Collecting Preference Rankings under Local Differential Privacy

Xiang Cheng, Jianyu Yang, Yufei Wang, Sen Su, Rui Chen, and Yuejia Li

Abstract—With the deep penetration of the Internet and mobile devices, preference rankings are being collected on a massive scale by diverse data collectors for various business demands. However, users' preference rankings in many applications are highly sensitive. Without proper privacy protection mechanisms, it either puts individual privacy in jeopardy or hampers business opportunities due to users' unwillingness to share their true rankings. In this paper, we initiate the study of collecting preference rankings under local differential privacy. The key technical challenge comes from the fact that the number of possible rankings could be large in practical settings, leading to excessive injected noise. To solve this problem, we present a novel approach SAFARI, whose main idea is to collect a set of distributions over small domains which are carefully chosen based on the riffle independent (RI) model to approximate the overall distribution of users' rankings, and then generate a synthetic ranking dataset from the obtained distributions. By working on small domains instead of a large domain, SAFARI can significantly reduce the magnitude of added noise. In SAFARI, we design two transformation rules, namely Rule I and Rule II, to instruct users to transform their data to provide the information about the distributions of the small domains. We propose a new method called LADE to precisely estimate the required distributions used for the structure learning of RI model. We also propose a new LDP method called SAFA for frequency estimation over multiple attributes that have small domains. We formally prove that SAFARI guarantees ε-local differential privacy. Extensive experiments on real datasets confirm the effectiveness of SAFARI.

Index Terms—Preference Rankings, Data Collection, Riffle Independent Model, Local Differential Privacy

1 Introduction

Preference ranking is one of the most common representations of personal data, which places different items into a total order based on individual opinions about their relative quality. With the deep penetration of the Internet and mobile devices, collecting and analyzing users' rankings are essential for service providers to supply better user experience and generate new revenue opportunities. However, users' preference rankings in many applications (e.g., political preference) are highly sensitive. Without proper privacy protection mechanisms, it either puts individual privacy in jeopardy or hampers business operations due to users' unwillingness to share their true rankings. Thus, developing solutions to address such privacy concerns in collecting preference rankings is an urgent need.

Local differential privacy (LDP) is the state-of-the-art privacy model for protecting individual privacy in the process of data collection. It has been adopted in real-world applications, including Google Chrome browser [1] and macOS [2]. Different from the traditional differential privacy setting [3], which assumes a trusted data collector that has access to the private data, LDP does not make assumptions about the credibility of the data collectors. In LDP, each user locally perturbs his/her real data before sending it to the collector. Since real data never leaves from users' devices, LDP protects both users against privacy risks and data collectors against damages due to potential privacy breaches.

In this paper, we, for the first time, investigate the problem of collecting preference rankings under LDP. Given a large number of users who have a ranking of a set of items, an untrusted data collector aims at collecting the rankings from the users while satisfying LDP. A straightforward solution is to consider each user's ranking as a categorical value in a domain that consists of all possible rankings, directly collect the overall distribution of users' rankings by existing LDP methods such as RAPPOR [1], and synthesize rankings from the obtained distribution. Nevertheless, since the number of possible rankings increases factorially in the number of items to rank, even a relatively small number of items lead to a large number of possible rankings. For existing LDP methods, this implies excessive noise to be added and thus leads to low data utility.

To address the deficiencies of the straightforward solution, we propose a novel effective solution to collect preference rankings under LDP. In contrast to collecting the overall distribution of rankings directly, we propose to collect a set of carefully chosen distributions over small domains to approximate the overall distribution of users' rankings. By working on small domains instead of a large domain, we can significantly reduce the magnitude of noise added to the collected data.

For this purpose, we make use of the riffle independent (RI) model [4], [5], a well-established probabilistic graphical model in the field of machine learning, to model rankings. The RI model can approximate the overall distribution of rankings through low-dimensional distributions. Our approach, called **S**ampling R**A**ndomizer **F**or Multiple **A**ttributes with **R**iffle **I**ndependent Model (SAFARI), proceeds as follows. At the beginning of SAFARI, each user transforms his/her ranking into multi-attribute data by a transformation rule, namely Rule I, to provide the information about the necessary distributions used for estimating all those distributions required for learning the structure of the RI model. In particular, such an estimation process is performed via a **L**asso b**A**sed **D**istribution **E**stimation (LADE) method cor-

- *Xiang Cheng, Jianyu Yang, Sen Su, Yufei Wang and Yuejia Li are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. E-mail: {chengxiang, jyyang, wyfs4321, susen, liyuejia}@bupt.edu.cn*
- *Rui Chen is with the College of Computer Science and Technology, Harbin Engineering University, Harbin, China. E-mail: ruichen@hrbeu.edu.cn*

responding to Rule I. Then, the data collector uses **S**ampling RA**n**domizer **F**or Multiple **A**ttributes (SAFA), a new LDP method for frequency estimation over multiple attributes, to collect the necessary distributions from the users' data transformed by Rule I. Next, the data collector employs LADE to estimate all required distributions from the collected distributions and construct the structure of the RI model. Subsequently, each user transforms his/her ranking into multi-attribute data by another transformation rule, namely Rule II, to provide the distribution information for learning the parameters of RI model. After that, the data collector uses SAFA to collect information from the users' data transformed by Rule II to obtain the parameters of RI model. Finally, the data collector synthesizes rankings from the learned model.

The main contributions of this work are summarized as follows:

- We formulate the problem of preference ranking collection under LDP and present a novel approach SAFARI based on the RI model. While being proposed for collecting rankings, our idea is also applicable to other types of data.
- We design two transformation rules to instruct users to transform their data into multi-attribute data in which each attribute has a small domain. We propose a new method called LADE to precisely estimate the required distributions used to learn the RI model's structure.
- We propose a new LDP method called SAFA for frequency estimation over multiple attributes that have small domains. We theoretically analyze the privacy and utility guarantees of SAFA, and show that SAFA achieves better data utility than the state-of-the-art technique [6].
- We show that SAFARI guarantees ε-LDP. Experimental results over real datasets demonstrate the effectiveness of SAFARI.

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 provides the preliminaries and problem statement. Section 4 describes four baseline approaches. Section 5 gives the details of SAFARI. Section 6 shows our experimental results. Finally, Section 7 concludes our work.

