EpicRec: Towards Practical Differentially Private Framework for Personalized Recommendation

Yilin Shen
Samsung Research America
665 Clyde Ave, Mountain View, CA 94043
yilin.shen@samsung.com

Hongxia Jin
Samsung Research America
665 Clyde Ave, Mountain View, CA 94043
hongxia.jin@samsung.com

ABSTRACT

Recommender systems typically require users' history data to provide a list of recommendations and such recommendations usually reside on the cloud/server. However, the release of such private data to the cloud has been shown to put users at risk. It is highly desirable to provide users high-quality personalized services while respecting their privacy.

In this paper, we develop the first *Enhanced Privacy-built-In Client for Personalized Recommendation (EpicRec)* system that performs the data perturbation on the client side to protect users' privacy. Our system needs *no* assumption of trusted server and *no* change on the recommendation algorithms on the server side; and needs minimum user interaction in their preferred manner, which makes our solution fit very well into real world practical use.

The design of EpicRec system incorporates three main modules: (1) *usable privacy control interface* that enables two user preferred privacy controls, overall and category-based controls, in the way they understand; (2) *user privacy level quantification* that automatically quantifies user privacy concern level from these user understandable inputs; (3) *lightweight data perturbation algorithm* that perturbs user private data with provable guarantees on both differential privacy and data utility.

Using large-scale real world datasets, we show that, for both overall and category-based privacy controls, EpicRec performs best with respect to both perturbation quality and personalized recommendation, with negligible computational overhead. Therefore, EpicRec enables two contradictory goals, privacy preservation and recommendation accuracy. We also implement a proof-of-concept EpicRec system to demonstrate a privacy-preserving personal computer for movie recommendation with web-based privacy controls. We believe EpicRec is an important step towards designing a practical system that enables companies to monetize on user data using high quality personalized services with strong provable privacy protection to gain user acceptance and adoption of their services.

CCS'16, October 24-28, 2016, Vienna, Austria
© 2016 ACM. ISBN 978-1-4503-4139-4/16/10...$15.00
DOI: http://dx.doi.org/10.1145/2976749.2978316

Keywords

Privacy-Preserving Recommendation; Differential Privacy; Privacy Paradox

1. INTRODUCTION

The last few decades have witnessed wide applications of recommender systems to provide personalized service to users, such as intelligent personal assistant and smart TV or other content recommendations. In fact, such personalized services become key business drivers for many companies. As one can understand, personalized service is based on user's data and oftentimes requires substantial user data in order to provide high-quality recommendation services.

However, many user concerns about recommender systems have been raised from privacy perspectives due to the release of users' private data. Consumer fears over privacy continue to escalate. Based on Pew Research, 68% consumers think that current laws are insufficient to protect their privacy and demand tighter privacy laws; and 86% of Internet users have taken proactive steps to remove or mask their digital footprints. Responding to increasing user privacy concerns, governments in US/EU are increasing and enforcing existing regulations. LG TV, as another example, was caught in lawsuit on illegally obtaining users' private data.

To resolve the tensions between business intelligence and user privacy, it is critically desirable to develop technologies that can preserve and control user data privacy and in the meanwhile still allow intelligence and personalization business. Without such technology enabler, future users will stop using services and companies will not be able to deploy services due to privacy law constraints and user concerns.

A majority of existing methods [5, 6, 9, 10, 11, 19] are developed based on the scenarios that recommender servers are trusted such as the Netflix movie recommendation system. One main reason for such assumption is that classic recommender system algorithms, such as Collaborative Filtering [25], require multiple users' data in order to perform personalized recommendation. It is easy to understand that a trusted server collects all users' data and can therefore perform such personalized recommendation. The most relevant privacy preserving approach is proposed by McSherry *et al.* [19], in which the server does the anonymization on user private data in which random noises are added into each step of aggregates in recommendation algorithm. All these methods attempted to protect user privacy when server releasing user data to third party applications and business partners.

Unfortunately, in such device-cloud based recommender systems, there are many other privacy attacks (as shown in

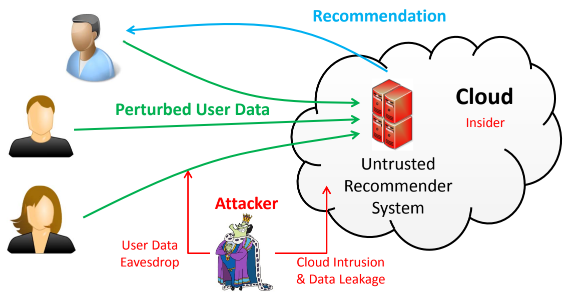
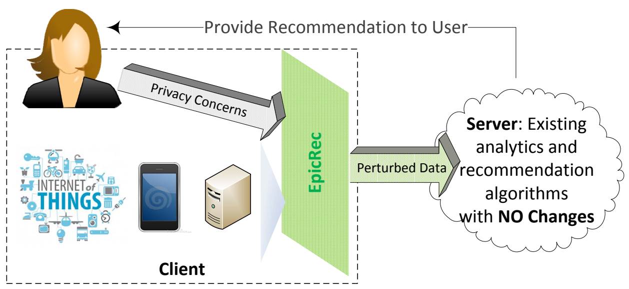


Figure 1: Attacking Model



Provide Recommendation to User
Privacy Concerns
EpicRec
Perturbed Data
Server: Existing analytics and recommendation algorithms with NO Changes
INTERNET of THINGS
Client

Figure 2: EpicRec System under Untrusted Server

Figure 1) that cannot be modeled and addressed in such a trusted server setting and require other types of protections. For example, when user data travels from device to cloud, attackers can eavesdrop the transmission channel and launch a "man-in-the-middle attack", therefore data may need to be encrypted during transmission. Malicious attackers can break into the cloud/server and steal user data. This demands many security measures to take to protect data on the cloud, such as encrypting data in storage. Moreover, server insiders may also leak user data to other parties.

As such, in this paper, we design a novel and practical privacy-built-in client under *untrusted* server settings, in which user data is perturbed and anonymized on their private devices before leaving their devices and users are given more peace of mind. As one can understand, data perturbation on device side under untrusted server settings poses extra challenges than that under trusted server settings because data perturbation has to be done without knowing other users' data.

There are some existing approaches developed under such untrusted server settings, including cryptography techniques [3, 21], differential privacy-based techniques [24], and randomization techniques [23]. Unfortunately, these approaches cannot be applied in practice due to various reasons such as computation cost, the need of an impractical trusted third party, lack of usability and so on.

In this paper, we propose the *first practical Enhanced Privacy-built-In Client for Personalized Recommendation (EpicRec)* system. As one can see in Figure 2, EpicRec, residing on the user's hub device (e.g., personal laptop, smartphone, etc.), collects user private data from a variety of user's devices and perturbs the private data based on user's privacy concerns. More importantly, the existence of EpicRec on user's device not only satisfies users' privacy needs but also requires no assumption of trusted server and no changes of recommendation algorithms, rendering EpicRec very practical. Our contributions are summarized as follows:

- We design the first privacy-preserving EpicRec framework on user client for personalized recommendation. EpicRec collects user private data from various devices, provides usable privacy control interfaces, quantifies user privacy control input and uses it to perturb user data. EpicRec enables user preferred overall privacy control (S-EpicRec) and category-based privacy control (M-EpicRec) to satisfy users' different needs; and in the meanwhile maintains low user cognitive load by minimizing the needs of user interactions.
- We design S-EpicRec and M-EpicRec systems respectively, both based on the state-of-the-art differential privacy and utility notions. We quantify the user privacy level by optimizing the utility based on the underlying data properties; and develop a light-weight and data perturbation algorithm to preserve the category aggregates with both privacy and utility theoretical guarantees, which significantly improves the existing approach [24] from both privacy and utility aspects.
- We conduct extensive experiments to evaluate the performance of EpicRec on large-scale real-world datasets. The results show that, from both privacy and utility perspectives, our proposed S-EpicRec and M-EpicRec systems consistently outperform other (pseudo) competitors that apply existing methods into some components of our EpicRec system. In addition, our approach takes less than 1.5 seconds on personal computers.
- We implement a proof-of-concept EpicRec system for personalized movie recommendation with web-based overall and category-based privacy concern controls.

The rest of paper is organized as follows. Section 2 discusses the related work. The background and architecture design of EpicRec system are presented in Section 3. Section 4 and 5 propose detailed design of S-EpicRec and M-EpicRec systems supporting different granularities of privacy controls. The experimental results and implementation of proof-of-concept system are presented in Section 6 and Section 7 respectively. Section 8 concludes the whole paper and discusses some future work.

2. RELATED WORK

Privacy-preserving Recommendation. Table 1 shows the comparison between our EpicRec system and existing approaches for personalized recommendation under untrusted server settings. The earliest work by Polat *et al.* [23] developed randomized mechanisms to perturb user private data before releasing to recommender systems. However, their method does not have provable privacy guarantees and was later identified that using clustering method on their perturbed data can still accurately infer users' original raw data with accuracy up to more than 70% [30] and make the privacy protection useless. In the meanwhile, cryptography-based approaches [21, 3] are proposed with privacy guarantees. However, these approaches require a trusted third-party (Cryptographic Service Provider (CSP)) and expensive private computation. Another class of orthogonal approaches [24] are based on the state-of-the-art differential privacy notion, with both privacy and utility guarantees. Unfortunately, in addition to the above limitations, all existing approaches largely ignore the usable privacy control

Table 1: Comparison between Privacy-Preserving Recommendation under Untrusted Server Settings

Approaches	No Change of Service Provider	No Need of Trusted Third Party	User Privacy Control	User-friendly Privacy Control Interface	Privacy Quantification	Utility Guarantee	Privacy Guarantee
Polat *et al.* [23]	✓	✓	Single	✗	✗	✗	✗
Nikolaenko *et al.* [21]	✗	✗	✗	✗	✗	✓	Cryptography
Canny *et al.* [3]	✗	✓	✗	✗	✗	✓	Cryptography
Xin *et al.* [27]	✗	✓	✗	✗	✗	✗	2nd Order Privacy
Shen *et al.* [24]	✓	✓	Single	✗	✗	✓	Differential Privacy
EpicRec	✓	✓	Single & Category-based	✓	✓	✓	Differential Privacy

such that users cannot provide their privacy concerns in a way they understand.

