Stronger Privacy for Federated Collaborative Filtering With Implicit Feedback

Lorenzo Minto
Moritz Haller
Brave Software
London, UK

Hamed Haddadi
Benjamin Livshits
Brave Software
London, UK
Imperial College London
London, UK

ABSTRACT

Recommender systems are commonly trained on centrally-collected user interaction data like views or clicks. This practice however raises serious privacy concerns regarding the recommender's collection and handling of potentially sensitive data. Several privacy-aware recommender systems have been proposed in recent literature, but comparatively little attention has been given to systems at the intersection of implicit feedback and privacy. To address this shortcoming, we propose a practical federated recommender system for implicit data under user-level local differential privacy (LDP). The privacy-utility trade-off is controlled by parameters ϵ and k, regulating the per-update privacy budget and the number of ϵ-LDP gradient updates sent by each user, respectively. To further protect the user's privacy, we introduce a proxy network to reduce the fingerprinting surface by anonymizing and shuffling the reports before forwarding them to the recommender. We empirically demonstrate the effectiveness of our framework on the MovieLens dataset, achieving up to *Hit Ratio* with K=10 (HR@10) 0.68 on 50,000 users with 5,000 items. Even on the full dataset, we show that it is possible to achieve reasonable utility with HR@10>0.5 without compromising user privacy.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Collaborative filtering**; • **Security and privacy** → **Privacy-preserving protocols**; • **Computing methodologies** → *Distributed artificial intelligence*; *Learning from implicit feedback*.

KEYWORDS

federated learning, recommender systems, local differential privacy

ACM Reference Format:
Lorenzo Minto, Moritz Haller, Hamed Haddadi, and Benjamin Livshits. 2021. Stronger Privacy for Federated Collaborative Filtering With Implicit Feedback. In *Fifteenth ACM Conference on Recommender Systems (RecSys '21), September 27-October 1, 2021, Amsterdam, Netherlands.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3460231.3474262

RecSys '21, September 27-October 1, 2021, Amsterdam, Netherlands

ACM ISBN 978-1-4503-8458-2/21/09…$15.00
https://doi.org/10.1145/3460231.3474262

1 INTRODUCTION

For many years now, *recommender systems* have been an integral part of the users' experience: helping personalize software products around users' particular tastes and needs. More specifically, *recommender systems* are used to estimate the user's preferences from a history of collected personal feedback, which can be either explicit, such as movie ratings, or implicit, which comprises more generic interaction data like views or clicks.

While such systems can significantly enhance user experience, they usually do so at the expense of privacy, by centrally collecting user data. Several studies [3, 16, 20, 22, 23, 34] have focused on the privacy risks that come with the centralised approach. Some of these studies [20, 23, 34] highlight how even seemingly non-sensitive data like movie ratings can be used to infer age, gender, and political affiliation of users. Moreover, Calandrino et al. [3] have shown how *recommender systems* can be easy targets of inference attacks, whereby the attacker manipulates the recommender to reveal personal information about other users. Finally, an inherent risk of any centralized system is the violation of privacy promises made by the data curator, either knowingly for illegitimate purposes, or through unintended data breaches[1].

These concerns motivate the need for a privacy-first approach. More specifically, we suggest:

(1) No interaction data should be collected or stored by the recommender.
(2) User contributions should be protected by user-level differential privacy.
(3) Recommendation should be local and on-device.

Current work on privacy-preserving recommendation frameworks either fails to provide formal privacy guarantees [1, 5] or relies on direct or partial access to users' interaction data [9, 14]. Shin et al. [28] address all the above concerns but rely on a combination of dimensionality reduction through random projection and sparse recovery to achieve reasonable utility. Their method is only suitable when the unperturbed gradient data is sparse, as is the case for recommendation with explicit feedback (i.e. ratings).

We propose a method to achieve all of the above, while minimizing trust on behalf of the user and maintaining reasonable utility. We do so by enhancing existing, federated recommendation methods with a privacy mechanism to prevent reconstruction attacks from gradient data [12, 33]. For each user, we control the privacy-utility trade-off by selecting a subset of k factors from the item gradient update matrix and applying a privatization mechanism

[1] https://en.wikipedia.org/wiki/List_of_data_breaches

to each that ensures the perturbed factor meets ϵ-local differential privacy. To reduce the fingerprinting surface and prevent a malicious recommender from building longitudinal representations of the user, we introduce a proxy network that breaks linkability between subsequent reports belonging to the same user by stripping metadata from said user's traffic and shuffling it together with reports from other users.

1.1 Contributions.

In summary, in this work we make the following contributions:

(1) We propose a federated recommender system for implicit feedback working under user-level local differential privacy, where the privacy-utility trade-off is controlled by parameters ϵ and k, regulating the per-update privacy budget and the number of ϵ-LDP gradient updates sent by each user, respectively.
(2) We experimentally demonstrate the effectiveness of our proposed framework by analysing the system's performance as a function of various communication and privacy budgets, as well as item and user set cardinalities. On the MovieLens dataset we show that the system achieves HR@10 up to 0.68 on 50k users with 5k items, and even on the full dataset we show that it is possible to achieve reasonable utility with HR@10>0.5, without compromising user privacy.
(3) We further examine the communication costs associated with running the recommendation protocol, and present an analysis of its formal privacy guarantees. We further compare our system against several non-private and privacy-preserving benchmarks, showing that we can provide acceptable levels of utility, while offering stronger privacy guarantees.

1.2 Paper organization.

The rest of this paper is organized as follows. Section 2 introduces the problem and notation of collaborative filtering algorithms for implicit feedback, and recalls the definition of local differential privacy. Section 3 gives an overview of the proposed system. Section 4 describes our experimental work and reports the results. In Section 5 we further discuss the proposed system by taking into consideration privacy and communication costs, and by comparing our solution with competing methods. Finally, Section 6 presents an overview of related work before concluding with Section 7.