2 Related Work

The local privacy model was first formalized in [7]. To achieve LDP, there exist several canonical methods [1], [8], [9], [10], which are used as building blocks of LDP solutions. Randomized response (RR) [8], first studied in survey design, can be adapted to estimate the statistics of a binary attribute under LDP. For collecting the frequencies of categorical values, Erlingsson *et al.* [1] propose RAPPOR by applying RR to a Bloom filter. However, when the number of categorical values is large, RAPPOR will lead to high communication cost. To tackle this problem, Bassily and Smith [9] propose succinct histogram (SH), in which each user only needs to send one bit to the data collector. Wang *et al.* [10] introduce generalized randomized response (GRR) and optimized local hashing (OLH), and provide a guideline as to which method to use in different scenarios.

For the case where the domain of the categorical values are unknown, Fanti *et al.* [11] present an extension of RAPPOR to estimate the joint distribution of multiple variables. In addition, Kairouz *et al.* [12] propose two algorithms using hash functions and cohorts to estimate discrete distribution under LDP.

In recent years, finding heavy hitters under LDP has been extensively studied. Assuming that each user has only one item drawn from an item set, Hsu *et al.* [13] design two algorithms based on random projection and concentration of measure to estimate heavy hitters from users' data. In the same problem setting, Bassily *et al.* [14] present two local differentially private algorithms, called TreeHist and Bitstogram, which achieve optimal or near-optimal worst-case error. Bun *et al.* [15] further study the problem of finding heavy-hitters by strengthening existing lower bounds on the error to incorporate the failure probability. To provide heavy hitters estimation over set-valued data, Qin *et al.* [16] propose LDPMiner, a two-phase LDP approach, in which the data collector gathers a candidate set of heavy hitters in the first phase and refines the candidate set in the second phase. Wang *et al.* [17] propose the padding-and-sampling-based frequency oracle (PSFO) protocol that enables users to sample one item from a given set to report. Based on the PSFO protocol, they design two algorithms to identify heavy hitters and frequent itemsets, respectively. To collect top-k frequent new terms from users, Wang *et al.* [18] propose PrivTrie, which uses an adaptive algorithm to build a trie under LDP.

There are some studies for releasing marginals under LDP. Nguyen *et al.* [6] propose Harmony to support mean and frequency estimates and complex machine learning tasks on multi-attribute data containing both numerical and categorical attributes. Ren *et al.* [19] present a scheme to estimate the joint distribution for multiple variables from multi-attribute data under LDP. Based on the Hadamard transformation, Cormode *et al.* [20] provide a set of effective methods for releasing marginal statistics of multi-attribute data. Zhang *et al.* [21] propose CALM, Consistent Adaptive Local Marginal, which takes advantage of the careful challenge analysis and performs consistently better than existing methods.

There also exist some works providing the theoretical understanding of the tradeoff between utility and LDP. Duchi *et al.* [22] provide information theoretic converse techniques for deriving minimax bounds under LDP. Kairouz *et al.* [23] develop a family of extremal privatization mechanisms, called staircase mechanisms, to maximize the utility of f-divergence utility functions under LDP. Gursoy *et al.* [24] propose a variant of LDP, namely Condensed Local Differential Privacy (CLDP), for utility-aware and privacy-preserving data collection.

Besides, LDP has been applied to the collection of other types of data such as locations [25], key-value data [26], [27], graphs [28], [29], [30], streaming data [31], [32], [33], and machine learning tasks such as federated learning [34] and deep learning [35], [36]. However, we are not aware of any existing methods are suitable for collecting ranking data.

This paper extends our previous work [37] and differs from [37] in the following three aspects. First, to precisely estimate the required distributions used to learn the RI model's structure, we design a new method called LADE, and a new version of Rule I for ranking transformation. Experimental results show that LADE can significantly improve the accuracy of the collected distributions as compared to the previous method proposed in [37] (referred to as NAIVE in this paper). Second, to further confirm the effectiveness of the Lasso regression model [38] adopted in LADE, we empirically compare Lasso regression with another two canonical regression models, i.e., Stepwise regression and Ridge regression. Third, we evaluate the performance of our SAFARI approach on two real datasets, and conduct new experiments to study the impact of the structure parameters of the RI model.

3 PRELIMINARIES AND PROBLEM STATEMENT

In this section, we first present three important concepts including preference ranking, riffle independent model and local differential privacy. Then we give the problem statement.

3.1 Preference Ranking

Given an item set $\mathcal{X} = \{x_1, x_2, \ldots, x_d\}$, a ranking of $\mathcal{X}$ is an ordered list which contains all d items in $\mathcal{X}$. A typical kind of ranking is preference ranking, which places different items into a total order based on individual opinions about their relative quality. We denote a preference ranking by $\sigma = \langle \sigma(1), \sigma(2), \ldots, \sigma(d) \rangle$ where $\sigma(j) = x_k$ means that x_k's rank under σ is j. We define the function $\sigma^{-1}(x_k) = j$ to retrieve x_k's rank j under σ. Without loss of generality, each user's most favorite item is assigned rank 1 while the least favorite item is assigned rank d.

A typical property of a ranking is the *mutual exclusivity*, which guarantees that two different items have different ranks in a given ranking. For clarity, we provide a running example throughout this paper below.

Example 1. *Consider the item set $\mathcal{X}$ that consists of five food items:*

$$\mathcal{X} = \{Beer, Cola, Fanta, Potato, Whisky\}.$$

We use $\mathbf{B}$, $\mathbf{C}$, $\mathbf{F}$, $\mathbf{P}$ *and* $\mathbf{W}$ *to represent these five food items, respectively. The user u_0's ranking $\sigma_0 = \langle \mathbf{W}, \mathbf{B}, \mathbf{C}, \mathbf{F}, \mathbf{P} \rangle$ means that Whisky is ranked first, Beer is ranked second and so on.*

3.2 Riffle Independent Model

Riffle independent (RI) model [4], [5] is a probabilistic graphical model proposed for modeling rankings. Unlike conditional independence models (e.g., Bayesian networks), it can effectively enforce mutual exclusivity among rankings. Initially, we introduce several important concepts in the RI model.

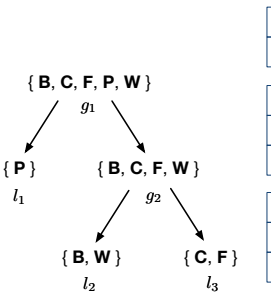
- *Full ranking*: A ranking that contains all items in $\mathcal{X}$.
- *Relative ranking*: A ranking that contains items in a subset of $\mathcal{X}$.
- *Interleaving*: A ranking that describes how to interleave two relative rankings.
- *Absolute rank*: An item's rank in a full ranking.
- *Relative rank*: An item's rank in a relative ranking.