In addition, there are some other privacy-preserving recommendation approaches under trusted server settings [19] or some particular recommendation services [17]. Moreover, system developers, and policy makers recently have been coming up with solutions at different levels [13, 29].

Our proposed EpicRec system provides a comprehensive solution at all levels, towards a practical and usable privacy-preserving client for untrusted recommender system, with strong privacy and utility guarantees.

Differential Privacy. Differential privacy [7, 8] has become the de facto standard for privacy preserving data analytics. Dwork *et al.* [8] established the guideline to guarantee differential privacy for individual aggregate queries by calibrating the Laplace noise to each query based on the global sensitivity. Various works have adopted this definition for publishing histograms [28], search logs [15], mining data streams [4], and record linkage [1]. Later on, a probabilistic relaxation was proposed by Machanavajjhala *et al.* [18], called probabilistic differential privacy. This novel differential privacy notion allows the privacy preservation with high probability, thereby improve the flexibility of global sensitivity analysis. An alternative approach for noise mitigation was instance-based noise injection approaches by Nissim *et al.* [22]. This paper first introduced local sensitivity and its upper bound smooth sensitivity, which allows the injection of Admissible noise to ensure differential privacy. Unfortunately, all these approaches require the strict satisfaction of perturbed aggregates in the sanitized data and restrict their applications to only statistical data publishing. A recent work [24] was proposed to address the above constraints, by perturbing data to guarantee both differential privacy and recommendation quality. Our data perturbation module in EpicRec framework provides better privacy and utility than [24] from both theoretical and empirical perspectives.

3. EpicRec SYSTEM DESIGN

In this section, we present the framework design of our proposed *Enhanced Privacy-built-In Client for Recommendation* (EpicRec) system. *The goal of EpicRec system is three-fold: (1) enable user-friendly privacy concern control on their private data in a way they understand; (2) quantify user's privacy level input from layman and user-understandable language to quantified private budget for data perturbation; (3) conduct light-weight perturbation of user private data on their device such that the perturbed data can be released to existing recommender systems to provide user high-quality personalized recommendations.* Next, we first briefly discuss about background to motivate and guide system design. We then describe the overall architecture of EpicRec system and the details of each component.

3.1 Background & Motivation

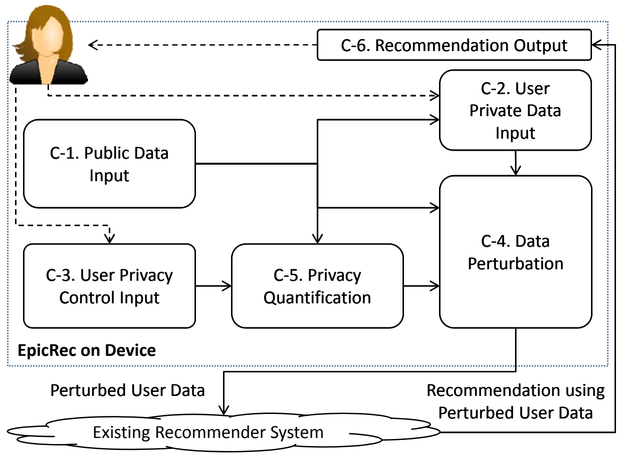
We first give a succinct overview of the key results from our two user studies (details are in separate papers to be submitted with preliminary results in [29, 26]), in order to motivate and guide the design of EpicRec system.

The first user study focuses on the research question: *Which information is considered as private by users?* Using a popular video recommender system as an example, we conducted an online user study by recruiting 161 participants through Amazon Mechanical Turk (MTurk) and studied 11 types of data collected by smart TV. The majority of the participants were male (66.5%) and White (75.8%). There are 77.0% of participants aged between 20 and 40. We created 11 types of data collected by smart TV including content, channel, time(watch, change, service), status(on/off, on duration, if using DVR), TV settings, clicked buttons and interacted services. The results show that most participants raised their most privacy concerns about their watching content history for either personalized program recommendation or targeted advertising purposes.

The second user study focuses on the research question: *How does level of control in a smart TV influence user's perceptions and behaviors?* We recruited 505 participants through MTurk and studied 15 different privacy control mechanisms with different levels and types of control. The majority of the participants were male (57.0%) and White (75.8%). The average age was 34.15 (range 19-72). We created 15 privacy control conditions by combining the following aspects: non-hierarchical and hierarchical controls on different content types (overall, category, maturity rating, watching time). The results show that the overall privacy control and finer-grained category-based privacy control are the best two control interfaces the participants selected among the 15 different designs. The participants rated them as the most useful in helping to make data disclosure decisions; the least privacy concerns; the most valuable in disclosing their information for personalized recommendations; and the most likely to use in the future.

3.2 EpicRec Architecture

Figure 3 shows the overall architecture of EpicRec system. Our focus is on user's device side where EpicRec system sits while the service provider remains unchanged. The goal of device-side EpicRec system is to perform data perturbation on user private data with user-specified privacy concern levels, such that the format of perturbed data remains the same and recommendation results remain accurate.



C-6. Recommendation Output
C-2. User Private Data Input
C-1. Public Data Input
C-4. Data Perturbation
C-3. User Privacy Control Input
C-5. Privacy Quantification
EpicRec on Device
Perturbed User Data
Recommendation using Perturbed User Data
Existing Recommender System

Figure 3: Architecture of EpicRec System

Motivated by the user study results in Section 3.1, our proposed EpicRec system provides users their most preferred overall and category-based privacy controls. The goal of data perturbation is to protect user concerned private history data. Specifically, EpicRec enables the protection of both individual records and categories of history data. Last but not least, EpicRec focuses on perturbing user's history data rather than rating data since users are more concerned about history data than rating data (based on our first user study) and history data can be easily obtained implicitly from user's client while it is infeasible to continuously request user interaction to rate each item in large amount of his history data.

As one can see in Figure 3, the dashed lines indicate the interactions between user and EpicRec. A user inputs his privacy levels using user privacy control input (C-3) and provides his private data on device to user private input (C-2). On the other hand, the solid lines indicate the interactions between different components. We next discuss the details of each component as well as the interactions between them:

C-1. Public Data Input: *obtain public knowledge associated with user private data from either inside or outside knowledge resources.* Specifically, C-1 collects two types of data from public resources: public data universe items and their associated public categories. The goal of this component is to preserve the quality of perturbed data without sacrificing any privacy breach that could be derived using public information.

C-2. User Private Data Input: *obtain user history data without extra user interaction,* such as location history data on iPhone; user's web browsing history based on the websites users clicked, or history of watched TV programs, movies on smart TV, etc.

C-3. User Privacy Control Input: *provide user interface to obtain user's privacy concern levels.* Motivated by our user study results discussed in Section 3.1, C-3 provides the following two granularities of privacy control interfaces:

Overall (Single-level) Privacy Control:
Provide users a single input of privacy concern level;

Category-based (Multiple-level) Privacy Control:
Provide users inputs of privacy concern levels for each category;

In each control, we use three privacy concern levels: *"No Release"* as releasing no information, *"Perturbed Release"* as releasing protected data, and *"All Release"* as releasing all private data (Note that "All Release" will release as much information as possible if conflict happens in category-based privacy control). The designs of EpicRec system with these two privacy controls are later presented in Section 4 and Section 5 respectively, referred to as S-EpicRec and M-EpicRec systems.

C-4. Data Perturbation: *perturb private user data from C-2, using public information from C-1 and the privacy parameters from C-5.* The perturbed data maintains the same format as user private data, such that it can be used by existing recommender algorithms. In addition, the perturbed data needs to meet two conflict goals, *privacy preservation* and *recommendation accuracy.* As such, this data perturbation module is associated with two corresponding notions: *privacy notion* and *utility notion.* Examples of privacy notions can be differential privacy, k-anonymity, information gain, etc. while examples of utility notions can be mean absolute error, root mean square error, TopK, etc.

C-5. Privacy Quantification: *quantify user specified privacy concern levels to mathematical privacy parameters to be used in data perturbation (C-4) component.* The examples of these quantified parameters based on different privacy notions can be as follows: (1) the privacy budget ϵ in differential privacy; (2) the value of k in k-anonymity privacy; (3) the threshold of information gain; etc.

C-6. Recommendation Output: *output recommendation results (e.g., overall recommendation, per-category recommendation) to users obtained from service provider using perturbed user data.*

4. DESIGN OF S-EpicRec: SINGLE-LEVEL PRIVACY CONTROL

In this section, we focus on the design of Single-level EpicRec (S-EpicRec) to enable overall privacy control. In the rest of this section, we first introduce our focus of privacy and utility notions, and some general notations. Then we present the detailed design of the main components (data perturbation (C-4) and privacy quantification (C-5) components) in S-EpicRec.

4.1 Privacy & Utility Notions

4.1.1 Privacy Notion

We consider using the state-of-the-art privacy notion, *Differential Privacy* [8], which not only provides strong privacy guarantee but also allows attackers to have unlimited background knowledge. Informally, an algorithm $\mathcal{A}$ is differentially private if the output is insensitive to any particular record in the dataset.