2 BACKGROUND

2.1 Collaborative Filtering with Implicit Feedback

Collaborative filtering is one of the most popularly used approaches in recommender systems. The goal of collaborative filtering is to model user preferences over a set of items by leveraging the "wisdom of the crowds". User behaviour is described by a set of *interactions* r_{ui}, where u is the user index and i is the index of the item being interacted with. For implicit data—our case, r_{ui} indicates user u has interacted with item i, or how many times that interaction has happened. Implicit feedback is more abundant, as it does not require an active effort by the user, but it is also less informative than its explicit counterpart, as there is no clear way to infer the qualitative judgment made by the user, i.e. simply knowing a film has been watched does not tell us anything about the user's *preference* for that film.

The first formulation of collaborative filtering for implicit feedback, relying on Singular Value Decomposition (SVD) of the user-item interaction matrix, is proposed in [13]. The authors adapt the matrix factorization methods already available for explicit data to the implicit case. First, they define *preference* p_{ui} as a binary indicator $\{0, 1\}$ of whether user i has a preference for item i, and assign a *confidence* c_{ui} score to this preference. They choose to model c_{ui} as the following:

$$c_{ui} = 1 + \alpha r_{ui}$$

where r_{ui} is the number of times user u has interacted with item i. This definition follows the intuition that an item that has been interacted with multiple times is more likely to be preferred by the user. Second, the cost function is modified from the explicit case to account for all possible u, i pairs, and to accommodate for the varying confidence levels in each observation. The implicit loss function is defined as the following:

$$L(\theta) = \sum_{u,i} c_{ui}(p_{ui} - \mathbf{x}_u^T \mathbf{y}_i)^2 + \lambda \left(\sum_u ||\mathbf{x}_u||^2 + \sum_i ||\mathbf{y}_i||^2 \right) \quad (1)$$

with λ being a joint regularization parameter. The partial derivatives with respect to $\mathbf{x}_u$ and $\mathbf{y}_i$ are given by:

$$\frac{\partial J}{\partial \mathbf{x}_u} = -2 \sum_i [c_{ui}(p_{ui} - \mathbf{x}_u^T \mathbf{y}_i)] \mathbf{y}_i + 2\lambda \mathbf{x}_u \quad (2)$$

$$\frac{\partial J}{\partial \mathbf{y}_i} = -2 \sum_u [c_{ui}(p_{ui} - \mathbf{x}_u^T \mathbf{y}_i)] \mathbf{x}_u + 2\lambda \mathbf{y}_i \quad (3)$$

2.2 Federated Collaborative Filtering

Federated Learning [15, 21] has been a fast growing field in machine learning research. It was proposed originally as a way to train a central model on privacy-sensitive data distributed across users' devices. In the federated learning paradigm, the user's data never has to leave the client. Instead, clients train a local model on their private data and share model updates with the server. These updates are then aggregated (typically by averaging) and a global model update is performed. Finally, the updated model is sent back to each client and the process is repeated, until convergence or until satisfying performance is achieved.

In a recently proposed method for federated collaborative filtering [1], the authors distribute the implicit matrix factorization problem introduced in the previous section between a server and multiple clients. At the very beginning, the server randomly initializes an item embeddings matrix V and clients initialize their own user embeddings $\mathbf{x}_u$ locally. The item matrix is then shipped to the clients, who locally compute an updated user vector in closed form

$$\mathbf{x}_u^* = (VC^u V^T + \lambda I)^{-1} VC^u \mathbf{p}(u) \quad (4)$$

Where C^u is an $M \times M$ diagonal matrix with $C^u_{ii} = c_{ui}$ and $\mathbf{p}(u)$ is a vector that contains the user preferences, and a client-based gradient update for the item matrix

$$f(u, i) = [c_{ui}(p_{ui} - \mathbf{x}_u^T \mathbf{v}_i)] \mathbf{x}_u \quad (5)$$

Gradient updates from all clients are aggregated together at the server and they are used to update the central item matrix with the following

$$\frac{\partial J}{\partial \mathbf{v}_i} = -2\sum_u f(u,i) + 2\lambda \mathbf{v}_i \tag{6}$$

$$\mathbf{v}_i = \mathbf{v}_i - \gamma \frac{\partial J}{\partial \mathbf{v}_i} \tag{7}$$

The updated item matrix is then sent down to all the clients and the federated process repeats.

2.3 Local Differential Privacy

Differential privacy [6, 7] has become the gold standard for strong privacy protection. It provides a formal guarantee that a model's results are negligibly affected by the participation or less of any *single* individual. Differential privacy was originally formulated in a *central* setting: a trusted aggregator collects raw user data and injects controlled noise either in the query inputs, outputs, or both. *Local* differential privacy, on the other hand, is a particular case of the differential privacy framework where the aggregator is not trusted and the noise injection is done directly by the client. Since no trust in the aggregator is required, the local version of DP can provide much stronger privacy protection. Companies like Google and Apple have applied the local model of differential privacy to their privacy-preserving data analytics protocols [8, 30, 35] and more recently to federated learning [2]. In order to define local differential privacy, we first define a local randomizer:

Definition 1. [7] A randomized algorithm $\Phi : \mathcal{U} \rightarrow \mathcal{Y}$ is an ϵ-local randomizer, where $\epsilon > 0$, if for all input pairs x and x' in D, and any output Y of Φ, we have

$$\Pr[\Phi(x) = Y] \leq e^{\epsilon} \cdot \Pr[\Phi(x') = Y],$$

Where x and x' belong to universe $\mathcal{U}$ of user data. Now that we have defined a local randomizer, we can proceed to give the definition of local differential privacy:

Definition 2. [7, 30] Let $\Phi : \mathcal{U}^n \rightarrow \mathcal{Z}$ be a randomized algorithm mapping a dataset with n records to some range $\mathcal{Z}$. Algorithm Φ is ϵ-local differentially private if it can be written as $\Phi(d^{(1)}, ..., d^{(n)}) = f(\Phi_1(d^{(1)}), ..., \Phi_n(d^{(n)}))$, where each $\Phi_i : \mathcal{U} \rightarrow \mathcal{Y}$ is an ϵ-local randomizer, and $f : \mathcal{Y}^n \rightarrow \mathcal{Z}$ is some post-processing function of the privatized records $\Phi_1(d^{(1)}), ..., \Phi_n(d^{(n)})$.

The privacy budget ϵ controls the trade-off between utility and privacy: when $\epsilon = 0$, we have perfect privacy and no utility, while for $\epsilon = \infty$ we would have no privacy and perfect utility.