The main idea of RI model is to hierarchically split the original item set into disjoint subsets whose items have strong correlations, and interleave the relative rankings of those subsets to a full ranking. Formally, the construction of RI model includes the following two phases.

1. Structure learning. In this phase, it constructs the structure of the RI model, named K-thin chain. A K-thin chain denoted by $\mathcal{T}$ is a binary tree, which consists of leaf item sets and internal item sets. We denote leaf item sets in $\mathcal{T}$ as $\mathcal{L} = \{l_j | 1 \leq j \leq |\mathcal{L}|\}$ and internal item sets as $\mathcal{G} = \{g_j | 1 \leq j \leq |\mathcal{G}|\}$. The size of each leaf item set is no more than the constant K.

Definition 1 (Tripletwise Mutual Information)**.** *Given any triplet of distinct items, $(x_{k_1}, x_{k_2}, x_{k_3})$, its tripletwise mutual information is:*

$$\mathcal{I}(x_{k_1}, x_{k_2}, x_{k_3}) = \sum_{\varphi_{k_1}} \sum_{\lambda_{k_2,k_3}} \mathsf{Pr}\left[\varphi_{k_1}, \lambda_{k_2,k_3}\right] \log \frac{\mathsf{Pr}\left[\varphi_{k_1}, \lambda_{k_2,k_3}\right]}{\mathsf{Pr}\left[\varphi_{k_1}\right] \mathsf{Pr}\left[\lambda_{k_2,k_3}\right]},$$



(a) 2-thin chain.

σ^{l_1}	$\Pr[\sigma^{l_1}]$
$\langle \mathbf{P} \rangle$	1

σ^{l_2}	$\Pr[\sigma^{l_2}]$
$\langle \mathbf{B}, \mathbf{W} \rangle$	0.7
$\langle \mathbf{W}, \mathbf{B} \rangle$	0.3

σ^{l_3}	$\Pr[\sigma^{l_3}]$
$\langle \mathbf{C}, \mathbf{F} \rangle$	0.1
$\langle \mathbf{F}, \mathbf{C} \rangle$	0.9

σ^{g_1}	$\Pr[\sigma^{g_1}]$
$\langle l_1, g_2, g_2, g_2, g_2 \rangle$	0.2
$\vdots$	$\vdots$
$\langle g_2, g_2, g_2, g_2, l_1 \rangle$	0.1

σ^{g_2}	$\Pr[\sigma^{g_2}]$
$\langle l_2, l_2, l_3, l_3 \rangle$	0.3
$\vdots$	$\vdots$
$\langle l_3, l_3, l_2, l_2 \rangle$	0.1

(b) Distributions.

Fig. 1: Riffle independent model over $\mathcal{X}$

where $\varphi_{k_1} = \sigma^{-1}(x_{k_1})$ and $\lambda_{k_2,k_3} \in \{1, 2\}$ is a variable. In particular, $\lambda_{k_2,k_3} = 1$ represents $\sigma^{-1}(x_{k_2}) < \sigma^{-1}(x_{k_3})$ while $\lambda_{k_2,k_3} = 2$ represents $\sigma^{-1}(x_{k_2}) > \sigma^{-1}(x_{k_3})$.

Then, it splits the original item set in the following manner. It splits the original item set into two disjoint subsets by the Anchors method [5], in which the item set is reasonably partitioned by optimizing an objective function of tripletwise mutual information. The smaller subset consists of no more than K items. The larger subset continues to be split in the same way. The process continues until either of the two subsets consists of no more than K items.

2. Parameter learning. In this phase, it computes the distributions over item sets in the K-thin chain $\mathcal{T}$. The distributions over leaf item sets are relative ranking distributions, while those over internal item sets are interleaving distributions.

Example 2. *Continuing with Example 1, Fig. 1 shows an example of RI model over the item set $\mathcal{X}$. Fig. 1(a) is a 2-thin chain. Specifically, l_1, l_2 and l_3 are leaf item sets; g_1 and g_2 are internal item sets. The original item set $g_1 = \{\mathbf{B}, \mathbf{C}, \mathbf{F}, \mathbf{P}, \mathbf{W}\}$ ("food") is first split into two disjoint subsets, i.e., $l_1 = \{\mathbf{P}\}$ ("vegetable") and $g_2 = \{\mathbf{B}, \mathbf{C}, \mathbf{F}, \mathbf{W}\}$ ("drink"). Since the size of g_2 is more than K (i.e., 2), it is further split into $l_2 = \{\mathbf{B}, \mathbf{W}\}$ ("liquor") and $l_3 = \{\mathbf{C}, \mathbf{F}\}$ ("sodas").*

Fig. 1(b) shows the distributions on the 2-thin chain. The relative ranking distributions are shown on the left side, while the interleaving distributions are shown on the right side. We use l_2, l_3 and g_2 to explain the above distributions. The relative ranking distribution over $l_2 = \{\mathbf{B}, \mathbf{W}\}$ (denoted by $\mathsf{Pr}\left[\sigma^{l_2}\right]$) describes the probabilities about possible relative rankings of $\{\mathbf{B}, \mathbf{W}\}$. $\sigma^{l_2} = \langle \mathbf{W}, \mathbf{B} \rangle$ means that the relative ranking of l_2 is $\langle \mathbf{W}, \mathbf{B} \rangle$ under σ. Like l_2, the relative ranking distribution over $l_3 = \{\mathbf{C}, \mathbf{F}\}$ (denoted by $\mathsf{Pr}\left[\sigma^{l_3}\right]$) describes the probabilities about possible relative rankings of $\{\mathbf{C}, \mathbf{F}\}$. $\sigma^{l_3} = \langle \mathbf{C}, \mathbf{F} \rangle$ means that the relative ranking of l_3 is $\langle \mathbf{C}, \mathbf{F} \rangle$ under σ. The interleaving distribution over g_2 (denoted by $\mathsf{Pr}\left[\sigma^{g_2}\right]$) describes how to interleave the relative rankings generated from l_2 and l_3 into a ranking of $\{\mathbf{B}, \mathbf{C}, \mathbf{F}, \mathbf{W}\}$. Interleaving $\sigma^{g_2} = \langle l_2, l_2, l_3, l_3 \rangle$ means that the two items of l_2 have higher ranks than those of l_3. With $\sigma^{g_2} = \langle l_2, l_2, l_3, l_3 \rangle$, we can interleave two relative rankings, e.g., $\sigma^{l_2} = \langle \mathbf{W}, \mathbf{B} \rangle$ and $\sigma^{l_3} = \langle \mathbf{C}, \mathbf{F} \rangle$, into a new ranking $\langle \mathbf{W}, \mathbf{B}, \mathbf{C}, \mathbf{F} \rangle$.

