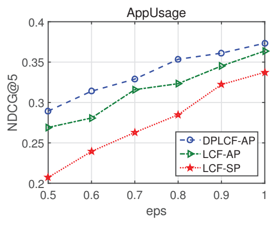
4.3 Impact of Noise Level (RQ2)

In the above experiments, we set the noise level of differential privacy, ϵ, to 1.0. To understand how the noise level affects the recommendation performance, we choose the AppUsage dataset for further investigation. We choose two widely accepted values of $K = 5$ and 10, to present the ranking performance of our methods and baseline methods. We set the range of ϵ to a range from 0.5 to 1.0 with a step size of 0.1. We present the experimental results in Figure 3, respectively. From the results, we can observe that our DPLCF always achieves the best performance on all top-K metrics for different ϵ among two datasets. As for the most widely used range, 0.5 to 1.0, of ϵ, DPLCF's recommendation performance is steadily better than other methods. When the noise level becomes extremely large (corresponding to a very small ϵ, such as 0.5), there is a performance decrease in HR@10. We analyzed the obfuscated data and found that when setting ϵ to 0.5, the reported protected data contains a very large number of items for each user. This makes users' reported records lose diversity. It further makes the estimated item-similarity not so accurate, which causes the performance to decrease. Nevertheless, our method still achieves the best performance under other metrics. It is also worth mentioning that metrics with smaller topK are more significant, and NDCG is a more strict metric compared with HR.

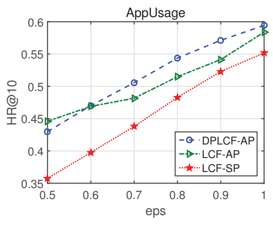
In summary, with a reasonable and acceptable range of ϵ, from 0.5 to 1.0, our DPLCF can steadily outperform other baseline methods, verifying its effectiveness in real-world applications.



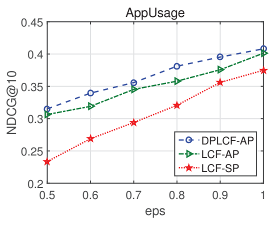
(a) HR@5



(b) NDCG@5



(c) HR@10



(d) NDCG@10

Figure 3: Top-K recommendation performance comparison on the AppUsage dataset (K is set to 5 and 10 and ϵ is set to 0.5 to 1.0).

Table 6: (a) NDCG@5 Performance comparison among methods on MovieLens dataset with different neighbor size (b) NDCG@10 Performance comparison among methods on MovieLens dataset with different neighbor size

(a)

#Neighbor	20	30	40	50
LCF-AP	0.4077	0.4170	0.4212	0.4245
LCF-SP	0.4041	0.4156	0.4195	0.4262
DPLCF-SP	**0.4533**	**0.4517**	**0.4500**	**0.4491**

(b)

#Neighbor	20	30	40	50
LCF-AP	0.4559	0.4666	0.4717	0.4755
LCF-SP	0.4470	0.4612	0.4666	0.4751
DPLCF-SP	**0.4870**	**0.4872**	**0.4861**	**0.4859**

4.4 Impact of Hyperparameters (RQ3)

To understand how hyper-parameters impact the performance, we focus on the number of neighbors in item-based collaborative filtering. We choose MovieLens dataset and present the recommendation performance setting the number from 20 to 50. The NDCG@5 and NDCG@10 recommendation results are shown in Table 6. From the results, we can observe that our DPLCF-SP can achieve the best performance for all settings. Note that in the stage of local recommendation, a larger number of neighbor means higher computation cost. Considering this stage's computation is employed in users' devices, the results that our DPLCF-SP can achieve a large performance improvement, further verify its effectiveness in a real-world scenario.

In conclusion, we conduct extensive experiments on two real-world datasets, verifying that our proposed DPLCF can outperform existing methods while protecting user privacy when choosing a proper flipping manner. Further studies demonstrate our model still works well for different noise levels.

5 RELATED WORK

5.1 Privacy-preserving Recommendation

Recommender systems are with a close relation to users' personal information or demographics. Some early researches [1, 11] have shown that those user preferences not so sensitive, such as ratings of movies or App usage records, can still be utilized to infer sensitive information, such as health condition or political inclinations. This drives the rapid development of privacy-preserving recommendation. Generally speaking, existing researches can be categorized into two groups according to whether the recommender is trusted or not.