Definition 1 (ϵ-Differential Privacy). *Let $\epsilon > 0$ be a small constant. A randomized algorithm $\mathcal{A}$ is ϵ-differentially private if for all data sets D_1 and D_2 differing on at most one element, i.e., $d(D_1, D_2) = 1$, and all $\mathcal{S} \subseteq \mathsf{Range}(\mathcal{A})$,*

$$Pr[\mathcal{A}(D_1) \in \mathcal{S}] \leq \exp(\epsilon) Pr[\mathcal{A}(D_2) \in \mathcal{S}] \tag{4.1}$$

The probability is taken over the coin tosses of $\mathcal{A}$.

The parameter $\epsilon > 0$ is called *privacy budget*, which allows user to control the level of privacy. A smaller ϵ suggests more limit posed on the influence of an individual item, leading

Table 2: Notations

Symbol	Description
I	public item set of size n
C	public category set of size c
$\mathbf{C}$	public item-category correlation matrix of size $n \times c$
$\mathbf{d_r}$	user's private item vector of size n
$\mathbf{d_p}$	user's perturbed item vector of size n
PT	privacy concern level
ϵ	quantified privacy budget

to stronger privacy protection. More importantly, the application of differential privacy ensures perturbed data independent of any auxiliary knowledge [7] the adversary may obtain from public data in C-2.

4.1.2 Utility Notion

Recommender systems typically require *many* users' history data for providing each user a list of his/her personalized recommendations. However, when perturbing data on each user's device side under untrusted server settings, the device does not even know other users' data (no matter private or perturbed). Therefore, it is impractical to directly use the quality of recommendation results as a utility notion.

As such, we alternatively consider using similar utility in [24] to measure data category aggregates on each user's perturbed data as our utility notion. More specifically, we use expected *Mean Absolute Error (MAE)* between user's raw and perturbed category aggregates, which is shown later in experiment to be sufficient to guarantee recommendation accuracy even without knowing other users' data.

4.2 Notations

We define notations based on in each component:

C-1: Let I be public universe/set of items of size $|I| = n$. Public category set is defined as C of size $|C| = c$, in which each item is associated with a subset of categories represented by a public item-category correlation matrix $\mathbf{C}$ of size $n \times c$. An item i is associated with category j if and only if the entry c_{ij} in $\mathbf{C}$ is equal to 1.

C-2: User's raw private history is denoted as a vector $\mathbf{d_r}$ of size n. The i^{th} entry in $\mathbf{d_r}$ is either 1 or 0, meaning that item i does or does not belong to user's private history. Note that those items in private history but not in collected public set will be simply be considered no release.

C-3: User's privacy concern level, denoted as PT, belongs to one of the following three levels, {*"No Release", "Perturbed Release", "All Release"* }. When PT is selected as "No Release" or "All Release", the device simply releases no private or all private data to recommender system respectively.

C-4: User's perturbed data is denoted as a vector $\mathbf{d_p}$ of size n. The i^{th} entry in $\mathbf{d_p}$ is either 1 or 0, meaning that item i does or does not belong to user's perturbed data.

C-5: ϵ is the quantified privacy parameter (privacy budget in differential privacy) when PT is selected as *Perturbed Release.*

For simplicity and consistency, we denote the i^{th} entry in a vector $\mathbf{v}$ as $v(i)$ in the rest of paper. For reference, we list all notations in Table 2.

4.3 Design of Data Perturbation (C-4)

As described in Section 3.2, data perturbation component (C-4) generates perturbed user data to meet both privacy preservation and recommendation quality, specifically via the privacy and utility notions in Section 4.1. The rest of this subsection consists of problem definition, challenges, proposal algorithm and theoretical analysis.

4.3.1 Problem Definition

PROBLEM 1 (S-PERTURBATION PROBLEM). *Given a user's private item vector $\mathbf{d_r}$ associated with public item set I, a public item-category correlation matrix $\mathbf{C}$, privacy budget $\epsilon > 0$. The objective is to generate the user's perturbed item vector $\mathbf{d_p}$ such that* ***(1) (privacy goal)*** *the category aggregates (number of items belonging to each category) of perturbed data satisfy ϵ-differential privacy with the presence or absence of an individual item to defend against privacy leakage via public category information;* ***(2) (utility goal)*** *the quality of category aggregates on perturbed data is well maintained using metrics in Section 4.1.2. Specifically, the formal mathematical definition of Expected Mean Absolute Error (MAE) is defined as $\mathsf{E}\Big[\frac{1}{c}\sum_{j=1}^{c}|CR(j) - CP(j)|\Big]$, given user raw and perturbed category aggregates* $\mathbf{CR}$ *and* $\mathbf{CP}$.

Remarks: Our defined S-Perturbation problem targets on a stronger privacy guarantee (ϵ-differential privacy rather than (ϵ, δ)-differential privacy in [24]) and relaxes the objective (discard the maximization of difference between private and perturbed data in [24]), which is shown in later experiments that does not hurt the quality of data perturbation.

4.3.2 Challenges

Large Magnitude of Noises for Achieving ϵ Differential Privacy. One of the most widely used mechanisms to achieve ϵ-differential privacy is Laplace mechanism [8] (Theorem 1), which adds random noises to the numeric output of a query, in which the magnitude of noises follows Laplace distribution with variance $\frac{\Delta f}{\epsilon}$ where Δf represents the global sensitivity of query f (Definition 2).

DEFINITION 2 (GLOBAL SENSITIVITY [8]). *For a query $f : \mathcal{D} \to \mathbb{R}^k$, the global sensitivity Δf of f is*

$$\Delta f = \max_{d(D_1,D_2)=1} \|f(D_1) - f(D_2)\|_1 \quad (4.2)$$

for all neighboring datasets D_1, D_2, i.e., $d(D_1, D_2) = 1$.

THEOREM 1 (LAPLACE MECHANISM [8]). *For $f : \mathcal{D} \to \mathbb{R}^k$, a randomized algorithm $\mathcal{A}_f = f(D) + \mathsf{Lap}^k(\frac{\Delta f}{\epsilon})$ is ϵ-differentially private. (The Laplace distribution with parameter β, denoted $\mathsf{Lap}(\beta)$, has probability density function $\hbar(z) = \frac{1}{2\beta}\exp(-\frac{|z|}{\beta})$ and cumulative distribution function $\frac{1}{2}(1 + \mathsf{sgn}(z)(1 - \exp(-\frac{|z|}{\beta})))$.)*

Unfortunately, as an item usually belongs to many categories, the naive application of Laplace mechanism results in the significantly large noise magnitude and uselessness of perturbed data because of large global sensitivity.

Intractability of Generating Useful Perturbed Data. Even after the noise magnitude is determined, the data perturbation still remains intractable (NP-hard) when we need to guarantee the usefulness of perturbed data.

4.3.3 Proposed S-DPDP Approach

In this subsection, we propose a novel *Single-Level Differentially Private Data Perturbation (S-DPDP) Algorithm* to solve Problem 1. In general, S-DPDP algorithm consists of two phases to overcome the aforementioned two challenges: (1) Phase 1, noise calibration, focuses on selecting the magnitude (denoted as $z(j)$) for each category using public domain knowledge that determines injected $\mathsf{Lap}(z(j))$ noises for each category; (2) Phase 2, data sanitization, aims to generate the useful perturbed data based on the noisy category aggregates. Note that this phase will not lead to any privacy loss without the access to user private data. We next present the details of these two phases (Algorithm 1).

Phase 1: Noise calibration. The selection of noise magnitude is determined by optimizing the expected MAE defined in 4.3.1. Here we denote $\mathbf{z} = (z(1), \ldots, z(c))$ in which $z(j)$ is
the magnitude of Laplace noise for category j. Fc
ity, we also denote $\mathbf{z_I} = (\frac{1}{z(1)}, \ldots, \frac{1}{z(c)})$. Therefore
magnitude of Laplace noises on each category aggı
be determined via the following mathematical pro;
(4.3):

$$\begin{array}{ll} \textit{minimize} & \|\mathbf{z}\|_1 \\ \textit{subject to} & \mathbf{C}\mathbf{z_I} \leq \epsilon\mathbf{1}, \mathbf{z}, \mathbf{z_I} \geq \mathbf{0} \end{array}$$

The objective in (4.3) is to minimize expected N
injected Laplace noises onto category aggregates
noise on each category aggregate has $\mathsf{E}[|\mathsf{Lap}(z(j)$
and the injected noises are independent. The first
serves two purposes: First, it imposes the ϵ-differ
vacy guarantee as later shown in privacy analysis
2); Second, it captures the correlation between
from the public information that what categories
belongs to. The last two constraints ensure the noı
noise magnitude of $\mathbf{z}$ and $\mathbf{z_I}$.

As the formulation (4.3) is non-convex, we then
it into the convex programming (4.4) to obtain a
timal solution. Specifically, we regard both $\mathbf{z}$ and
variables. For the sake of clarification, we intrc
more variables $\mathbf{z_1} = \mathbf{z}, \mathbf{z_2} = \mathbf{z_I}$. Then, we add additional constraints $z_1(j)z_2(j) = 1$ for each category j to ensure their reciprocal relationship. Moreover, we further relax this constraint to $\mathbf{z_1}\mathbf{z_2}^T \geq \mathbf{I}$.

$$\begin{array}{ll} \textit{minimize} & \|\mathbf{z_1}\|_1 \\ \textit{subject to} & \mathbf{C}\mathbf{z_2} \leq \epsilon\mathbf{1}, \mathbf{z_1}\mathbf{z_2}^T \geq \mathbf{I}, \mathbf{z_1}, \mathbf{z_2} \geq \mathbf{0} \end{array} \tag{4.4}$$

In this phase, our data perturbation algorithm first solves the convex programming (4.4). Then, we set $\mathbf{z_I} = \mathbf{z_2}$ such that the first constraint in (4.4) is not violated; and set $\mathbf{z}$ by letting each entry $z(j)$ be the reciprocal of the j^{th} entry in $\mathbf{z_I}$. Thanks to the convexity property of (4.4), our optimized noise calibration algorithm is guaranteed to outperform the traditional Laplace mechanism.