According to the RI model, we can obtain the probability of each ranking. For instance, the probability of ranking

TABLE 1: List of Frequent Notations

Notation	Description
$\mathcal{X}$	The item set
d	The total number of items in $\mathcal{X}$
n	The total number of users
u_i	The i-th user
σ_i	The ranking of u_i
t_i	The transformed tuple of u_i
$\mathcal{A}$	The attribute set
A_j	The j-th attribute in $\mathcal{A}$
$dom(A_j)$	The domain of attribute A_j
$t_i[A_j]$	The value of A_j in t_i
$I(t_i[A_j])$	The index of $t_i[A_j]$ in $dom(A_j)$
$\mathcal{T}$	The K-thin chain
K	The maximum size of leaf item sets in $\mathcal{T}$
l_j	The leaf item set
g_j	The internal item set
$\sigma_i^{l_j}$	The relative ranking of l_j under σ_i
$\sigma_i^{g_j}$	The interleaving of g_j under σ_i
$Pr\left[\sigma_i^{l_j}\right]$	The relative ranking distribution over l_j under σ_i
$Pr\left[\sigma_i^{g_j}\right]$	The interleaving distribution over g_j under σ_i

$\sigma_0 = \langle \mathbf{W}, \mathbf{B}, \mathbf{C}, \mathbf{F}, \mathbf{P} \rangle$ *is*

$$\begin{aligned}\mathsf{Pr}\left[\sigma_0\right] &= \mathsf{Pr}\left[\sigma_0^{g_1}\right] \cdot \mathsf{Pr}\left[\sigma_0^{g_2}\right] \cdot \mathsf{Pr}\left[\sigma_0^{l_1}\right] \cdot \mathsf{Pr}\left[\sigma_0^{l_2}\right] \cdot \mathsf{Pr}\left[\sigma_0^{l_3}\right] \\ &= \mathsf{Pr}\left[g_2, g_2, g_2, g_2, l_1\right] \cdot \mathsf{Pr}\left[\mathbf{P}\right] \cdot \mathsf{Pr}\left[l_2, l_2, l_3, l_3\right] \\ &\quad \cdot \mathsf{Pr}\left[\mathbf{W}, \mathbf{B}\right] \cdot \mathsf{Pr}\left[\mathbf{C}, \mathbf{F}\right] \\ &= 0.1 \times 1 \times 0.3 \times 0.3 \times 0.1 \\ &= 0.0009.\end{aligned}$$

3.3 Local Differential Privacy

Local differential privacy (LDP) [7] is defined as follows.

Definition 2 (ε-LDP)**.** *Given a privacy budget ε, a randomized algorithm $\mathcal{R}$ satisfies ε-LDP, if for any two tuples t and $t' \in dom(\mathcal{R})$, and for any possible output $\tilde{t} \in Range(\mathcal{R})$, we have*

$$\mathsf{Pr}\left[\mathcal{R}(t) = \tilde{t}\right] \leq e^{\varepsilon} \cdot \mathsf{Pr}\left[\mathcal{R}(t') = \tilde{t}\right],$$

where the probability space is over the coin flips of $\mathcal{R}$.

For a sequence of locally differentially private algorithms, the sequential composition property [39] guarantees the overall privacy.

Theorem 1 (Sequential Composition)**.** *Given q random algorithms $\mathcal{R}_i\,(1 \leq i \leq q)$, each of which satisfies ε_i-LDP, the sequence of $\mathcal{R}_i$ satisfies $\left(\sum_{i=1}^{q} \varepsilon_i\right)$-LDP.*

3.4 Problem Statement

Consider an item set $\mathcal{X} = \{x_1, x_2, \ldots, x_d\}$. Let n be the total number of users, and $u_i\,(1 \leq i \leq n)$ denote the i-th user. Each user u_i possesses a preference ranking σ_i of full or partial items in $\mathcal{X}$. Our goal is to design an approach to enable an untrusted data collector to collect the distribution of rankings from the n users and generate a synthetic ranking dataset that approximates the user's original rankings while satisfying LDP. In particular, in this paper we focus on the problem of collecting the full preference rankings. Meanwhile, the problem of collecting partial preference rankings can also be supported by inserting dummy tuples to pad the size of partial preference ranking to the number of all items.

In the collection process, for any user, an adversary could be the untrusted data collector, another participating user or an outside attacker. In general, the objective of an adversary is to learn a user's true ranking, which might be very sensitive in many applications. To prevent the leakage of each user's privacy, we aim to enforce LDP for each user.

In ε-LDP, since random perturbation is done on the user side, each user can set a different privacy budget ε according to his/her own privacy requirement. In this paper, we assume that all users use the same ε for ease of presentation and analysis.

Table 1 summarizes notations that will be frequently used in this paper.

4 Baseline Approaches

In this section, we design four baseline approaches based on four existing LDP methods, including GRR [10], Harmony [1], CALM [9] and OLH [10]. In particular, in these four approaches, we consider each user's ranking as a categorical value in a domain that consists of all possible rankings. In the following, we will first show the details of these four approaches, respectively, and then point out their drawbacks.

4.1 GRR Based Approach

In the collection process, each user u_i first samples an index that corresponds to his/her ranking σ_i with probability $\frac{e^{\varepsilon}}{e^{\varepsilon}+d!-1}$ while any other possible ranking with probability $\frac{1}{e^{\varepsilon}+d!-1}$, and then reports the sampled index to the data collector. After receiving all users' perturbed indexes, the data collector computes an unbiased estimate of the frequency of each of the $d!$ rankings.

4.2 Harmony Based Approach

In the collection process, each user u_i first represents his/her ranking σ_i using a length-$d!$ bit vector, whose bits are all zeros except the bit corresponding to σ_i which is one. Then, each user multiplies his/her length-$d!$ bit vector by a $d! \times m$ matrix Φ (where $m = O(n)$) to generate a length-m vector. In particular, each element in Φ is selected randomly from two possible values: $\frac{1}{\sqrt{m}}$ and $-\frac{1}{\sqrt{m}}$. Finally, each user selects a value in his/her length-m vector randomly and releases it using RR to the data collector. After receiving all users' perturbed values, the data collector computes the frequency of each of the $d!$ rankings by combining these values with the matrix Φ.