Trusted Recommender For privacy-preserving recommendation of which the recommender itself is trusted, the adversary is defined as the attacker who wants to infer private user information from the released model (user embedding, for example) or recommendation results. For protecting the trained model, some works introduced privacy-preserving mechanism in training models [7, 17, 26, 28, 46, 47]. Chen *et al.* [7] presented an MF based model which splits latent user vectors into local and global parts to protect user privacy in the task of point-of-interest recommendation. Mcsherry *et al.* [28] introduced differential privacy during the training of the MF model via adding noise to latent factors. For the latter, some works applied protection mechanism to final recommendation results [3, 35]. Riboni *et al.* [35] proposed the use of differential privacy to extract statistics about users' preferences and then provided recommendation from those statistics. Berlioz *et al.* [3] applied perturbation to MF's output, the recommendation results, to meet the criterion of differential privacy.

Distrusted Recommender For the privacy-preserving recommendation of which the recommender is not trusted, the recommendation service provider is regarded as the potential attacker. Thus, the problem is to keep the raw user data not accessible for the recommender. There are two types of solutions to this problem in existing researches. There are works [25, 30] adopting protection mechanism when collecting data for recommendation. Li *et al.* [25] studied privacy-preserving data collection for point-of-interest recommendation. The authors first transformed the raw user trajectory into a bipartite graph and then extracted the association matrix to inject carefully calibrated noise to meet differential privacy. This matrix is further used for recommendation. Plat *et al.* [30] proposed to adopt data disguising for each user's rating data before uploading to the server. Then the disguised rating data is merged at the server for SVD-based recommendation. However, these researches only take rating data or data in particular form into consideration. In other words, these methods cannot be applied to implicit feedback data.

Another solution is to perform decentralized recommendation. Unlike in a centralized recommendation of which the user side only provides data, decentralized recommendation works in a cooperative manner (user-server or multiple users). There are some works utilizing distributed matrix factorization [16] to perform decentralized recommendation [20, 21, 27, 41]. User privacy is protected to some extent, with only transferring gradient or parameter between the user and the server side. However, this distributed MF based methods are only suitable for rating data. Recently, Chen [8] defined the decentralized recommendation as a task of federated learning. That is, each user is considered as a federation not willing to share raw data but to hope to benefit from others. A meta-learning based method is utilized to solve the task. In this paper, we only use CF data, and this method can not be applied as it relies on attributes to transfer knowledge across federations.

Federated learning [23, 37] is a machine learning technique allowing multiple clients (so-called federation) take participants in the model training stage without directly sharing data. However, it is required in federated learning that there are some homogeneous attributes across these clients' data. In our task for a privacy-preserving recommendation for implicit data, for each client, only discrete interaction data is used for recommendation. Such discrete 0/1 data is hard to transfer knowledge across clients and therefore, we federated learning based methods cannot be applied in our task. There are also some works [14, 15] study a similar problem, privacy-preserving cross-domain recommendation, of which multiple service providers share data to enhance recommendation.

In our paper, we also follow the more strict assumption that the recommender is not trusted. We use differential privacy for protecting data and decentralized recommender manner for further protecting privacy. In short, our model is featured with advantages of two mainstream privacy-preserving recommender systems with distrusted recommenders.

5.2 Differentially Private Data Collection

Differential privacy is widely used in data collection. Different from the application in data release [6, 9, 39], applying differential privacy to data collection aims to protect each user's data, rather than in an aggregated manner. Some works combine differential privacy with location privacy to collect user's location data while preserving privacy [2, 5]. Some other works proposed to utilize differential privacy to crowd-sourcing [13, 40] and data with more complicated form [42, 43]. There are some works extending data collection into data sharing across multiple data owners with the help of differential privacy [29, 32].

Local differential privacy (LDP) [17, 22, 31], is a differential privacy protocol used to collect user data. Readers may confuse LDP with our proposed framework, especially the data collection step. Actually, our framework belongs to DP-based models rather than LDP-based ones. To be precise, LDP aims to preserve user privacy by reducing the impact of the participation of a certain *user*. It usually requires more effort to provide a privacy guarantee for users with many interaction records. Different from this, our proposed model aims to reduce the impact of a certain *interaction*. In the recommendation task, removing the effects of all records from a user is too harsh, making it impossible to provide personalized recommendations. Therefore, we adopt the DP-based notation to preserve data utility as much as possible.