3 SYSTEM

3.1 System Overview

We propose a novel framework to solve the recommendation task under the constraints set out in Section I by integrating federated recommendation [1, 18] and local differentially private protection mechanisms [4, 24]. Figure 1 illustrates the three components of our proposed system, which we describe in detail below.

3.1.1 Client with LDP Module. Initially, each client randomly initializes an F-dimensional local user embedding $\mathbf{x}_u$. This user embedding *never* leaves the client. Each participant receives user-agnostic item matrix V from the server, signalling the start of a federated epoch. The local user embedding and the item matrix are used to compute (1) an updated user embedding $\mathbf{x}_u^*$, which is, again, kept strictly private, and (2) a gradient update ∇V_u for item matrix V. Item-matrix gradient update ∇V_u is passed to the *LDP Module*, where we pick k factors uniformly at random and to each apply our privatization mechanism with per-report privacy budget ϵ. These k ϵ-LDP reports are then sent to a proxy network. On arrival of the new item matrix V', a new federated epoch starts and the above process repeats.

Once the training is finished, the user embedding $\mathbf{x}_u$ can be used in conjunction with the latest item matrix V to compute on-device the confidence that the user will interact with item i by taking the dot product between the two relevant embeddings $\langle \mathbf{x}_u, \mathbf{v}_i \rangle$.

Input. Item matrix V, per-update privacy budget ϵ, number of updates k.

Output. k ϵ-LDP gradient reports.

3.1.2 Proxy Network. The proxy network sits between the server and the client. Once the messages containing the k ϵ-LDP reports reach the proxy network, they are stripped of their metadata (such as IP address), split into the single reports, shuffled with the reports from other users' messages to break any existing timing patterns, and forwarded to the server. The proxy network takes care of breaking linkability between the streams of k reports coming from each client at each epoch. This greatly reduces the user fingerprinting surface and prevents the recommender from building any longitudinal representation of the user, both within and across epochs.

Input. k ϵ-LDP gradient reports from each of N users.

Output. $N \times k$ unlinkable ϵ-LDP gradient reports.

3.1.3 Server. The server randomly initializes an $M \times F$ item matrix, which constitutes the global part of the shared model. This is shipped to all the clients to initiate the federated learning process in epoch 0. At each epoch, once enough ϵ-LDP gradient reports reach the server, they are aggregated together to reconstruct a global item matrix update ∇V. This is used to compute an updated item matrix V', which is then shipped to each client to initiate the next federated learning epoch.

Input. $N \times k$ unlinkable gradient reports.

Output. Updated item matrix V'.

3.2 Proposed Method

For our study, we adopt the matrix factorization-based federated collaborative filtering framework as proposed in [1]. At the start of each learning epoch, each participant in the recommendation network receives the most recent item matrix V from the server. Each client computes an updated F-dimensional user embedding $\mathbf{x}_u^*$ using (4), and an MF-dimensional item matrix gradient update ∇V_u using (7). To privatize the gradient updates before shipping them to the server we adapt to our recommendation task the randomized binary response mechanism proposed in [24]. This method allows computing aggregates of users' gradients while satisfying local differential privacy. In particular, the proposed method takes in a numerical, multidimensional gradient matrix, randomly selects one item and one factor from it and encodes the selected value in the transmission frequency of two opposite global constants $\{B, -B\}$,

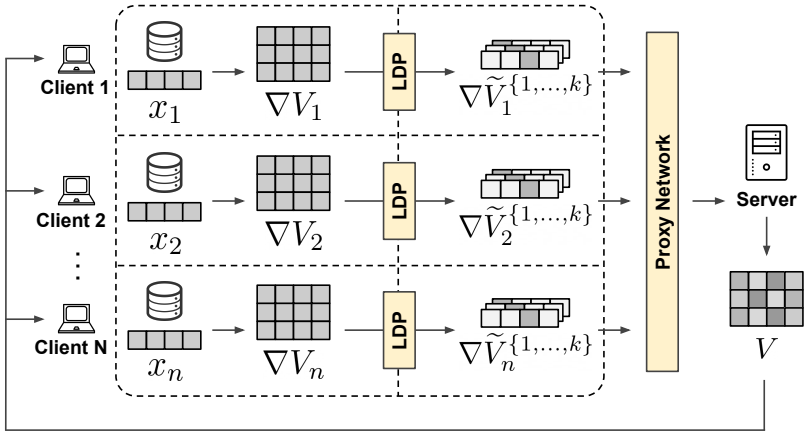

Figure 1: The proposed system design of privacy-preserving federated collaborative filtering with implicit feedback. The user vector x never leaves the device.

where B is defined as follows

$$B = \frac{e^\epsilon + 1}{e^\epsilon - 1} \cdot MF \tag{8}$$

with B only depending on the input data dimensionality $M \cdot F$ (same for all users) and per-update privacy budget ϵ (shared between all users). Algorithm (1) illustrates the adopted perturbation mechanism.

Algorithm 1: Gradient LDP perturbation mechanism, adapted from [24]

Input: Item gradient matrix ∇V_u

Initialize $\nabla \tilde{V}_u = 0^{M \times F}$

Sample item i uniformly at random

Project ∇V_u^i onto $[-1, 1]$

Sample factor f uniformly at random

Sample a Bernoulli variable β such that

$$Pr[\beta = 1] = \frac{\nabla V_u^{i,f} \cdot (e^\epsilon - 1) + e^\epsilon + 1}{2e^\epsilon + 2}$$

if $\beta = 1$ **then**

  $\nabla \tilde{V}_u^{i,f} = \frac{e^\epsilon + 1}{e^\epsilon - 1} \cdot MF$

else

  $\nabla \tilde{V}_u^{i,f} = -\frac{e^\epsilon + 1}{e^\epsilon - 1} \cdot MF$

end

return $\nabla \tilde{V}_u$

The output of this mechanism $\nabla \tilde{V}$ can be represented by a tuple (i, j) where i is the index of the selected gradient factor, and j is either 0 or 1, for whether B or $-B$ has been selected. This message is sent to the recommender, where it's aggregated with those from other users to form a global item matrix gradient update. More in line with the conventional federated learning literature and differing from [1], we average over the collected gradients.

$$\nabla \tilde{V} = \frac{1}{N} \sum_u^N \nabla \tilde{V}_u \tag{9}$$

Nguyên et al. [24] show that the mean of all users' raw values can be estimated by taking the average over the perturbed values, which in our adaptation translates to (9) being an unbiased estimator of ∇V. However, from the experiments run on this setup we observe that our federated protocol takes long to learn and ultimately performs poorly. The information exchanged with the server at each epoch is extremely sparse and very little learning can be extrapolated from it. More specifically, each user only contributes to only as much as $\frac{1}{MF}$ of the gradient matrix, with reports being further randomized. This ultimately leads to the computed aggregates being very poor estimates of the target values, even with a large number of users participating in the protocol.