4.3 OLH Based Approach

In the collection process, each user u_i first invokes a hash function H, which is randomly chosen from the universal hash function family, to hash his/her ranking σ_i into a new value in a smaller domain $\{1, \ldots, \lceil e^{\varepsilon} + 1 \rceil\}$. Then, each u_i uses GRR to perturb the hashed value. Finally, each u_i sends the perturbed hashed value and the hash function H to the data collector. After receiving all users' perturbed hashed values and hash functions, the data collector computes an unbiased estimate of the frequency of each of the $d!$ rankings.

4.4 CALM Based Approach

CALM [21] is the state-of-the-art method for multi-way marginal release under LDP. In the collection process, each user u_i first represents the relative ranking between each pair of items using the 2-D marginals and reporting these 2-D marginals via OLH. Then, the data collector reconstruct the d-D marginals via maximum entropy model based on all collected 2-D marginals. Finally, the data collector converts the d-D marginals into the frequency of each of the $d!$ rankings.

4.5 Drawbacks of Baselines

Obviously, all these four approaches have serious drawbacks. For the GRR based approach, since the possibility that each user samples the index that corresponds to his/her true ranking is $\frac{e^{\varepsilon}}{e^{\varepsilon}+d!-1}$, which is too small, it is difficult for the data collector to gain valuable information. For the Harmony based approach, although its communication cost between each user and the data collector is $O(1)$, the scale of the random projection matrix Φ is huge, which introduces considerable noise, and thus leads to low data utility. For the OLH based approach, its communication cost between each user and the data collector is also $O(1)$. However, the use of hash functions inevitably causes information loss, which leads to low data utility as well. For the CALM based approach, its estimation error caused by maximum entropy model is huge, which leads to low data utility.

5 SAFARI

In this section, we present our approach SAFARI for collecting preference rankings under LDP.

5.1 Overview

As shown in Section 4, all the baseline approaches suffer poor data utility. The root cause is the large domain of rankings. Even for a moderate number d of items, the total number of possible rankings is massive (i.e., $d!$). Thus directly collecting the users' rankings results in excessive noise.

To address this problem, we propose a novel approach, called **S**ampling RA**n**domizer **F**or Multiple **A**ttributes with **R**iffle **I**ndependent Model (SAFARI), for collecting rankings under LDP. Its main idea is to collect a set of distributions over small domains which are carefully chosen based on the RI model to approximate the overall distribution of users' rankings, and then generate a synthetic ranking dataset from the obtained distributions. Since SAFARI works on small domains instead of a large domain, it can significantly reduce the magnitude of added noise. Specifically, SAFARI consists of the following phases:

Phase 1. Each user transforms his/her ranking according to a transformation rule, namely Rule I, to provide the information about the necessary distributions used to estimate all the distributions required to compute the tripletwise mutual information. In particular, such an estimation process is performed via a Lasso based distribution estimation method called LADE. The details of Rule I and LADE are shown in Section 5.2.

Phase 2. The data collector first invokes a new locally differentially private method called SAFA, using privacy budget $\varepsilon/2$, to collect those distributions from users' transformed data generated in Phase 1. The details of SAFA are described in Section 5.4. Then, from the collected distributions, the data collector employs LADE to estimate all those distributions required to compute the tripletwise mutual information. After that, the data collector computes the tripletwise mutual information of all triplets of distinct items, and constructs a K-thin chain $\mathcal{T}$ (K is a small integer set by the data collector). Finally, the data collector broadcasts $\mathcal{T}$ to all users.

Phase 3. Each user transforms his/her ranking according to another transformation rule, namely Rule II, to provide the information about the distributions of relative rankings and interleavings over the constructed K-thin chain $\mathcal{T}$. The details of Rule II are given in Section 5.3.

Phase 4. The data collector invokes SAFA, using privacy budget $\varepsilon/2$, to collect the relative ranking distributions and interleaving distributions over $\mathcal{T}$ from users' transformed data generated in Phase 3.

Phase 5. The data collector synthesizes n rankings independently from the constructed RI model to generate a ranking dataset. To efficiently synthesize a ranking from the RI model, the data collector first samples a relative ranking for each leaf item set in $\mathcal{T}$, and then samples an interleaving for each internal item set in $\mathcal{T}$. Finally, with sampled interleavings, the data collector interleaves these relative rankings to a full ranking according to $\mathcal{T}$ in a bottom-up manner.

In the following, we first show the details of LADE and Rule I in Section 5.2, and Rule II in Section 5.3. Then we elaborate the SAFA method used in SAFARI in Section 5.4. Finally, we give a comprehensive analysis of SAFARI in Section 5.5.

5.2 Design of LADE and Rule I

According to the definition of tripletwise mutual information shown in Section 3.2, to compute the tripletwise mutual information for every possible triplet $(x_{k_1}, x_{k_2}, x_{k_3})$ of distinct items, the data collector needs to collect three types of distributions:

$$\mathcal{S}_1 = \{\mathsf{Pr}[\varphi_{k_1}] \,|1 \le k_1 \le d\};$$
$$\mathcal{S}_2 = \{\mathsf{Pr}[\lambda_{k_2,k_3}] \,|1 \le k_2, k_3 \le d\};$$
$$\mathcal{S}_3 = \{\mathsf{Pr}[\varphi_{k_1}, \lambda_{k_2,k_3}] \,|1 \le k_1, k_2, k_3 \le d\}.$$

To achieve this task, a simple idea is to let each user transform his/her ranking to provide the information about all the three types of distributions. Specifically, each user transforms his/her ranking into a tuple that contains $d + \binom{d}{2} + d \cdot \binom{d-1}{2}$ attributes, each of which corresponds to one distribution in $\mathcal{S}_1 \bigcup \mathcal{S}_2 \bigcup \mathcal{S}_3$.