EXAMPLE 1. *Figure 4 shows a running example that explains why our novel noise calibration approach (phase 1) can always outperform the existing Laplace mechanism in Theorem 1. In this tiny example, public set of items contains five items and their associated five categories. A check represents that an item belongs to this category (e.g., item 1 belongs to category 1,2,3).* ***The row in gray*** *shows our novel category based sensitivity obtained by solving* (4.4) *with* $\epsilon = 1$. *For a category associated with more items, the sensitivity of this category intends to be larger since there is higher probability that the aggregate of this category will be affected by adding or removing a single item. Therefore, the last column of last two rows shows a better MAE error using*

```
     Input : private user data $\mathbf{d_r}$, item-category matrix $\mathbf{C}$,
             privacy budget $\epsilon$
     Output: perturbed user's data $\mathbf{d_p}$
     // Phase 1: Noise calibration
   1 Solve mathematical programming (4.4);
   2 $\mathbf{z} \leftarrow$ reciprocal of each entry in $\mathbf{z_I}$;
   3 Sample noises from $\mathsf{Lap}(z(j))$ for each category $j$;
   4 Set $\mathbf{NA}$ with each entry
       $NA(j) = \sum_{i \in I} c_{ij} d_r(i) + \mathsf{Lap}(z(j))$;
     // Phase 2: Data sanitization
   5 Relax integral constraints in (4.5);
   6 Solve the relaxed (4.5) by replacing $\mathbf{d_p}$ with $\mathbf{d_p^r} \in [0,1]^n$;
   7 foreach each element $i$ in $\mathbf{d_p^r}$ do
   8     Randomly select a number $\xi$ between 0 and 1;
   9     $d_p(i) \leftarrow 1$ if $\xi \leq d_p^r(i)$ and $d_p(i) \leftarrow 0$ otherwise;
```

item/category	category 1	category 2	category 3	category 4	category 5	**MAE error when privacy budget $\epsilon = 1$**
item 1	✓	✓	✓			
item 2	✓		✓			
item 3	✓		✓	✓		
item 4	✓				✓	
item 5		✓		✓		
Sensitivity	3.61	2.36	3.34	2.36	1.38	2.61
	global sensitivity equals to 3 for all categories					3

Figure 4: Running Example of Noise Calibration

Phase 2: Data sanitization. This phase takes the above noise magnitude vector output $\mathbf{z}$ as input and generates the useful perturbed user data. We first quantify the usefulness of perturbed data by minimizing the error between the category aggregates on perturbed data and the noisy category aggregates. Specifically, we introduce two error vectors $\mathbf{l}, \mathbf{r}$ and consider the root mean square error (RMSE) as $\frac{1}{2}\|\mathbf{l}+\mathbf{r}\|_2^2$. Then, we formulate the following optimization formulation (4.5):

$$\begin{array}{ll} \textit{minimize} & \frac{1}{2}\|\mathbf{l}+\mathbf{r}\|_2^2 \\ \textit{subject to} & \mathbf{NA} - \mathbf{l} \leq \mathbf{C}^T\mathbf{d_p} \leq \mathbf{NA} + \mathbf{r}, \mathbf{d_p} \in \{0,1\}^n \end{array} \tag{4.5}$$

where $\mathbf{NA}$ is the noisy category aggregate vector in which each entry $NA(j) = \sum_{i \in I} c_{ij} d_r(i) + \mathsf{Lap}(z(j))$.

It is not hard to see that solving (4.5) is NP-hard by reducing it from Exact Cover problem (The proof is similar to that in [24] and omitted due to space limit). Therefore, our data perturbation algorithm solves the relaxed formulation of (4.5) by replacing $\mathbf{d_p}$ with $\mathbf{d_p^r}$ in which $\mathbf{d_p^r} \in [0,1]^n$. Then, we obtain $\mathbf{d_p}$ by rounding each entry $d_p^r(i)$ to 1 with probability $d_p^r(i)$.

4.3.4 Theoretical Analysis

We theoretically analyze privacy and utility, as well as time complexity.

Privacy Analysis. We show the differential privacy guarantee of S-DPDP algorithm:

THEOREM 2 (S-DPDP PRIVACY ANALYSIS). *S-DPDP algorithm enjoys ϵ-differential privacy.*

PROOF. We observe that there is no privacy loss in phase 2 of S-DPDP algorithm as it is considered as post-processing on differentially private category aggregates without the access of user private data. Since any post-processing of the answers cannot diminish this rigorous privacy guarantee according to Hey *et al.* [12], we only need to focus on analyzing the privacy guarantee in phase 1 of S-DPDP algorithm.

Let D_1, D_2 be neighboring datasets (i.e., $d(D_1, D_2) = 1$) and $f(D_i)$ be the category aggregates of user's private data D_i (w.r.t. $\mathbf{d_r}^i$). For any $\mathbf{r} = (r_1, \ldots, r_c) \in \mathsf{Range}(\textit{S-DPDP})$, we have the following analysis:

$$\begin{aligned}\frac{\Pr[\textit{S-DPDP}(D_1) = \mathbf{r}]}{\Pr[\textit{S-DPDP}(D_2) = \mathbf{r}]} &= \prod_{j\in C} \frac{\Pr[\textit{S-DPDP}(D_1)(j) = r(j)]}{\Pr[\textit{S-DPDP}(D_2)(j) = r(j)]}\\ \geq\ & \exp\Big(-\sum_{j\in C}\frac{1}{z(j)}|f_j(D_1) - f_j(D_2)|\Big)\\ \geq\ & \exp\Big(-\max_{d(D_1,D_2)=1}\sum_{j\in C}\frac{1}{z(j)}|f_j(D_1) - f_j(D_2)|\Big) \geq e^{-\epsilon}\end{aligned}$$

The first step holds due to the independently injected noises on each category aggregate; the second step is derived from the injected Laplace noises and triangle inequality; and the last step holds from the first constraint in (4.4), that is,

$$\max_{d(D_1,D_2)=1}\sum_{j\in C}\frac{1}{z(j)}|f_j(D_1) - f_j(D_2)| = \max_{i\in I}\sum_{j\in C}\frac{1}{z(j)}c_{ij} \leq \epsilon$$

The proof is complete. $\square$

Utility Analysis. We show the utility bound of MAE on category aggregates between raw and perturbed data:

THEOREM 3 (S-DPDP UTILITY ANALYSIS). *The expected MAE between raw and perturbed category aggregates (via S-DPDP algorithm) is upper bounded by* $2\|\mathbf{z}\|_1/c$, *where* $\mathbf{z}$ *is the optimal solution of* (4.4).

PROOF. Let $z(j), l_j, r_j$ be the j^{th} entry in vectors $\mathbf{z}, \mathbf{l}, \mathbf{r}$.

$$\begin{aligned}\mathsf{E}[c\cdot\mathsf{MAE}] &= \mathsf{E}\Big[\sum_{j=1}^{c}|CR(j) - CP(j)|\Big]\\ &\leq \mathsf{E}\Big[\sum_{j=1}^{c}|\mathsf{Lap}(z(j))|\Big] + \mathsf{E}\Big[\sum_{j=1}^{c}|\max\{l_j, r_j\}|\Big]\\ &\leq \sum_{j=1}^{c}\mathsf{E}[|\mathsf{Lap}(z(j))|] + \mathsf{E}\Big[\sum_{j=1}^{c}(l_j + r_j)\Big]\\ &= \|\mathbf{z}\|_1 + \mathsf{E}\Big[\|\mathbf{l}+\mathbf{r}\|_1\Big]\end{aligned}$$

Then, let x_j be the random variable representing noisy aggregate on category j with probability density function $\phi(x_j)$. We analyze the bound of $\mathsf{E}\Big[\|\mathbf{l}+\mathbf{r}\|_1\Big]$ as follows:

$$\begin{aligned}&\mathsf{E}\Big[\|\mathbf{l}+\mathbf{r}\|_1\Big]\\ =\ & \int\cdots\int \mathsf{E}\Big[\|\mathbf{l}+\mathbf{r}\|_1|x_1,\ldots,x_c\Big]\prod_{j=1}^{c}\phi(x_j)dx_1\cdots dx_c\\ \leq\ & \int\cdots\int\Big(\sum_{j=1}^{c}\mathsf{E}\Big[l_j + r_j|x_1,\ldots,x_c\Big]\Big)\prod_{j=1}^{c}\phi(x_j)dx_1\cdots dx_c\\ =\ & \sum_{j=1}^{c}\Big[\int\mathsf{E}\Big[l_j + r_j|x_j\Big]\phi(x_j)dx_j\Big] \leq \sum_{j=1}^{c}|z(j)| = \|\mathbf{z}\|_1\end{aligned}$$

where the first step derives from the law of total expectation; the last step holds because for each category $j \in C$, we have the following inequality:

$$\begin{aligned}\mathsf{E}\Big[l_j + r_j|x_1,\ldots,x_c\Big] &= \mathsf{E}\Big[l_j + r_j|x_j\Big]\\ =\ & \sum_{i=1}^{n}|c_{ij}d_p^r(i) - NA(j)|d_p^r(i) \leq |x_j| = |z_j|\end{aligned}$$

where the last step holds from the fact that $d_p^r(i) \leq 1$ and the optimal solution to relaxed (4.4). $\square$

Time Complexity Analysis. We analyze the time complexity to phase 1 and 2 respectively. The time complexity of phase 1 is determined by solving (4.4). It is not hard to see that both vector variables $\mathbf{z_1}$ and $\mathbf{z_2}$ have dimension c and there are $n + 3c$ constraints. In practice, c is equal to the number of categories from public information, which is actually a constant. Therefore, according to Megiddo [20], the time complexity of solving (4.4) is $O(n)$. In phase 2, we first solve the relaxed (4.5), which is equivalent to solving the non-negative least square programming $\frac{1}{2}\|\mathbf{C}^T\mathbf{d_p^r} - \mathbf{NA}\|_2^2$, *s.t.* $\mathbf{d_p^r} \geq \mathbf{0}$ that has been well studied in literature [2] and will be shown in Section 6.2 with fast running time in practice. The last rounding phase takes another $O(n)$ time.