In order to improve the accuracy of our system, we increase the amount of information shared by each user at each epoch by setting the number of randomly elected factors to k. Each gradient factor gets perturbed separately resulting in k tuples (i, j) being produced by each client and subsequently being shared with the server. This procedure, however, increases the privacy budget by a factor of k from the composition theorem [7]. If with $k = 1$ our system satisfied ϵ-LDP, now it satisfies $k\epsilon$-LDP, where k is the number of tuples exchanged by each user with the server. The k ϵ-LDP reports are sent to the proxy network, which then forwards them to the server where the global item matrix gradient update is estimated as (9). The full learning protocol is shown in Algorithm 2.

4 EXPERIMENTS

4.1 Data

Experiments are run on the MovieLens [10] dataset. The data consists of ratings from 138,493 users on 27,278 movies. Ratings range from 0.5 to 5 in 0.5 increments. To convert the 20M interactions into implicit feedback, we transform each rating to a binary value,

Algorithm 2: Local differentially private federated recommender system

Initialize X, V
for $t \in \{0, 1, 2...T\}$ **do**
  for $u \in \{0, 1, 2...N\}$ **do**
    Derive $\mathbf{x}_u^*$ with (4) ; ▷ ∀ clients
    Compute ∇V_u with (5)
    for $i \in 0, 1, 2...K$ **do**
      $\nabla \tilde{V}_u^{\{i\}} = \text{Algorithm1}(\nabla V_u, \epsilon)$
    end
  end
  Collect $\nabla \tilde{V}_u^{\{1,...,k\}}$ from all clients ; ▷ server
  $\nabla \tilde{V} = \frac{1}{N} \sum_u^N \nabla \tilde{V}_u$
  $V = V - \gamma\{\nabla \tilde{V} + 2\lambda V\}$
end
return X, V

with the following rule:

$$r_{ui}^{\text{imp}} = \begin{cases} 1, & \text{if } r_{ui}^{\text{exp}} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

In short, r_{ui} is 1 if user u has watched item i, 0 otherwise. Analogous to [9], we retain only users and items with at least 60 interactions. Our final dataset is 97.7% sparse with 17,386,309 interactions, from 75,040 users on 9,781 films. To further study the impact of user and item set cardinality on system performance we sample nine subsets from the full data set, with item set cardinalities *small* (1,000), *medium* (5,000), *large* (9,781) and user set cardinalities *small* (1,000), *medium* (10,000), *large* (50,000).

4.2 Evaluation

For evaluation we adopt the commonly used leave-one-out technique [9, 11]. For each user we randomly sample one interaction and use it as the test item. We cross-validate the model by running multiple random leave-one-out splits and presenting the mean results. To measure the performance of our recommender system, we adopt the popular *Hit Ratio* (HR@K) metric, where K is the number of recommended items. HR@K measures the probability of the recommender ranking the left-out test item in its top-K recommendations. To establish comparability between data from different items spaces, we follow recent literature [9, 11] by sampling 99 items that the user hasn't interacted with, appending the test item, and ranking the resulting subset.

4.3 Parameters

As parameters we chose $f = 5$ factors, a regularization term of $\lambda = 10^{-6}$, and a learning rate of $\gamma = 10^{-3}$, though values may vary slightly between experiments. The parameters were optimised with grid search over multiple cross-validation runs. Each epoch of training performs 20 gradient descent steps to update the global item matrix. We consider different per-update privacy budgets and numbers of updates, ϵ and k, respectively.

4.4 Results

In the following we outline the results of our experiments. We first study the privacy-utility trade-off as determined by ϵ and k. We then look at learning curves, as well as the impact of item set cardinality on utility. We end by comparing our method against various private and non-private benchmarks.

4.4.1 Privacy-utility trade-off. Figure 2 shows the system utility as measured by HR@10 for the small item set, all three user set sizes and varying per-update privacy budget ϵ and number of updates k. As expected, utility is on par with the random baseline for $k = 1$ across all user set sizes, as the signal sent to the recommender is insufficient to guide any learning. However, with increasing user set size, as well as per-report privacy budget and number of updates, we observe a consistent and significant increase in utility—in some cases reaching HR@10 of 0.7 and higher, while maintaining strong privacy—demonstrating the efficacy of the proposed system. When analysing the rate of improvement, it is apparent that increasing the number of updates k and increasing the number of users, both yield diminishing returns. E.g. we observe a ≈150% increase in utility from small to medium user set size and only a ≈50% increase from medium to large. Similarly, increasing k from 50 to 100 results in a 10% utility improvement over merely 5% when increasing k from 100 to 250. It should be noted that increasing the number of updates k, unlike increasing user set size, has a direct impact on communication as well as privacy costs, so it should always be justified in terms of utility gain.

4.4.2 Learning curves. Figure 3 shows HR@10 against the number of epochs on the validation set for a small item set with medium and large user set sizes and two different communication budgets of k=50 and k=100. The privacy budget is fixed at ϵ=2.5. Unsurprisingly, we can observe that the number of exchanged updates k has significant impact on the rate of learning as a higher quality signal reaches the recommender at each epoch. Reasonable utility of HR@10>0.5 can already be achieved after only 5-10 epochs, which is important for producing quality recommendations quickly, as well as keeping apace with user interest drift. Finally, user set size seems to affect only asymptotic performance, with larger user sets showing higher performance at convergence, corroborating findings from Figure 2.