However, such a transformation will make the tuple contain a large number of attributes and result in plenty of redundant information in the transformed data due to the natural dependencies among $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$, i.e.,

$$\mathsf{Pr}[\varphi_{k_1}] = \sum_{\lambda_{k_2,k_3}} \mathsf{Pr}[\varphi_{k_1}, \lambda_{k_2,k_3}],$$
$$\mathsf{Pr}[\lambda_{k_2,k_3}] = \sum_{\varphi_{k_1}} \mathsf{Pr}[\varphi_{k_1}, \lambda_{k_2,k_3}]. \quad (1)$$

Under LDP, it will incur the curse of dimensionality, and thus lead to excessive noise in the collected distributions in $\mathcal{S}_1 \bigcup \mathcal{S}_2 \bigcup \mathcal{S}_3$.

To obtain more accurate distributions in $\mathcal{S}_1 \bigcup \mathcal{S}_2 \bigcup \mathcal{S}_3$, we propose to only collect necessary distributions, and then use them to estimate those in $\mathcal{S}_1 \bigcup \mathcal{S}_2 \bigcup \mathcal{S}_3$ in an effective way. Following this idea, we first design two baseline methods, namely NAIVE and CALM, and then put forward our LADE method. To instruct

users to transform their rankings to provide the information about the necessary distributions for these three methods, we further design three transformation rules, namely Rule I^N, Rule I^C and Rule I, which correspond to NAIVE, CALM and LADE, respectively. The details of these methods and rules are described in the following.

5.2.1 NAIVE and Rule I^N

The NAIVE method is inspired by the natural dependencies among $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$, as shown in Equation (1). In NAIVE, the data collector only collects the distributions in $\mathcal{S}_3$, and then derives the distributions in $\mathcal{S}_1 \bigcup \mathcal{S}_2$ by integrating the corresponding distributions in $\mathcal{S}_3$ according to the dependencies.

For NAIVE, we design Rule I^N to instruct each user to transform his/her ranking to provide the information about the distributions in $\mathcal{S}_3$. Since there are $d \cdot \binom{d-1}{2}$ different distributions in $\mathcal{S}_3$, each user only needs to transform his/her ranking into a tuple that contains $d \cdot \binom{d-1}{2}$ attributes, each of which corresponds to one distribution in $\mathcal{S}_3$. The details of Rule I^N is given below.

Rule I^N. To start with, each user u_i transforms his/her ranking σ_i into a tuple t_i that contains all the attributes in an attribute set

$$\mathcal{A} = \{A_j | 1 \leq j \leq d \cdot \binom{d-1}{2}\}.$$

Here each attribute A_j in $\mathcal{A}$ corresponds to a triplet $(x_{k_1}, x_{k_2}, x_{k_3})$, where k_2 is less than k_3. The domain of A_j is

$$\begin{aligned} dom(A_j) &= \{1, 2, \ldots, d\} \times \{1, 2\} \\ &= \{(1,1), (1,2), \ldots, (d,1), (d,2)\}, \end{aligned}$$

which consists of $2d$ possible 2-tuples. In a 2-tuple, the first value represents the rank of x_{k_1} and the second value represents the relative rank relation between x_{k_2} and x_{k_3}. More concretely, for the second value, 1 represents $\sigma^{-1}(x_{k_2}) < \sigma^{-1}(x_{k_3})$ while 2 represents $\sigma^{-1}(x_{k_2}) > \sigma^{-1}(x_{k_3})$. Then, for each attribute A_j in $\mathcal{A}$, each user u_i assigns $t_i[A_j]$'s value that is derived from σ_i.

Example 3. *We illustrate how to transform a ranking according to Rule I^N. Given $\mathcal{X} = \{\mathbf{B}, \mathbf{C}, \mathbf{F}, \mathbf{P}, \mathbf{W}\}$, we have $\mathcal{A} = \{A_j | 1 \leq j \leq 30\}$. Assume that the attribute A_1 corresponds to the triplet $(\mathbf{B}, \mathbf{C}, \mathbf{F})$. For $\sigma_0 = \langle \mathbf{W}, \mathbf{B}, \mathbf{C}, \mathbf{F}, \mathbf{P} \rangle$, since the rank of* **B** *is 2 and the rank of* **C** *is higher than that of* **F**, *after user u_0 transforms σ_0 into a tuple t_0 by Rule I^N, we have $t_0[A_1] = (2, 1)$. The values of other attributes can be derived in a similar way.*

However, when d is relatively large, it will also lead to a considerable amount of added noise in the collected distributions in $\mathcal{S}_3$ under LDP due to the curse of dimensionality. Moreover, since the integration process causes the accumulation of noise in S_3, it will result in a large amount of added noise in $\mathcal{S}_1$ and $\mathcal{S}_2$.

5.2.2 CALM and Rule I^C

As the state-of-the-art method for marginal release method under LDP, CALM can also be applied to obtain the distributions in $\mathcal{S}_1 \bigcup \mathcal{S}_2 \bigcup \mathcal{S}_3$. Specifically, to obtain the distributions in $\mathcal{S}_1 \bigcup \mathcal{S}_2 \bigcup \mathcal{S}_3$, the data collector first employs CALM to collect all 2-D marginals over users' rankings, where the set of 2-D marginals is denoted as

$$\mathcal{S}_4 = \{\mathsf{Pr}[\varphi_{k_1}, \varphi_{k_2}] | 1 \leq k_1, k_2 \leq d\}.$$

Then, using $\mathcal{S}_4$, the data collector reconstructs all 3-D marginals

$$\mathcal{S}_5 = \{\mathsf{Pr}[\varphi_{k_1}, \varphi_{k_2}, \varphi_{k_3}] | 1 \leq k_1, k_2, k_3 \leq d\}.$$

Next, from $\mathcal{S}_4$, the data collector derives the distributions in $\mathcal{S}_1 \bigcup \mathcal{S}_2$ by integrating the corresponding marginals in $\mathcal{S}_4$. Finally, the data collector derives $\mathcal{S}_3$ from $\mathcal{S}_5$ in a similar way.

For CALM, we design Rule I^C to instruct each user to transform his/her ranking to provide the information about the 2-D marginals. In Rule I^C, each user transforms his/her ranking into a tuple that contains $\binom{d}{2}$ attributes, each of which corresponds to one of the total $\binom{d}{2}$ 2-D marginals. The details of Rule I^C is shown as follows.