4.4 Design of Privacy Quantification (C-5)

In this subsection, we design the privacy quantification (C-5) component to quantify "Perturbed Release" level to a privacy budget ϵ, which is used to feed into S-DPDP algorithm in data perturbation component (C-4) discussed above. Specifically, we devise a novel *Single Privacy Budget Quantification (S-PBQ)* algorithm to select a privacy budget ϵ to optimize the utility of perturbed data.

The idea of S-PBQ algorithm is based on our observation that the utility loss of perturbed data will not significantly decrease any more when privacy budget ϵ is larger than some threshold. In detail, S-PBQ algorithm first understands the distribution of noise magnitude to each category aggregate and then search the optimal privacy budget after which utility loss can be negligible. Next, we discuss about the details of these two phases in S-PBQ algorithm.

Phase 1: Noise magnitude determination. We follow the idea of phase 1 in S-DPDP algorithm. The key tweak is based on the following observation: when we replace ϵ in phase 1 of Algorithm 1 with an arbitrary constant α, it is easy to see that each entry in new solution $\mathbf{z}'$ is proportional to that in $\mathbf{z}$ obtained in phase 1 of Algorithm 1. The proportionality constant is equal to $\frac{\epsilon}{\alpha}$, i.e., $\mathbf{z}' = \frac{\epsilon}{\alpha}\mathbf{z}$.

Therefore, in this phase, we consider using $\alpha = 1$ for simplicity and then obtain $\mathbf{z_1'}, \mathbf{z_2'}$ using the following mathematical programming similar as (4.4):

$$\begin{array}{ll}\textit{minimize} & \|\mathbf{z_1'}\|_1\\ \textit{subject to} & \mathbf{Cz_2} \leq \mathbf{1}, \mathbf{z_1'}\mathbf{z_2'}^T \geq \mathbf{I}, \mathbf{z_1'}, \mathbf{z_2'} \geq \mathbf{0}\end{array} \tag{4.6}$$

Our S-PBQ algorithm first solves (4.6) and then sets $\mathbf{z_I'} = \mathbf{z_2'}$ and each entry in $\mathbf{z}'$ as the reciprocal of each entry in $\mathbf{z_I'}$.

Phase 2: Privacy budget optimization. This phase determines the privacy budget ϵ using the above $\mathbf{z_I'}, \mathbf{z}'$. Based on the idea that the utility loss of perturbed data will not significantly decrease after ϵ is larger than some threshold, we search for the optimal ϵ from 0 using a small learning step

```
    Input  : item-category matrix $\mathbf{C}$, learning step rate $\delta$
    Output: quantified optimal privacy budget $\epsilon$
    // Phase 1:  Noise magnitude determination
 1  Solve mathematical programming (4.6);
 2  $\mathbf{z}' \leftarrow$ reciprocal of each entry in $\mathbf{z_I}$;
    // Phase 2:  Privacy budget calculation
 3  Formulate (4.7) with sampled $\mathbf{S}$ and $\mathbf{RandA_s}$;
 4  foreach $i = 1, 2, \dots$ do
 5      Solve relaxed (4.7) with $\epsilon_I = \frac{1}{i\delta}$ and round $\mathbf{d_p}$ (similar
          as line 5-10 in Algorithm 1);
 6      $err(i) \leftarrow \frac{1}{m}\|\mathbf{l}+\mathbf{r}\|_1 + \epsilon_I$ with rounded $\mathbf{d_p}$;
 7      if $i \geq 2$ & $err(i) - err(i-1) \geq err(i-1) - err(i-2)$
          then
 8          $\epsilon \leftarrow (i-1)\delta$;
 9          break;
10      end
11  end
    // Repeat Phase 2
12  Repeat [Phase 2] $N$ times and return averaged $\epsilon$;
```

Algorithm 2: S-PBQ Algorithm

rate δ (we choose $\delta = 0.02$ as an experience value in practice). That is, we solve the following formulation (4.7) by substituting ϵ_I with $\frac{1}{\delta}, \frac{1}{2\delta}, \frac{1}{3\delta}, \dots$ The algorithm terminates at i^{th} iteration when the improvement of optimal utility loss obtained by (4.7) using $\epsilon_I = \frac{1}{i\delta}$ over $\epsilon_I = \frac{1}{(i-1)\delta}$ is no smaller than that using $\epsilon_I = \frac{1}{(i-1)\delta}$ over $\epsilon_I = \frac{1}{(i-2)\delta}$. At last, we set $\epsilon = (i-1)\delta$.

$$\begin{aligned} &\textit{minimize} && \tfrac{1}{2}\|\mathbf{l}+\mathbf{r}\|_2^2 \\ &\textit{subject to} && \mathbf{RandA_s} + \epsilon_I \mathbf{S}\mathbf{z}' - \mathbf{l} \leq \mathbf{C}^T\mathbf{d_p} \\ &&& \leq \mathbf{RandA_s} + \epsilon_I \mathbf{S}\mathbf{z}' + \mathbf{r} \\ &&& \mathbf{d_p} \in \{0,1\}^n \end{aligned} \tag{4.7}$$

where $\mathbf{S} = diag(s_1, \dots, s_m)$, in which each diagonal entry s_j is sampled from Laplace distribution $\mathsf{Lap}(1)$ such that $z'(j)s(j)$ is a sample from Laplace distribution $\mathsf{Lap}(z'(j))$ due to the property of Laplace distribution $kX \sim \mathsf{Lap}(k\mu, k\beta)$ for a random variable $X \sim \mathsf{Lap}(\mu, \beta)$ for a positive constant k [16]. In addition, in order to avoid potential privacy leakage caused by ϵ quantification, we do not use user's private history $\mathbf{d_r}$. Instead, we consider using $\mathbf{RandA_s}$, the category aggregates on randomly selected items from all public items. For simplicity, we use uniform sampling with sampling rate as the averaged number of items for each user.

Our S-PBQ algorithm repeatedly samples different $\mathbf{S}$ and $\mathbf{RandA_s}$ to obtain ϵ_I by solving (4.7) in this phase. In the experiment, we select the number of repeat times N to be 10 by considering the efficiency of S-PBQ algorithm. At last, the averaged ϵ is chosen as the quantified privacy budget.

Time Complexity Analysis. S-PBQ algorithm is similar as S-DPDP algorithm with $O(n)$ (phase 1 and rounding steps in phase 2) and the running time to solve relaxed (4.7). The second phase has its time complexity $O(T)$ similar as that of solving relaxed (4.5). To provide strong privacy guarantee, we consider $\epsilon \leq 1$ and therefore the repeat of phase 2 will take at most $O(T+n)$. Our experimental results in Figure 7 shows the efficient running time in practice.

4.5 Overall Analysis of S-EpicRec System

We summarize the performance of S-EpicRec, that is, the output of S-DPDP algorithm with the quantified privacy budget using S-PBQ algorithm.

Privacy Guarantee. The perturbed data using S-EpicRec satisfies ϵ-differential privacy where ϵ is determined by S-PBQ algorithm (Algorithm 2).

Utility Guarantee. The expected MAE between raw and perturbed category aggregates (output of S-EpicRec) is upper bounded by $O(\frac{\|\mathbf{C}\|_1}{c})$, where $\|\mathbf{C}\|_1 = \max_{1 \leq i \leq n} \sum_{j=1}^{c} c_{ij}$.

It is not hard to see that the optimal solution of (4.6) is upper bounded by $\|\mathbf{C}\|_1$. Therefore, the optimal solution of (4.4) is upper bounded by $O(\frac{\|\mathbf{C}\|_1}{c})$ based on Theorem 3, in which the constant factor is dependent on the quantified ϵ.

Time Complexity. The overall time complexity of S-EpicRec is $O(n+T)$ where T is the time complexity of solving relaxed (4.5) as discussed in [2].

5. DESIGN OF M-EpicRec: MULTI-LEVEL PRIVACY CONTROL

In this section, we further design a M-EpicRec framework to enable the category-based privacy concern controls. The idea of M-EpicRec is extended from S-EpicRec proposed in Section 4. Basically, we consider using the same privacy and utility notions for designing C-4 and C-5 components in M-EpicRec system. The rest of this section focuses on the different parts between S-EpicRec and M-EpicRec, mainly in terms of notations and the design of C-4 and C-5.

5.1 Notations

In M-EpicRec, the only different notation from those in S-EpicRec is in user privacy control component (C-3). Specifically, we define the vector of privacy concern levels $\mathbf{PT} = \{\mathsf{PT}(1), \dots, \mathsf{PT}(c)\}$ to replace a single PT. Each entry $\mathsf{PT}(j)$ is the user-specified privacy concern level for category j, which still belongs to one of the same three levels {*"No Release", "Perturbed Release", "All Release"*}. Correspondingly, we define the vector privacy budget $\boldsymbol{\epsilon} = \{\epsilon(1), \dots, \epsilon(c)\}$, where $\epsilon(j) = \epsilon$ when $\mathsf{PT}(j)$ is selected as *"Perturbed Release"* and $\epsilon(j) = 0$ when $\mathsf{PT}(j)$ is selected as either *"No Release"* or *"All Release"* indicating that no randomness is considered for these categories.