4.4.3 Impact of item set cardinality on performance. We extend our discussion to different item set sizes and in Figure 4 report HR@10 for various combinations of user and item set sizes. Users only report a small subset of their local gradient. As the number of items increases by ΔM, the local gradient dimensionality increases by $\Delta M \times F$, where F is the number of factors. As a consequence, while maintaining the number of updates k fixed, the signal sent to the recommender in each epoch becomes sparser and loses in quality, thus negatively impacting the overall performance of the system. Empirically, we observe that utility decreases with increasing item set size. The effect is more pronounced for smaller user set sizes, while for the large user set size, utility degrades slower, showing acceptable performance (HR@10>0.5) across all item set sizes.

4.4.4 Benchmark against private and non-private benchmarks. The final experiment aims to compare the proposed system against

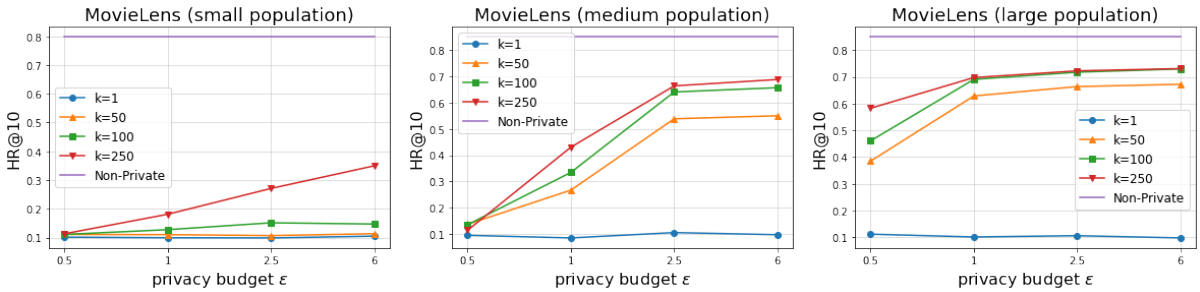

Figure 2: HR@10 performance with varying privacy budget ϵ and *small, medium* and *large* user set sizes. Number of epochs has been fixed to 20 using the *small* item set.

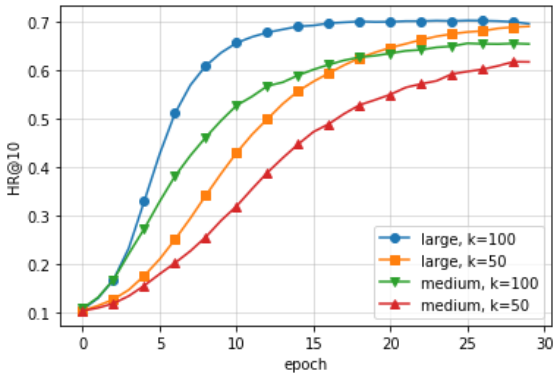

Figure 3: Learning curves with HR@10 validation set performance across 30 epochs, for varying k and user set sizes.

Users/Items	1K	5K	10K
1,000	0.1160	0.1010	0.1090
10,000	0.6325	0.1988	0.1216
50,000	**0.6972**	0.6801	0.5654

Figure 4: HR@10 performance for datasets of varying user set and item set cardinalities. Number of epochs fixed at 20, ϵ to 2.5 and number of updates $k = 100$.

various private and non-private benchmarks on the full dataset with number of updates k=100 (*FMF-LDP-100*) and k=250 (*FMF-LDP-250*) and fixed privacy budget of ϵ=2.5. We chose a random baseline model to establish the lower bound and traditional non-private matrix factorizaion (*MF-NP*) as a natural upper bound. We further compare our system with Gao et al. [9] *DPLCF* which, to the best of our knowledge, is the most comparable method in existing literature. We replicate their validation split by using only the latest interaction as the test item for each user. Models are trained over 20 epochs and the utility is measured as top-K hit rate for $K = \{2, 5, 10\}$. The results are shown in Figure 5. Our system clearly out-performs the random baseline with HR@10 of 0.5 and higher, corresponding to 5× improvement, showing that it is possible to provide reasonable utility without compromising user privacy. As expected, the system does not match performance of its non-private

Method	HR@2	HR@5	HR@10
MF-NP	0.4310	0.6700	0.8179
DPLCF [9]	0.4203	0.5969	0.7000
FMF-LDP-100	0.2163	0.3755	0.5131
FMF-LDP-250	**0.2315**	**0.3980**	**0.5384**
Random	0.0200	0.0500	0.1000

Figure 5: HR@K for various benchmarks on the latest split of the full dataset after at most 20 epochs of training and with $\epsilon = 2.5$, where applicable.

equivalent. On the full dataset the system also falls short of Gao et al. [9], however we argue that our method is in fact a competitive alternative on data with smaller item set cardinality than the full MovieLense dataset as shown in Figure 2. In section 5.1 we go on to show that our intuitive and formal privacy guarantees are significantly stronger.

5 DISCUSSION

In the following, we will discuss the privacy guarantees of our proposed system and show that it provides stronger privacy compared to similar methods. We will end with a practical analysis of the communication cost of running the protocol.

5.1 Privacy Analysis

This section analyzes how our system ensures user-level local differential privacy. We will first show that individual gradient updates satisfy event-level ϵ-local differential privacy, yielding $k\epsilon$-local differential privacy at user-level. Then we will argue how the proxy network can further safeguard privacy by establishing unlinkability of subsequent and concurrent gradient updates.

Theorem 3. *Algorithm 1 satisfies ϵ-local differential privacy.*

Proof. To show that Algorithm 1 is ϵ-local differentially private, for all inputs $\nabla V_q, \nabla V_q'$ and all outputs v_q, we have

$$\frac{P(\nabla\tilde{V}_q = v_q|\nabla V_q)}{P(\nabla\tilde{V}_q = v_q|\nabla V_q')} \leq \max_{v_q} \frac{P(\nabla\tilde{V}_q = v_q|\nabla V_q)}{P(\nabla\tilde{V}_q = v_q|\nabla V_q')} = \frac{\max_{v_q} P(\nabla\tilde{V}_q = v_q|\nabla V_q)}{\min_{v_q} P(\nabla\tilde{V}_q = v_q|\nabla V_q')}$$

where $\nabla\tilde{V}_q$, as per Algorithm 1, is non-zero only at index j. Further assuming that the randomly chosen constant is positive, i.e. $\beta = 1$, we have

$$\frac{\max_{v_q^j} P(\beta = 1|\nabla V_q^j)}{\min_{v_q^j} P(\beta = 1|\nabla V_q^{j,\prime})} = \frac{\max\limits_{\nabla V_q^j \in [-1,1]} \nabla V_q^j (e^\epsilon - 1) + e^\epsilon + 1}{\min\limits_{\nabla V_q^{j,\prime} \in [-1,1]} \nabla V_q^{j,\prime} (e^\epsilon - 1) + e^\epsilon + 1} = e^\epsilon$$