Rule I^C. Initially, each user u_i transforms his/her ranking σ_i into a tuple t_i that contains all the attributes in an attribute set $\mathcal{A} = \{A_j | 1 \leq j \leq \binom{d}{2}\}$. Here each attribute A_j in $\mathcal{A}$ corresponds to an item pair (x_{k_1}, x_{k_2}). The domain of A_j is

$$\begin{aligned} dom(A_j) = &\{1, 2, \ldots, d\} \times \{1, 2, \ldots, d\} \\ &- \{(1,1), (2,2), \ldots, (d,d)\}, \end{aligned}$$

which consists of $d \cdot (d-1)$ possible values, representing the possible pair of the absolute ranks of x_{k_1} and x_{k_2}. Then, for each attribute A_j in $\mathcal{A}$, each user u_i assigns $t_i[A_j]$'s value that is derived from σ_i.

Example 4. *We illustrate how to transform a ranking according to Rule I^C. Given $\mathcal{X} = \{\mathbf{B}, \mathbf{C}, \mathbf{F}, \mathbf{P}, \mathbf{W}\}$, we have $\mathcal{A} = \{A_j | 1 \leq j \leq 10\}$. Assume that the attribute A_1 corresponds to the item pair $(\mathbf{B}, \mathbf{C})$. For $\sigma_0 = \langle \mathbf{W}, \mathbf{B}, \mathbf{C}, \mathbf{F}, \mathbf{P} \rangle$, since the rank of* **B** *is 2 and the rank of* **C** *is 3, after user u_0 transforms σ_0 into a tuple t_0 by Rule I^C, we have $t_0[A_1] = (2, 3)$. The values of other attributes can be derived in a similar way.*

However, similar to NAIVE, when d is relatively large, it will also produce excessive noise in the distributions in $\mathcal{S}_1 \bigcup \mathcal{S}_2 \bigcup \mathcal{S}_3$, since the integration process leads to the accumulation of noise in $\mathcal{S}_4$ and $\mathcal{S}_5$.

5.2.3 LADE and Rule I

To address the issues raised by NAIVE and CALM, i.e., the curse of the dimensionality and the accumulation of LDP noise, we design a **L**asso b**A**sed **D**istribution **E**stimation (LADE) method, whose main idea is to only collect the distributions in $\mathcal{S}_1$, and then make use of the Lasso regression model to estimate the distributions in $\mathcal{S}_2$ and $\mathcal{S}_3$. In particular, such an estimation corresponds to a sparse linear regression problem. Compared with other regression models, the Lasso regression model [38] only collects S_1 from local preference rankings, which can avoid the curse of the dimensionality and the accumulation of LDP noise. Moreover, LADE will not lead to excessive estimation error since the probability space of S_2 and S_3 is much small than ordinary 2-way marginals and 3-way marginals. All indicate that LADE can strike a balance between LDP noise and estimation error, resulting in a better data utility compared with NAIVE and CALM.

LADE only collects S_1 from local preference rankings, which can avoid the curse of the dimensionality and the accumulation of LDP noise. Moreover, LADE will not lead to excessive estimation error since the probability space of S_2 and S_3 is much small than ordinary 2-way marginals and 3-way marginals. All indicate that LADE can strike a balance between LDP noise and estimation error, resulting in a better data utility compared with NAIVE and CALM.

For LADE, we design Rule I to instruct each user to transform his/her ranking to provide the information about the distributions in $\mathcal{S}_1$. Apparently, each user only needs to transform his/her ranking into a tuple that contains d attributes, each of which corresponds to one distribution in $\mathcal{S}_1$.

In LADE, since Rule I avoids the curse of dimensionality, the accuracy of the collected distributions in $\mathcal{S}_1$ can be guaranteed under LDP. In addition, using the Lasso regression model to estimate the distributions in $\mathcal{S}_2 \bigcup \mathcal{S}_3$ from $\mathcal{S}_1$ evades the accumulation of LDP noise. Therefore, the accuracy of the estimated $\mathcal{S}_2$ and $\mathcal{S}_3$ from $\mathcal{S}_1$ can also be maintained.

In the following, we explain the details of Rule I and how to estimate the distributions in $\mathcal{S}_2$ and $\mathcal{S}_3$ by LADE.

Rule I. Initially, each user u_i transforms his/her ranking σ_i into a tuple t_i that contains all the attributes in an attribute set $\mathcal{A} = \{A_j | 1 \leq j \leq d\}$. Here each attribute A_j in $\mathcal{A}$ corresponds to an item x_{k_1}. The domain of A_j is $dom(A_j) = \{1, 2, \ldots, d\}$ consisting of d possible values, which represent the possible absolute ranks of x_{k_1}. Then, for each attribute A_j in $\mathcal{A}$, each user u_i assigns $t_i[A_j]$'s value that is derived from σ_i.

Example 5. *We illustrate how to transform a ranking according to Rule I. Given $\mathcal{X} = \{\mathbf{B}, \mathbf{C}, \mathbf{F}, \mathbf{P}, \mathbf{W}\}$, we have $\mathcal{A} = \{A_j | 1 \leq j \leq 5\}$. Assume that the attribute A_1 corresponds to the item $\mathbf{B}$ and the attribute A_2 corresponds to the item $\mathbf{C}$. For $\sigma_0 = \langle \mathbf{W}, \mathbf{B}, \mathbf{C}, \mathbf{F}, \mathbf{P} \rangle$, since the rank of $\mathbf{B}$ is 2 and the rank of $\mathbf{C}$ is 3, after user u_0 transforms σ_0 into a tuple t_0 by Rule I, we have $t_0[A_1] = 2$ and $t_0[A_2] = 3$. The values of other attributes can be derived in a similar way.*

Distribution Estimation by LADE. Based on the collected distributions in $\mathcal{S}_1$, the data collector estimates the distributions in $\mathcal{S}_2$ and $\mathcal{S}_3$ in the following manner.

To start with, the data collector estimates the distributions in $\mathcal{S}_2$ based on the collected distributions in $\mathcal{S}_1$. Specifically, for each distribution $\mathsf{Pr}[\lambda_{k_2,k_3}]$ in $\mathcal{S}_2$, the data collector constructs a Lasso regression model $\mathbf{y}_{k_2,k_3} = \mathbf{M}_{k_2,k_3} \cdot \mathbf{p}_{k_2,k_3}$, where

1) $\mathbf{y}_{k_2,k_3}$ is a length-$2d$ column vector that reserves the distributions $\mathsf{Pr}[\varphi_{k_2}]$ and $\mathsf{Pr}[\varphi_{k_3}]$;
2) $\mathbf{M}_{k_2,k_3}$ is a $2d \times d(d-1)$ binary matrix;
3) $\mathbf{p}_{k_2,k_3}$ is a length-$d(d-1)$ column vector that reserves the joint distribution $\mathsf{Pr}[\varphi_{k_2}, \varphi_{k_3}]$.