6 CONCLUSION

In this paper, we present a new privacy-preserving solution for building recommendation for implicit collaborative filtering, which considers the privacy issue in both data collection and recommendation results inferring. We conduct extensive experiments on two real-world datasets, demonstrating that our proposed DPLCF method can improve the recommendation performance almost on all the metrics while preserving user privacy. To the best of our knowledge, this is the first work to approach a general privacy-preserving recommendation framework for implicit feedbacks with a differential privacy guarantee, considering both privacy-preserving data collection and privacy-preserving decentralized recommendation.

ACKNOWLEDGMENTS

This work was supported in part by The National Key Research and Development Program of China under grant 2018YFB1800804, the National Natural Science Foundation of China under U1936217, 61971267, 61972223, 61941117, 61861136003, Beijing Natural Science Foundation under L182038, Beijing National Research Center for Information Science and Technology under 20031887521, and research fund of Tsinghua University - Tencent Joint Laboratory for Internet Innovation Technology.

REFERENCES

[1] Esma Aïmeur, Gilles Brassard, José M Fernandez, and Flavien Serge Mani Onana. 2008. A lambic: a privacy-preserving recommender system for electronic commerce. *International Journal of Information Security* 7, 5 (2008), 307–334.
[2] Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. 2013. Geo-indistinguishability: Differential privacy for location-based systems. In *CCS*. 901–914.
[3] Arnaud Berlioz, Arik Friedman, Mohamed Ali Kaafar, Roksana Boreli, and Shlomo Berkovsky. 2015. Applying differential privacy to matrix factorization. In *RecSys*. 107–114.
[4] Joseph A Calandrino, Ann Kilzer, Arvind Narayanan, Edward W Felten, and Vitaly Shmatikov. 2011. " You Might Also Like:" Privacy Risks of Collaborative Filtering. In *IEEE S&P*. 231–246.
[5] Yang Cao, Yonghui Xiao, Li Xiong, and Liquan Bai. 2019. PriSTE: from location privacy to spatiotemporal event privacy. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1606–1609.
[6] Yang Cao, Masatoshi Yoshikawa, Yonghui Xiao, and Li Xiong. 2018. Quantifying differential privacy in continuous data release under temporal correlations. *IEEE Transactions on Knowledge and Data Engineering* (2018).
[7] Chaochao Chen, Ziqi Liu, Peilin Zhao, Zhou Jun, and Li Xiaolong. 2018. Privacy Preserving Point-of-Interest Recommendation Using Decentralized Matrix Factorization. In *AAAI*.
[8] Fei Chen, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. 2018. Federated meta-learning for recommendation. *arXiv preprint arXiv:1802.07876* (2018).
[9] Rui Chen, Noman Mohammed, Benjamin CM Fung, Bipin C Desai, and Li Xiong. 2011. Publishing set-valued data via differential privacy. *Proceedings of the VLDB Endowment* 4, 11 (2011), 1087–1098.
[10] Cynthia Dwork. 2008. Differential privacy: A survey of results. In *TAMC*. 1–19.
[11] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.
[12] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.
[13] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the*