The same holds if the randomly chosen constant is negative, i.e. ($\beta = 0$). □

Having shown that individual gradient updates are ϵ-local differentially private at event-level, we can easily see that at user-level we have $k\epsilon$-local differential privacy for k updates via the composition theorem. This guarantee extends by Definition 2 to the aggregated updates across clients in Algorithm 2, showing that the system ensures $k\epsilon$-local differential privacy at user level.

The selected range of per-update ϵ in our experiments is common to related work in industry, such as [30]. It is also worth noting that the choice of ϵ needs to be calibrated to the type of data being protected. The same privacy budget applied to different types of data such as raw feedback, model parameters or model parameter updates, implies profoundly different privacy guarantees.

We now consider the privacy advantages of using a proxy network. First, the LDP-privatized reports are anonymized by the proxy network. This is done by removing metadata and shuffling the reports with reports from other users before forwarding them to the server. The server is thus unable to link together multiple reports from any individual client. This significantly reduces the fingerprinting surface on each user's gradient collection and prevents the recommender from building longitudinal profiles of the user, both within, as well as across epochs. Finally, Erlingsson et al. [36] show that shuffling anonymized reports can significantly amplify privacy guarantees when viewed from the central model.

We conclude by illustrating how relatively little data the client has to send to the recommender. Assuming a round of 20 epochs and with $k = 100$ updates each, a single user contributes to at most $20 \times 100 = 2,000$ factors of the shared item gradient matrix. Using the full-size MovieLens as an example and setting 5 as the number of factors, a single user only sends at most 4% of the entire item gradient matrix with $M \times D \approx 50,000$.

5.2 Comparison with Similar Methods

Gao et al. [9] also focus on the problem of privacy-preserving top-n recommendation with implicit feedback. In their proposed method, each user perturbs their interaction data before sending it to the recommender, which in return estimates item-to-item similarity based on the perturbed feedback. The learned item-matrix is then disseminated back to the users for local on-device recommendations based on past interaction history using kNN. Although we share similar motivations, their method differs significantly from ours.

As shown in Section 4, our framework's performance for $k = 250$ is 22% below the performance of the privacy-preserving system proposed in [9]. However, our framework can still provide reasonably strong performance HR@10=0.5384, while providing significantly stronger privacy protection to the user. Gao et al. [9] privatization algorithm is applied on interaction data (i.e. how many times a film has been watched) and satisfies user-level $M\epsilon'$-differential privacy, where M is the item set size. Whereas, our privatization algorithm is applied on gradient data and satisfies $k\epsilon$-differential privacy, where k is the number of updates. Since the two privatization mechanism act on different types of data, it is hard, if not impossible, to formally compare the respective budgets and guarantees. Intuitively, however, gradient data can be considered less privacy-sensitive than raw interaction data, meaning that, for a shared privacy budget ϵ, a privacy guarantee on gradient data is stronger than one on interaction data. Even in the scenario where we assume these two privacy budgets to be comparable, since by design $k \ll M$, we have that our framework's user-level privacy budget is significantly smaller than the one of the competing method.

5.3 Communication Cost

Downstream, at each epoch of federated training each client receives an $M \times F$ real-valued item matrix V from the server, where F we consider fixed at 5. Depending on the size of the chosen item set, this message will have different sizes: in the best case, where number of items is 1,000, we have that V weighs about 20 KB; whereas, when using the full 9,781 item set we have that V weighs about 0.2 MB. Assuming a maximum of 20 epochs of training and that the user participates to all of them, the total worst-case download cost for a single run of our federated system is 4 MB. Upstream, each client transmits k tuples (i, j), where i and j are respectively one 4-byte integer and one-bit boolean. At each epoch, each client uploads about $4k$ bytes, where k is the number of tuples. Assuming $k = 100$, which is the most common communication budget in our analysis, the uploaded data per epoch by each user would amount to 0.4 KB. For a total of 20 epochs, the total upload cost is 8 KB.

6 RELATED WORK

6.1 Privacy-Preserving Recommendation

In this review, we only consider work that treats the recommender as a potential attacker. We also define three levels of user data granularity: raw, model weights, gradient updates. Sharing raw data has the greatest privacy risk, while sharing gradient updates the lowest. Under the defined setting, we identify two main categories of methods.

6.1.1 Data Collection. In this category, a protection mechanism is adopted to privatize user *raw* data before sharing it with the recommender. These methods rely on a central model and data collection, however obfuscated, happens at *raw* level. Polat et al. [25] inject Gaussian noise in the reported ratings and use SVD-based collaborative filtering to produce privacy-preserving recommendations, however the authors fail to provide any formal privacy

guarantees for their method. Li et al. [17] study the problem of point-of-interest recommendation and propose an ad-hoc solution that involves transforming the raw trajectories into bipartite graphs before injecting carefully calibrated noise to meet ϵ-differential privacy guarantees. Shen and Jin [26, 27] propose privacy-preserving recommendation systems where user ratings are perturbed locally with provable privacy guarantees before being submitted to the recommender. More closely related to our specific problem setting, Gao et al. [9] adopt a differentially private protection mechanism based on randomized response to obfuscate private *raw* level interaction data before reporting it to the server. This data is used to estimate an item similarity matrix which is sent to users, who can then produce the recommendation results locally, further safeguarding their privacy.

All the above methods guarantee ϵ-differential privacy at *event-level*, that is they can only reduce the impact on the results of a certain interaction — a rating or click. Our framework, instead, strives to guarantee *user-level* differential privacy in order to protect the entirety of a user's interactions. It should be noted that an ϵ-DP guarantee at event-level naively translates, by the composition theorem [7], to a $d\epsilon$-DP guarantee at user-level, where d is the dimensionality of the transmitted data and ϵ the event-level privacy budget.