Moreover, we define three vectors $\mathbf{L_n}, \mathbf{L_p}, \mathbf{L_a}$ with respect to three privacy concern levels {*"No Release", "Perturbed Release", "All Release"*}, based on $\mathbf{PT}$. For example, when $\mathbf{PT}$=(no, perturbed, no, all, all, perturbed) w.r.t. 6 categories, we have $\mathbf{L_n} = (1,0,1,0,0,0), \mathbf{L_p} = (0,1,0,0,0,1), \mathbf{L_a} = (0,0,0,1,1,0)$. Note that each category has one privacy tolerance level and $\mathbf{L_n} + \mathbf{L_p} + \mathbf{L_n} = \mathbf{1}$.

5.2 Design of Data Perturbation (C-4)

5.2.1 Problem Definition

We consider M-Perturbation Problem which differs from S-Perturbation Problem defined in Section 4.3.1 from the following aspects: (1) M-Perturbation problem takes two different inputs: the category-based privacy concern levels $\mathbf{L_n}, \mathbf{L_p}, \mathbf{L_a}$ (derived from $\mathbf{PT}$) and a privacy budget ϵ for categories with "perturbed release" privacy concern level; (2) M-Perturbation problem targets on maintaining the quality of category aggregates on perturbed data only associated with "perturbed release" categories while releasing no data in "no release" categories and all raw data only associated with "all release" categories. (Note that we choose to prioritize privacy protection when it conflicts with utility. That is, we do not release an item as long as one of its categories is set "no release"; we perturb an item if one of its categories is set "perturbed release" privacy concern level and none of its categories is set "no release"; we release an item only if all of its categories are set "all release".)

5.2.2 Challenges

In addition to the challenges of S-Perturbation problem described in Section 4.3.2, M-Perturbation problem poses some additional challenges mainly from the constraints from "no release" and "all release" categories. Especially when an item belongs to multiple categories, the preservation of utility on category aggregates on perturbed data in the "perturbed release" categories becomes more difficult.

5.2.3 Proposed M-DPDP Approach

In this subsection, we focus on discussing about the novel part of M-DPDP approach beyond S-DPDP approach (in Section 4.3.3), which is developed to support multi-level privacy control. Specifically, as the key idea of these two approaches are consistent, we will mainly present the difference in phase 1 and 2 respectively.

Phase 1: Noise calibration. The major difference between phase 1 in M-DPDP and that in S-DPDP is that the noises are injected only on categories with "perturbed release" privacy level while the noise magnitude on other categories are enforced to be 0. Thus, (4.3) can be rewritten as follows:

$$\begin{array}{ll} \textit{minimize} & \|\mathbf{z}\|_1 \\ \textit{subject to} & \mathbf{C}\mathbf{z_I}\mathbf{L_p}\mathbf{I} \le \epsilon\mathbf{1}, (\mathbf{L_n}+\mathbf{L_a})^T\mathbf{z} = 0 \\ & (\mathbf{L_n}+\mathbf{L_a})^T\mathbf{z_I} = 0, \mathbf{z}, \mathbf{z_I} \ge \mathbf{0} \end{array} \quad (5.1)$$

where the first constraint imposes the satisfaction of differential privacy on these categories with "perturbed release" privacy level; the next two constraints ensure no noise calibration into other categories. Accordingly, the quadratic programming (4.4) can be rewritten as follows:

$$\begin{array}{ll} \textit{minimize} & \|\mathbf{z_1}\|_1 \\ \textit{subject to} & \mathbf{C}\mathbf{z_2}\mathbf{L_p}\mathbf{I} \le \epsilon\mathbf{1} \\ & (\mathbf{L_n}+\mathbf{L_a})^T\mathbf{z_1} = 0, (\mathbf{L_n}+\mathbf{L_a})^T\mathbf{z_2} = 0 \\ & \mathbf{z_1}\mathbf{z_2}^T \ge \mathbf{I}^T\mathbf{L_p}\mathbf{I}, \mathbf{z_1}, \mathbf{z_2} \ge \mathbf{0} \end{array} \quad (5.2)$$

In this phase, M-DPDP algorithm solves (5.2) and sets $z(j) = \frac{1}{z_2(j)}$ if category j has "perturbed release" privacy level and $z(j) = 0$ otherwise.

Phase 2: Data sanitization. The major difference in phase 2 is from the following two aspects:

First, in order to address the constraints of categories with "no release" and "all release" privacy levels, we select "all release" user data (denoted as $\mathbf{d_r^a}$) from user raw data $\mathbf{d_r}$, where the i^{th} entry in $\mathbf{d_r^a}$ is 1 if and only if this data belongs to user raw data (i^{th} entry in $\mathbf{d_r}$ is 1) and all of its associated categories have "all release" privacy levels.

Second, we inject $\mathsf{Lap}(z(j))$ into the j^{th} category for those categories with "perturbed release" privacy level. Then, we reformulate (4.7) as follows:

$$\begin{array}{ll} \textit{minimize} & \frac{1}{2}\|\mathbf{l}+\mathbf{r}\|_2^2 \\ \textit{subject to} & \mathbf{NA} - \mathbf{l} \le \mathbf{C}^T\mathbf{d_p} \le \mathbf{NA} + \mathbf{r} \\ & \mathbf{d_p} \ge \mathbf{d_r^a}, \mathbf{d_p} \in \{0,1\}^n \\ & (\mathbf{L_n}+\mathbf{L_a})^T\mathbf{l} = 0, (\mathbf{L_n}+\mathbf{L_a})^T\mathbf{r} = 0 \end{array} \quad (5.3)$$

where $NA(j) = \sum_{i\in I} c_{ij}d_r(i)$ for categories with "all release" privacy level; $NA(j) = 0$ for categories with "no release" privacy level; and $NA(j) = \sum_{i\in I} c_{ij}d_r(i) + \mathsf{Lap}(z(j))$ for categories with "perturbed release" privacy level. The second constraint imposes that the raw data only in categories with "all release" privacy level will be released; the last two constraints guarantee the equivalence for categories with "all release" and "no release" privacy levels.

5.2.4 Theoretical Analysis.

THEOREM 4 (M-DPDP PRIVACY ANALYSIS). *M-DPDP algorithm enjoys ϵ-differential privacy.*

THEOREM 5 (M-DPDP UTILITY ANALYSIS). *The expected MAE between raw and perturbed aggregates (via M-DPDP algorithm) is upper bounded by $2\|\mathbf{z}\|_1/c$, where $\mathbf{z}$ is the optimal solution of (5.2).*

The proofs are similar as Theorem 2, 3 and omitted due to space limit. The time complexity of M-DPDP is also similar as S-DPDP and omitted.

5.3 Design of Privacy Quantification (C-5)

We propose M-PBQ approach in this section, extending from S-PBQ algorithm in S-EpicRec system (Section 4.4). Similarly, in phase 1, M-PBQ replace unknown ϵ in (5.2) by 1 and solve it to obtain $\mathbf{z}'$.

The difference between them mainly lies in the second phase. In phase 2, we substitute (4.7) in Algorithm 2 with the following formulation:

$$\begin{array}{ll} \textit{minimize} & \frac{1}{2}\|\mathbf{l}+\mathbf{r}\|_2^2 \\ \textit{subject to} & \mathbf{RandA_m} + \epsilon_I\mathbf{S}\mathbf{z}' - \mathbf{l} \le \mathbf{C}^T\mathbf{d_p} \\ & \le \mathbf{RandA_m} + \epsilon_I\mathbf{S}\mathbf{z}' + \mathbf{r} \\ & \mathbf{d_p} \ge \mathbf{d_r^a}, \mathbf{d_p} \in \{0,1\}^n \\ & (\mathbf{L_n}+\mathbf{L_a})^T\mathbf{l} = 0, (\mathbf{L_n}+\mathbf{L_a})^T\mathbf{r} = 0 \end{array} \quad (5.4)$$

where $\mathbf{S} = \textit{diag}(s_1, \ldots, s_m)$, in which the j^{th} diagonal entry $s(j)$ is sampled from Laplace distribution $\mathsf{Lap}(1)$ if this category has "perturbed release" privacy level and 0 otherwise; $\mathbf{RandA_m}$ is the category aggregates on randomly selected items from all public items, in which $Rand_m(j) = 0$ for categories with "no release" privacy level. The time complexity of M-PBQ is similar as S-PBQ and omitted.

5.4 Performance Analysis of M-EpicRec

We summarize the performance of M-EpicRec as follows:

Privacy Guarantee. The category aggregates of perturbed data from M-EpicRec satisfy ϵ-differential privacy where ϵ is determined by M-PBQ algorithm.

Utility Guarantee. The expectation of MAE between raw and perturbed category aggregates (output of M-EpicRec) is upper bounded by $O(\frac{\|\mathbf{C}\|_1}{c})$, where $\|\mathbf{C}\|_1 = \max_{1\le i\le n}\sum_{j=1}^m c_{ij}$.

Time Complexity. The overall time complexity of S-EpicRec is $O(n+T)$ where T is the time complexity of solving relaxed (5.3) as discussed in [2].

6. EXPERIMENTAL EVALUATION

6.1 Datasets, Metrics, Competitors & Settings

Datasets. We test EpicRec on two real-world datasets:

MovieLens[1]*:* a movie rating dataset collected by the GroupLens Research Project at the University of Minnesota through the website movielens.umn.edu during the 7-month period from September 19th, 1997 through April 22nd, 1998. The number of movie categories is 18. We use the MovieLens-1M, with 1000,209 ratings from 6,040 users on 3,883 movies.

[1]http://grouplens.org/datasets/movielens

Yelp[2]*:* a business rating data provided by RecSys Challenge 2013, in which Yelp reviews, businesses and users are collected at Phoenix, AZ metropolitan area. The number of business categories is 21. We use all reviews in training dataset, with 229,907 reviews from 43,873 users on 11,537 businesses.