2014 ACM SIGSAC conference on computer and communications security. ACM, 1054–1067.
[14] Chen Gao, Xiangning Chen, Fuli Feng, Kai Zhao, Xiangnan He, Yong Li, and Depeng Jin. 2019. Cross-domain Recommendation Without Sharing User-relevant Data. In *The World Wide Web Conference*. 491–502.
[15] Chen Gao, Chao Huang, Yue Yu, Huandong Wang, Yong Li, and Depeng Jin. 2019. Privacy-preserving Cross-domain Location Recommendation. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 1 (2019), 1–21.
[16] Rainer Gemulla, Erik Nijkamp, Peter J Haas, and Yannis Sismanis. 2011. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 69–77.
[17] Rachid Guerraoui, Anne-Marie Kermarrec, Rhicheek Patra, and Mahsa Taziki. 2015. D 2 p: distance-based differential privacy in recommenders. *Proceedings of the VLDB Endowment* 8, 8 (2015), 862–873.
[18] Jiayuan He, Jianzhong Qi, and Kotagiri Ramamohanarao. 2019. A Joint Context-Aware Embedding for Trip Recommendations. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 292–303.
[19] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.
[20] István Hegedűs, Árpád Berta, Levente Kocsis, András A Benczúr, and Márk Jelasity. 2016. Robust decentralized low-rank matrix decomposition. *ACM Transactions on Intelligent Systems and Technology (TIST)* 7, 4 (2016), 62.
[21] Jingyu Hua, Chang Xia, and Sheng Zhong. 2015. Differentially Private Matrix Factorization. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15)*. AAAI Press, 1763–1770. http://dl.acm.org/citation.cfm?id=2832415.2832494
[22] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. 2014. Extremal mechanisms for local differential privacy. In *Advances in neural information processing systems*. 2879–2887.
[23] Jakub Konečný, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).
[24] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).
[25] Chao Li, Balaji Palanisamy, and James Joshi. 2017. Differentially private trajectory analysis for points-of-interest recommendation. In *BigData Congress*. 49–56.
[26] Jianqiang Li, Ji-Jiang Yang, Yu Zhao, Bo Liu, Mengchu Zhou, Jing Bi, and Qing Wang. 2016. Enforcing differential privacy for shared collaborative filtering. *IEEE Access* 5 (2016), 35–49.
[27] Qing Ling, Yangyang Xu, Wotao Yin, and Zaiwen Wen. 2012. Decentralized low-rank matrix completion. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2925–2928.
[28] Frank McSherry and Ilya Mironov. 2009. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *SIGKDD*. 627–636.
[29] Huseyin Polat and Wenliang Du. 2005. Privacy-preserving top-n recommendation on horizontally partitioned data. In *WI*. 725–731.
[30] Huseyin Polat and Wenliang Du. 2005. SVD-based collaborative filtering with privacy. In *Proceedings of the 2005 ACM symposium on Applied computing*. ACM, 791–795.
[31] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2016. Heavy hitter estimation over set-valued data with local differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 192–203.
[32] Arun Rajkumar and Shivani Agarwal. 2012. A differentially private stochastic gradient descent algorithm for multiparty classification. In *Artificial Intelligence and Statistics*. 933–941.
[33] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.
[34] Paul Resnick and Hal R Varian. 1997. Recommender systems. *Commun. ACM* 40, 3 (1997), 56–59.
[35] Daniele Riboni and Claudio Bettini. 2012. Private context-aware recommendation of points of interest: An initial investigation. In *PERCOM Workshops*. 584–589.
[36] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. 285–295.
[37] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. 2017. Federated multi-task learning. In *Advances in Neural Information Processing Systems*. 4424–4434.
[38] Rade Stanojevic, Mohamed Nabeel, and Ting Yu. 2017. Distributed cardinality estimation of set operations with differential privacy. In *2017 IEEE Symposium on Privacy-Aware Computing (PAC)*. IEEE, 37–48.
[39] Sen Su, Peng Tang, Xiang Cheng, Rui Chen, and Zequn Wu. 2016. Differentially private multi-party high-dimensional data publishing. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 205–216.
[40] Hien To, Gabriel Ghinita, and Cyrus Shahabi. 2014. A framework for protecting worker location privacy in spatial crowdsourcing. *Proceedings of the VLDB Endowment* 7, 10 (2014), 919–930.
[41] Yu Xin and Tommi Jaakkola. 2014. Controlling privacy in recommender systems. In *Advances in Neural Information Processing Systems*. 2618–2626.
[42] Shengzhi Xu, Sen Su, Li Xiong, Xiang Cheng, and Ke Xiao. 2016. Differentially private frequent subgraph mining. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 229–240.
[43] Jianyu Yang, Xiang Cheng, Sen Su, Rui Chen, Qiyu Ren, and Yuhan Liu. 2019. Collecting Preference Rankings Under Local Differential Privacy. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1598–1601.
[44] Donghan Yu, Yong Li, Fengli Xu, Pengyu Zhang, and Vassilis Kostakos. 2018. Smartphone app usage prediction using points of interest. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)* 1, 4 (2018), 174.
[45] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 785–788.
[46] Tianqing Zhu, Gang Li, Yongli Ren, Wanlei Zhou, and Ping Xiong. 2013. Differential privacy for neighborhood-based collaborative filtering. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 752–759.
[47] Xue Zhu and Yuqing Sun. 2016. Differential privacy for collaborative filtering recommender algorithm. In *Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics*. 9–16.