6.1.2 Distributed/Federated. In this category, learning is distributed between clients and server. Users' interaction data is kept strictly local and only gradient updates of a local model are shared with the server. As raw gradients have been shown to leak user information [12, 33], gradient data is passed through a privatization mechanism and only then sent to the server. Hua et al. [14] inject Gaussian noise into the local gradient updates to ensure that the transmitted gradient meets local differential privacy guarantees. However, since their method only shares the gradients for items that have been rated by the user, it indirectly gives away information about the user's interaction data. Shin et al. [28] also take a distributed approach to recommendation and are the first to adopt *user-level* local differential privacy as privacy requirement. They, however, have to deal with a large perturbation error caused by the stricter privacy requirements. They reduce the perturbation error by adopting dimensionality reduction through random projection, and sparse recovery, offering the first evidence of a recommender system working under local differential privacy guarantees. However, none of the above methods focus on recommendation for implicit feedback, and are only suitable for *explicit* (i.e. ratings) data. Recently, Duriakova et al. [5] proposed a decentralised matrix factorization protocol that enhances privacy by letting the user select the amount and type of information shared. However, the authors don't provide any formal differential privacy guarantees for their proposed method.

6.2 Local Differential Privacy for Analytics and ML

Because *local* DP both eliminates the need for a central trusted data curator and defuses the risks of inference attacks, it has recently garnered more attention than its *central* counterpart. Google's RAPPOR [8] proposes a combination of randomized response and Bloom filters to make possible crowd-sourcing simple statistics, such as frequently visited websites or OS process names, under LDP guarantees. Similarly, Apple propose Private Count Mean Sketch [30], a data collection protocol to identify popular emojis and popular health data types from users, while guaranteeing local differential privacy. More generally, a series of LDP mechanisms have been proposed to privatize tasks like frequency-estimation and heavy-hitters detection [4, 24, 32, 35].

Local differential privacy has also recently been applied to the task of federated learning [19, 29, 31] and, more broadly, to the task of distributed gradient descent [24, 28, 32]. To prevent private information leaking from raw gradient updates, each client feeds his local gradient updates to a privatization mechanism before shipping it to the server. This procedure, however, similarly to what we observed in our experiments, causes the estimation error of models learned on this privatized updates to increase with the dimensionality of the updates, raising utility issues even for very simple models. Several techniques have been proposed to increase the utility of these exchanges. Nguyên et al. [24] propose sampling a *single* dimension at random from the gradient vector, and submitting a perturbed version of the value to the server. The perturbed values are collected and the mean gradient of all users can be estimated on each gradient dimension. This technique succeeds in reducing the noise injected in the perturbed gradient update but, at the same time, induces extreme sparsity in communication between user and server. Shin et al. [28] build on the previous method and adopt dimensionality reduction through random projection in order to increase the utility of the gradient data and to lower the estimation error. However, their method relies on sparse recovery to retrieve the full-dimensional gradient matrix from the projection, which is only suitable when the unperturbed gradient data is sparse, as in the case of recommendation based on explicit data. With a similar objective, Liu et al. [19] propose a top-k dimensions selection mechanism to send higher "quality" information to the server, while avoiding the reconstruction loss caused by random projection. Finally, Sun et al. [29] propose splitting and shuffling gradient updates locally before sending them to the aggregator model to break linkability. However, the authors make the erroneous assumption that breaking linkability between the single gradient factors prevents the per-report privacy budgets from composing.

7 CONCLUSIONS

In this paper we presented what is to the best of our knowledge the first general privacy-preserving federated recommendations framework for implicit feedback under user-level local differential privacy guarantees. We empirically demonstrated the effectiveness of our federated recommendation framework across a variety of privacy and communication budgets, as well as item and user set cardinalities. On the MovieLens dataset, our system achieves up to HR@10=0.68 on 50,000 users with 5,000 items, and even on the full data set we show that our system can achieve good utility, with HR@10>0.5, without compromising user privacy. While likely not being an optimal solution for utility-first applications, where privacy is more of a concession than a requirement, we have shown how it is possible with a privacy-first approach to achieve reasonable performance on recommendation with implicit feedback.

REFERENCES

[1] Muhammad Ammad-ud din, Elena Ivannikova, Suleiman A. Khan, Were Oyomno, Qiang Fu, Kuan Eeik Tan, and Adrian Flanagan. 2019. *Federated Collaborative Filtering for Privacy-Preserving Personalized Recommendation System.* Technical Report. arXiv:1901.09888

[2] Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. 2018. *Protection Against Reconstruction and Its Applications in Private Federated Learning.* Technical Report. arXiv:1812.00984

[3] Joseph A. Calandrino, Ann Kilzer, Arvind Narayanan, Edward W. Felten, and Vitaly Shmatikov. 2011. "You Might Also Like:" Privacy Risks of Collaborative Filtering. *Proceedings - IEEE Symposium on Security and Privacy* (2011). https://doi.org/10.1109/SP.2011.40

[4] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. 2013. Local Privacy and Statistical Minimax Rates. *2013 51st Annual Allerton Conference on Communication, Control, and Computing, Allerton 2013* (2013). https://doi.org/10.1109/Allerton.2013.6736718

[5] Erika Duriakova, Elias Z. Tragos, Barry Smyth, Neil Hurley, Francisco J. Peña, Panagiotis Symeonidis, James Geraci, and Aonghus Lawlor. 2019. PDMFRec: A Decentralised Matrix Factorisation with Tunable User-Centric Privacy. *Proceedings of the 13th ACM Conference on Recommender Systems* (2019). https://doi.org/10.1145/3298689.3347035

[6] Cynthia Dwork. 2006. Differential Privacy. *Automata, Languages and Programming* (2006).

[7] Cynthia Dwork and Aaron Roth. 2013. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* (2013).