Metrics. We evaluate perturbation quality of S-EpicRec and M-EpicRec systems from the following aspects:

• *Perturbed Category Aggregates Quality:* we use the expected MAE metric discussed in Section 4.1.2 with its mathematical definition in Section 4.3.4;

• *Recommendation Accuracy:* we consider the *MAE Loss* between the recommendation results using raw and perturbed data, defined as $\frac{\sum_{i=1}^{n}\sum_{u=1}^{U}|Rec_p^{ui}-GT^{ui}|}{\sum_{i=1}^{n}\sum_{u=1}^{U}|Rec_r^{ui}-GT^{ui}|}-1$, where n is the number of items, U is the number of all users and Rec_r^{ui}, Rec_p^{ui} are the elements in i^{th} column and u^{th} row (item i for user u) in predicted recommendation matrices $\mathbf{Rec}_r, \mathbf{Rec}_p$ using user raw data $\mathbf{d_r}$ and perturbed data $\mathbf{d_p}$.

In addition, we also show the scalability of our EpicRec system is further validated via running time.

(Pseudo) Competitors. As this paper is the first attempt for designing a privacy preserving system to enable user-understandable privacy concern control, there is no existing work to fairly compare with our approach. Therefore, we consider embedding existing approaches into our system, called *pseudo-competitors*. Specifically, we plug them into S-DPDP/M-DPDP algorithms to replace phase 1 for noise calibration only, which first uses our quantified privacy budget ϵ from S-PBQ/M-PBQ algorithms and then sanitizes data by phase 2 in S-DPDP/M-DPDP algorithms.

We plug the following two existing noise calibration algorithms into EpicRec system for comparison:

• *Pseudo-LPA (Pseudo Laplace Mechanism):* the baseline method that injects Laplace perturbation noise to each count using domain-specific global sensitivity Δf;

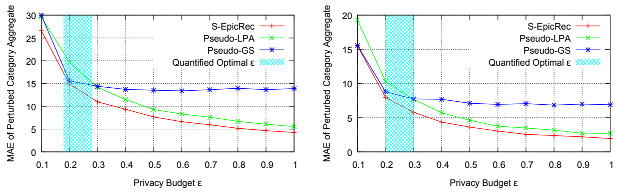
• *Pseudo-GS (Pseudo Grouping&Smoothing Mechanism):* the best method for aggregate release with grouping and smoothing proposed in [14].

Note that we do not compare with the approach in [24] since it only supports larger ϵ ($\epsilon > 1$) which our proposed EpicRec focuses on stronger privacy protection with $\epsilon \leq 1$.

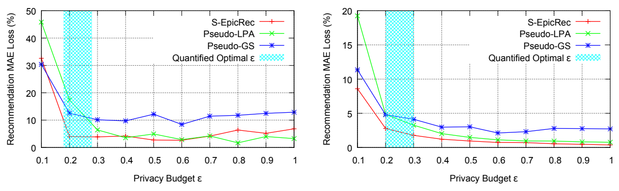
Settings. We conduct the classic recommender system algorithm, collaborative filtering [25], using GraphLab[3]. The only parameter δ is set to 0.02 according to differential privacy literature. We test the experiments on personal computer which is equipped with 1.9GHz CPU and 8GB RAM. We run each experiment 10 times and report the average result. To evaluate recommendation accuracy, we use 10-fold cross-validation and stochastic gradient descent algorithm for collaborative filtering. In M-EpicRec case, we randomly select the privacy levels for each category for each user.

6.2 Evaluation Results

Perturbation Quality. Figure 5 reports the results of S-EpicRec system. As shown in Figure 5(a), the perturbed category aggregates quality of S-DPDP algorithm in S-EpicRec system continuously outperforms other competitors up to 10% in both MovieLens and Yelp datasets. The reason is that our S-EpicRec determines the calibrated noises based on the underlying data property via the correlation between



(a) Perturbed Category Aggregate Quality



(b) Recommendation Accuracy

Figure 5: S-EpicRec Results (L:MovieLens; R:Yelp)

categories as described in (4.3), thereby capturing the minimum noise magnitude that is sufficient to ensure the privacy guarantees. In the meanwhile, data sanitization (phase 2 in Algorithm 1) can also take advantage of the category correlations to minimize the error when sanitizing data. On the other hand, the grouping and averaging of category aggregates in Pseudo-GS loses a lot of correlation information between categories and lead to a larger error. Likewise, Pseudo-LPA applies the global sensitivity to determine the noise magnitude, which only captures the maximum number of categories an item can belong to and largely ignores the correlation between categories. Moreover, the recommendation loss in Figure 5(b) is up to 5% and 3% in MovieLens and Yelp datasets with strong privacy guarantees ($\epsilon = 0.2$ in MovieLens and $\epsilon = 0.3$ in Yelp).

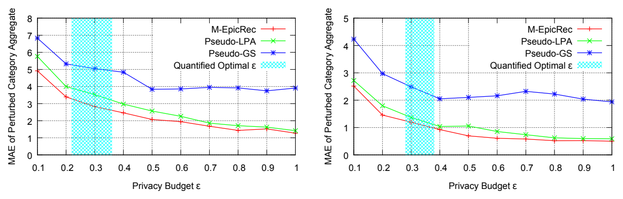
Figure 6 shows the similar performance results in M-EpicRec system that our M-DPDP algorithm outperforms other competitors from both aspects. The recommendation quality of M-DPDP algorithm (in Figure 6(b)) has slightly worse than S-DPDP algorithm in S-EpicRec due to constraints from items in "No Release" categories.

One may question that what if a user has privacy concern on some category rather than particular items in this category. The bottom two figures in Figure 6(b) show that our proposed M-EpicRec framework can indeed provide this function by allowing user to select "no release" for those categories. Thanks to the correlation between categories (an item usually belongs to many categories), we did some additional experiments showing that the recommendation MAE loss on movies in "no release" categories is less than 5% worse than that in "perturbed release" categories.

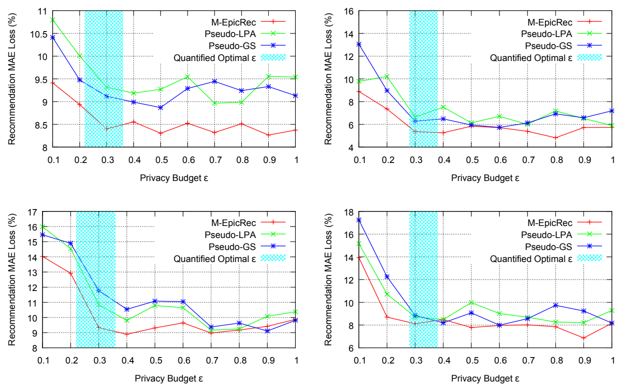
Privacy Budget Quantification. As we can see in Figure 5, the blue shadows show the range of quantified optimal privacy budget ϵ spanning in our 10 testings. It is interesting to see that our quantified ϵ values fall in the range of around $\epsilon = 0.23$ in MovieLens dataset and $\epsilon = 0.25$ in Yelp dataset respectively, after which the recommendation loss does not reduce dramatically and maintains relatively stable. Similar observations are shown in Figure 6 for category-based privacy control, with $\epsilon = 0.29$ in MovieLens dataset and $\epsilon = 0.33$ in Yelp dataset.

Scalability. Figure 7 shows that the running time of both S-EpicRec and M-EpicRec systems is no longer than 0.7 and 1.5 seconds respectively. It takes slightly longer on Yelp dataset since the number of public items is relatively

[2]https://www.kaggle.com/c/yelp-recsys-2013/data
[3]http://select.cs.cmu.edu/code/graphlab/pmf.html



(a) Perturbed Category Aggregate Quality



(b) Recommendation Accuracy (Top: "Perturbed Release" Categories; Bottom: "No Release" Categories)

Figure 6: M-EpicRec Results(L:MovieLens; R:Yelp)

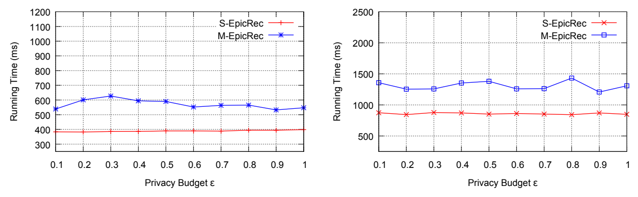


Figure 7: Running Time (L:MovieLens; R:Yelp)

larger than that on MovieLens dataset. M-EpicRec takes longer than S-EpicRec due to its more complex optimization process (with more constraints). As the perturbation process is usually conducted offline, the running time of our proposed framework is considered good enough.

7. IMPLEMENTATION

In this section, we present the implementation of a proof-of-concept EpicRec system. As shown in Figure 8, we implement a web-based EpicRec client for movie recommendation, incorporated with a standard recommender server using classic recommendation approaches. The rest of this section first briefly discusses about server side implementation and then focuses on the implementation of each component in EpicRec client. We implement EpicRec client on a laptop with Ubuntu Desktop OS and the recommender server on a workstation with Ubuntu Server OS.

7.1 Movie Recommendation Service Provider

In our PoC system, we use a workstation with Ubuntu Server OS as the recommender system. We conduct the personalized recommendation using collaborative filtering (stochastic gradient descent algorithm) via GraphLab[3]. Recommendation results are ranked overall and in each category. All transmissions between client and server are via SSL/TLS security protocol.

7.2 EpicRec for Movie Recommendation

On the device side, we maintain a local database to store the input from public data input (C-1) component and user private data input (C-2) component. Then, we design and implement user interfaces for user privacy control (C-3) component and read the data from user input.

Public Data Input (C-1): We crawl ∼6,000 recent movies' meta-data from the public "My API Films" website[4], including movie title, genre, plot description and poster image; we then store them in the table "allMovies" in local database. Moreover, the physical files of poster images are stored locally with the corresponding names as in the allMovies table. Each movie/record in this table is associated with an additional boolean attribute "watched", which is initialized as 0 (indicating that no movies have been watched).