[8] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. *Proceedings of the ACM Conference on Computer and Communications Security.* https://doi.org/10.1145/2660267.2660348 arXiv:1407.6981

[9] Chen Gao, Chao Huang, Dongsheng Lin, Depeng Jin, and Yong Li. 2020. DPLCF: Differentially Private Local Collaborative Filtering. *SIGIR 2020 - Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2020). https://doi.org/10.1145/3397271.3401053

[10] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* (2015). https://doi.org/10.1145/2827872

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat Seng Chua. 2017. Neural Collaborative Filtering. *26th International World Wide Web Conference, WWW 2017* (2017). https://doi.org/10.1145/3038912.3052569 arXiv:1708.05031

[12] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. 2017. Deep Models under the GAN: Information Leakage from Collaborative Deep Learning. *Proceedings of the ACM Conference on Computer and Communications Security* (2017). https://doi.org/10.1145/3133956.3134012 arXiv:1702.07464

[13] Yifan Hu, Chris Volinsky, and Yehuda Koren. 2008. Collaborative Filtering for Implicit Feedback Datasets. *Proceedings - IEEE International Conference on Data Mining, ICDM* (2008). https://doi.org/10.1109/ICDM.2008.22

[14] Jingyu Hua, Chang Xia, and Sheng Zhong. 2015. Differentially Private Matrix Factorization. *IJCAI International Joint Conference on Artificial Intelligence* (2015).

[15] Jakub Konecný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated Learning: Strategies for Improving Communication Efficiency. *CoRR* abs/1610.05492 (2016). arXiv:1610.05492 http://arxiv.org/abs/1610.05492

[16] Shyong K. Lam, Dan Frankowski, and John Riedl. 2006. Do You Trust Your Recommendations? An Exploration of Security and Privacy Issues in Recommender Systems. *Emerging Trends in Information and Communication Security* (2006).

[17] C. Li, B. Palanisamy, and J. Joshi. 2017. Differentially Private Trajectory Analysis for Points-of-Interest Recommendation. *2017 IEEE International Congress on Big Data (BigData Congress)* (2017). https://doi.org/10.1109/BigDataCongress.2017.16

[18] Guanyu Lin, Feng Liang, Weike Pan, and Zhong Ming. 2020. FedRec: Federated Recommendation with Explicit Feedback. *IEEE Intelligent Systems* (2020). https://doi.org/10.1109/MIS.2020.3017205

[19] Ruixuan Liu, Yang Cao, Masatoshi Yoshikawa, and Hong Chen. 2020. FedSel: Federated SGD Under Local Differential Privacy with Top-k Dimension Selection. (2020). arXiv:2003.10637

[20] Pól Mac Aonghusa and Douglas J. Leith. 2016. Don't Let Google Know I'm Lonely. *ACM Transactions on Privacy and Security* (2016). https://doi.org/10.1145/2937754 arXiv:1504.08043

[21] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. 2016. Federated Learning of Deep Networks using Model Averaging. *CoRR* (2016). arXiv:1602.05629 http://arxiv.org/abs/1602.05629

[22] Frank McSherry and Ilya Mironov. 2009. Differentially Private Recommender Systems: Building Privacy into the Netflix Prize Contenders. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2009). https://doi.org/10.1145/1557019.1557090

[23] Arvind Narayanan and Vitaly Shmatikov. 2008. Robust De-anonymization of Large Sparse Datasets. *Proceedings - IEEE Symposium on Security and Privacy* (2008). https://doi.org/10.1109/SP.2008.33

[24] Thông T. Nguyên, Xiaokui Xiao, Yin Yang, Siu Cheung Hui, Hyejin Shin, and Junbum Shin. 2016. *Collecting and Analyzing Data from Smart Device Users with Local Differential Privacy.* Technical Report. arXiv:1606.05053 http://arxiv.org/abs/1606.05053

[25] Huseyin Polat and Wenliang Du. 2005. SVD-based Collaborative Filtering with Privacy. *Proceedings of the ACM Symposium on Applied Computing* (2005). https://doi.org/10.1145/1066677.1066860

[26] Y. Shen and H. Jin. 2014. Privacy-Preserving Personalized Recommendation: An Instance-Based Approach via Differential Privacy. *2014 IEEE International Conference on Data Mining* (2014). https://doi.org/10.1109/ICDM.2014.140

[27] Yilin Shen and Hongxia Jin. 2016. EpicRec: Towards Practical Differentially Private Framework for Personalized Recommendation. *Proceedings of the ACM Conference on Computer and Communications Security* (2016). https://doi.org/10.1145/2976749.2978316

[28] Hyejin Shin, Sungwook Kim, Junbum Shin, and Xiaokui Xiao. 2018. Privacy Enhanced Matrix Factorization for Recommendation with Local Differential Privacy. *IEEE Transactions on Knowledge and Data Engineering* (2018). https://doi.org/10.1109/TKDE.2018.2805356

[29] Lichao Sun, Xun Chen, Jianwei Qian, and Philip S. Yu. 2020. LDP-FL: Practical Private Aggregation in Federated Learning with Local Differential Privacy. arXiv:2007.15789

[30] Differential Privacy Team. 2017. *Learning with Privacy at Scale.* Technical Report. https://machinelearning.apple.com/2017/08/02/inverse-text-normal.html

[31] Stacey Truex, Ling Liu, Ka Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. 2020. LDP-Fed: Federated Learning with Local Differential Privacy. *EdgeSys 2020 - Proceedings of the 3rd ACM International Workshop on Edge Systems, Analytics and Networking, Part of EuroSys 2020* (2020). https://doi.org/10.1145/3378679.3394533 arXiv:2006.03637v1

[32] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. 2019. Collecting and Analyzing Multidimensional Data with Local Differential Privacy. (2019). arXiv:1907.00782v1

[33] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. 2018. Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning. *CoRR* (2018). arXiv:1812.00535 http://arxiv.org/abs/1812.00535

[34] Udi Weinsberg, Smriti Bhagat, Stratis Ioannidis, and Nina Taft. 2012. BlurMe: Inferring and Obfuscating User Gender Based on Ratings. *RecSys'12 - Proceedings of the 6th ACM Conference on Recommender Systems* (2012). https://doi.org/10.1145/2365952.2365989

[35] Wennan Zhu, Peter Kairouz, Brendan Mcmahan, Haicheng Sun, Wei Li, Rpi Google, and Google Google Google. 2020. Federated Heavy Hitters Discovery with Differential Privacy. (2020).

[36] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. 2020. Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. arXiv:1811.12469 [cs.LG]