User Private Data Input (C-2): In order to obtain user private history of watched movies, we implement in a simplified manner by scanning the history files of each web browser. We mainly focus on two popular web browsers, Google Chrome and Mozilla Firefox, where we download the history files "∼/.config/google-chrome/Default/History" and "∼/.mozilla/firefox/*.default /places.sqlite*" respectively. We next search for each movie title in all these history files and update the "watched" attribute to 1 if a movie's title exists in the history file.

User Privacy Control Input (C-3): We designed the user interface for user privacy control input (see C-3 in Figure 8), in which a user is allowed first to select his overall privacy concern level from "no release", "perturbed release" or "all release". If "perturbed release" is selected, user can further select if he wants to select different privacy concern levels for different categories of movies. If so, a list of categories is popped up with privacy concern level drop-down boxes.

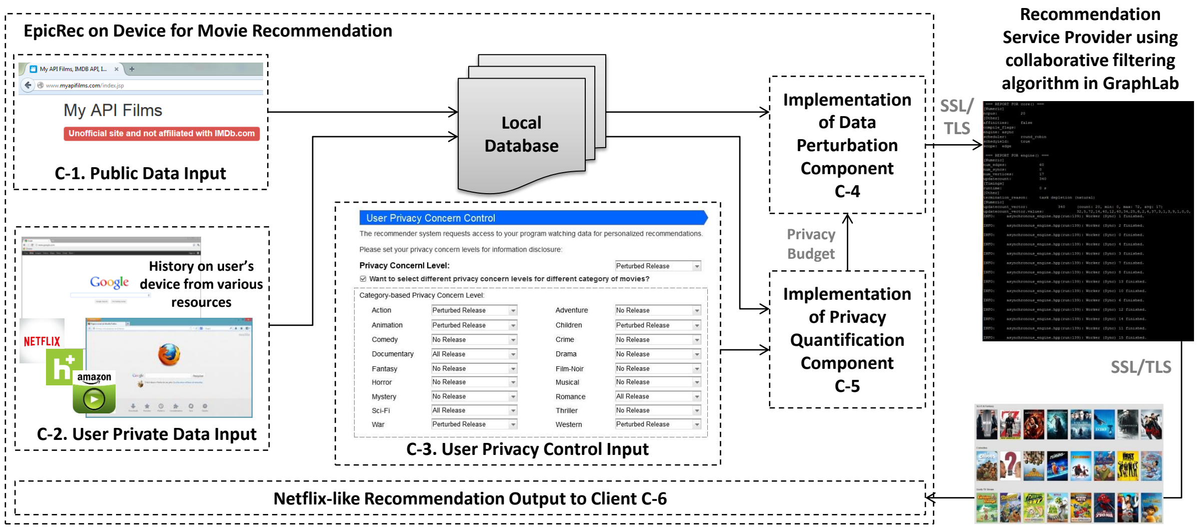
Privacy Quantification (C-5) & Data Perturbation (C-4): If a user selects "perturbed release" and does not check the box to set category-based privacy concern levels, we call C-4 and C-5 components in S-EpicRec system. (When "no release" or "all release" is selected, we simply release no data or all raw data.) Otherwise, we call C-4 and C-5 components in M-EpicRec system to support user specified category-based multiple privacy concern levels.

Recommendation Output (C-6): We simply use a netflix-style output to provide users overall and per-category top movie recommendation. Moreover, the categories are ranked on the client side by the number of movies the user actually watched to capture user's preference on different categories.

8. CONCLUSION AND FUTURE WORK

Conclusion. In this paper, we designed a novel practical privacy-preserving system on client, EpicRec, for personalized recommendation via state-of-the-art differential privacy. EpicRec provides users a privacy control interface such that users can control their privacy concerns in a way they understand and of their preferred granularities, either overall or category-based concerns. EpicRec further quantifies these layman privacy concern levels to privacy budget, which is next used as input to conduct data perturbation algorithm via differential privacy. With these key components, EpicRec can also work with other data collection and output components. We believe this is an important step

[4]http://www.myapifilms.com/index.jsp


EpicRec on Device for Movie Recommendation
My API Films
Unofficial site and not affiliated with IMDb.com
C-1. Public Data Input
History on user's device from various resources
NETFLIX
amazon
C-2. User Private Data Input
Local Database
C-3. User Privacy Control Input
Implementation of Data Perturbation Component C-4
Privacy Budget
Implementation of Privacy Quantification Component C-5
SSL/ TLS
Recommendation Service Provider using collaborative filtering algorithm in GraphLab
SSL/TLS
Netflix-like Recommendation Output to Client C-6

User Privacy Concern Control

The recommender system requests access to your program watching data for personalized recommendations.

Please set your privacy concern levels for information disclosure:

Privacy Concernl Level: Perturbed Release

☑ **Want to select different privacy concern levels for different category of movies?**

Category-based Privacy Concern Level:

Action	Perturbed Release	Adventure	No Release
Animation	Perturbed Release	Children	Perturbed Release
Comedy	No Release	Crime	No Release
Documentary	All Release	Drama	No Release
Fantasy	No Release	Film-Noir	No Release
Horror	No Release	Musical	No Release
Mystery	No Release	Romance	All Release
Sci-Fi	All Release	Thriller	No Release
War	Perturbed Release	Western	Perturbed Release

Figure 8: Proof-of-Concept Implementation of EpicRec for Movie/TV Recommendation

towards designing a practical privacy-preserving system for personalized recommendation.

Future work. We will extend EpicRec into more comprehensive and practical cases from different aspects: (1) we will improve the implementation of C-1 and C-2 components in our PoC system by potentially developing browser extensions; (2) we will conduct a large-scale field study to observe and understand users' natural behaviors, in-situ attitude and perceptions when using our browser extensions to interact with EpicRec system; (3) we will continue to develop data perturbation techniques to support user's streaming private data; different types of user private data; and allow users to iteratively adjust their privacy levels for trading off privacy and recommendation.

9. REFERENCES

[1] L. Bonomi, L. Xiong, and J. J. Lu. Linkit: privacy preserving record linkage and integration via transformations. In *SIGMOD*, pages 1029–1032, 2013.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] J. Canny. Collaborative filtering with privacy. In *IEEE Symposium on S&P*, pages 45–57, 2002.

[4] T.-H. H. Chan, M. Li, E. Shi, and W. Xu. Differentially private continual monitoring of heavy hitters from distributed streams. In *PETS*, pages 140–159, 2012.

[5] K. Chaudhuri, A. Sarwate, and K. Sinha. Near-optimal differentially private principal components. In *NIPS*, pages 989–997. 2012.

[6] K. Chaudhuri and S. A. Vinterbo. A stability-based validation procedure for differentially private machine learning. In *NIPS*, pages 2652–2660. 2013.

[7] C. Dwork. Differential privacy: A survey of results. In *TAMC*, pages 1–19, 2008.

[8] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[9] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42(4):14:1–14:53, 2010.

[10] A. Guha Thakurta and A. Smith. (nearly) optimal algorithms for private online learning in full-information and bandit settings. In *NIPS*, pages 2733–2741. 2013.

[11] M. Hardt, K. Ligett, and F. Mcsherry. A simple and practical algorithm for differentially private data release. In *NIPS*, pages 2339–2347. 2012.

[12] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *VLDB*, 3(1-2):1021–1032, 2010.

[13] B. Heitmann, J. G. Kim, A. Passant, C. Hayes, and H.-G. Kim. An architecture for privacy-enabled user profile portability on the web of data. In *HetRec*, pages 16–23, 2010.

[14] G. Kellaris and S. Papadopoulos. Practical differential privacy via grouping and smoothing. *VLDB*, 6(5):301–312, Mar. 2013.

[15] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas. Releasing search queries and clicks privately. In *WWW*, pages 171–180, 2009.

[16] S. Kotz, T. J. Kozubowski, and K. Podgorski. *The Laplace distribution and generalizations: a revisit with applications to communications, economics, engineering, and finance*. 2001.

[17] B. Liu and U. Hengartner. ptwitterrec: A privacy-preserving personalized tweet recommendation framework. In *Proceedings of ASIA CCS*, pages 365–376, 2014.

[18] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In *ICDE*, pages 277–286, 2008.

[19] F. McSherry and I. Mironov. Differentially private recommender systems: Building privacy into the net. In *KDD*, pages 627–636, 2009.

[20] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31(1):114–127, Jan. 1984.

[21] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh. Privacy-preserving matrix factorization. In *CCS*, pages 801–812, 2013.

[22] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84, 2007.

[23] H. Polat and W. Du. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *ICDM*, pages 625–628, 2003.

[24] Y. Shen and H. Jin. Privacy-preserving personalized recommendation: An instance-based approach via differential privacy. In *ICDM*, pages 540–549, 2014.

[25] P. Symeonidis, A. Nanopoulos, A. N. Papadopoulos, and Y. Manolopoulos. Collaborative recommender systems: Combining effectiveness and efficiency. *Expert Syst. Appl.*, 34(4):2995–3013, 2008.

[26] J. Wang, N. Wang, and H. Jin. Context matters?: How adding the obfuscation option affects end users' data disclosure decisions. In *IUI*, pages 299–304, 2016.

[27] Y. Xin and T. Jaakkola. Controlling privacy in recommender systems. In *NIPS*, pages 2618–2626. 2014.

[28] J. Xu, Z. Zhang, X. Xiao, Y. Yang, and G. Yu. Differentially private histogram publication. In *ICDE*, pages 32–43, 2012.

[29] B. Zhang, N. Wang, and H. Jin. Privacy concerns in online recommender systems: Influences of control and user data input. In *SOUPS*, pages 159–173, 2014.

[30] S. Zhang, J. Ford, and F. Makedon. Deriving private information from randomly perturbed ratings. In *SDM*, pages 59–69, 2006.