By fitting the Lasso regression model via Least angle regression [40], the data collector can estimate $\mathbf{p}_{k_2,k_3}$ and obtain the joint distribution $\mathsf{Pr}[\varphi_{k_2}, \varphi_{k_3}]$. From $\mathsf{Pr}[\varphi_{k_2}, \varphi_{k_3}]$, the data collector can easily derive the distribution $\mathsf{Pr}[\lambda_{k_2,k_3}]$.

Then, the data collector estimates the distributions in $\mathcal{S}_3$ based on the obtained distributions in $\mathcal{S}_1$ and $\mathcal{S}_2$. Specifically, for each distribution $\mathsf{Pr}[\varphi_{k_1}, \lambda_{k_2,k_3}]$ in $\mathcal{S}_3$, the data collector constructs a Lasso regression model $\mathbf{y}_{k_1,k_2,k_3} = \mathbf{M}_{k_1,k_2,k_3} \cdot \mathbf{p}_{k_1,k_2,k_3}$, where

1) $\mathbf{y}_{k_1,k_2,k_3}$ is a length-$(d+2)$ column vector that reserves the distributions $\mathsf{Pr}[\varphi_{k_1}]$ and $\mathsf{Pr}[\lambda_{k_2,k_3}]$;
2) $\mathbf{M}_{k_1,k_2,k_3}$ is a $(d+2) \times 2d$ binary matrix;
3) $\mathbf{p}_{k_1,k_2,k_3}$ is a length-$2d$ column vector that reserves the distribution $\mathsf{Pr}[\varphi_{k_1}, \lambda_{k_2,k_3}]$.

Similarly, by fitting the Lasso regression model via Least angle regression, the data collector can estimate $\mathbf{p}_{k_1,k_2,k_3}$ and obtain the distribution $\mathsf{Pr}[\varphi_{k_1}, \lambda_{k_2,k_3}]$.

Here, we illustrate how to estimate a distribution in $\mathcal{S}_2$ and a distribution in $\mathcal{S}_3$ by Example 6.

Example 6. *Continuing with Example 5, we assume that the data collector has collected the distributions in $\mathcal{S}_1$. To obtain the distribution $\mathsf{Pr}[\lambda_{1,2}]$ in $\mathcal{S}_2$, the data collector constructs the Lasso regression model $\mathbf{y}_{1,2} = \mathbf{M}_{1,2} \cdot \mathbf{p}_{1,2}$, where*

$$\mathbf{M}_{1,2} = \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\mathbf{p}_{1,2} = \begin{pmatrix} \mathsf{Pr}[\varphi_1 = 1, \varphi_2 = 2] \\ \vdots \\ \mathsf{Pr}[\varphi_1 = 1, \varphi_2 = 5] \\ \vdots \\ \mathsf{Pr}[\varphi_1 = 5, \varphi_2 = 1] \\ \vdots \\ \mathsf{Pr}[\varphi_1 = 5, \varphi_2 = 4] \end{pmatrix}, \mathbf{y}_{1,2} = \begin{pmatrix} \mathsf{Pr}[\varphi_1 = 1] \\ \vdots \\ \mathsf{Pr}[\varphi_1 = 5] \\ \mathsf{Pr}[\varphi_2 = 1] \\ \vdots \\ \mathsf{Pr}[\varphi_2 = 5] \end{pmatrix}.$$

In particular, $\mathbf{M}_{1,2}$ is a 10×20 binary matrix, in which only four values in each row are 1. $\mathbf{p}_{1,2}$ is a length-20 column vector. Due to the mutual exclusivity among rankings, $\mathbf{p}_{1,2}$ excludes the probabilities $\mathsf{Pr}[\varphi_1 = \varphi_2]$ (e.g. $\mathsf{Pr}[\varphi_1 = 1, \varphi_2 = 1]$). We have $\mathbf{y}_{1,2} = \mathbf{M}_{1,2} \cdot \mathbf{p}_{1,2}$, as

$$\mathsf{Pr}[\varphi_1] = \sum_{\varphi_2} \mathsf{Pr}[\varphi_1, \varphi_2] \quad \textit{and} \quad \mathsf{Pr}[\varphi_2] = \sum_{\varphi_1} \mathsf{Pr}[\varphi_1, \varphi_2].$$

For instance, we can get the probability $\mathsf{Pr}[\varphi_1 = 1]$ by multiplying the first row of $\mathbf{M}_{1,2}$ by $\mathbf{p}_{1,2}$, i.e.,

$$\mathsf{Pr}[\varphi_1 = 1] = \sum_{\varphi_2} \mathsf{Pr}[\varphi_1 = 1, \varphi_2],$$

and the probability $\mathsf{Pr}[\varphi_2 = 5]$ by multiplying the last row of $\mathbf{M}_{1,2}$ by $\mathbf{p}_{1,2}$, i.e.,

$$\mathsf{Pr}[\varphi_2 = 5] = \sum_{\varphi_1} \mathsf{Pr}[\varphi_1, \varphi_2 = 5].$$

By fitting this Lasso regression model, the data collector can obtain the joint distribution $\mathsf{Pr}[\varphi_1, \varphi_2]$ and easily derive the distribution $\mathsf{Pr}[\lambda_{1,2}]$ from it.

In the following, we assume that the data collector has obtained the distributions in $\mathcal{S}_1$ and $\mathcal{S}_2$. To obtain the distribution $\mathsf{Pr}[\varphi_1, \lambda_{2,3}]$ in $\mathcal{S}_3$, the data collector constructs the Lasso regression model $\mathbf{y}_{1,2,3} = \mathbf{M}_{1,2,3} \cdot \mathbf{p}_{1,2,3}$, where

$$\mathbf{M}_{1,2,3} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix},$$

$$\mathbf{p}_{1,2,3} = \begin{pmatrix} \mathsf{Pr}[\varphi_1 = 1, \lambda_{2,3} = 1] \\ \mathsf{Pr}[\varphi_1 = 1, \lambda_{2,3} = 2] \\ \vdots \\ \mathsf{Pr}[\varphi_1 = 5, \lambda_{2,3} = 1] \\ \mathsf{Pr}[\varphi_1 = 5, \lambda_{2,3} = 2] \end{pmatrix}, \mathbf{y}_{1,2,3} = \begin{pmatrix} \mathsf{Pr}[\varphi_1 = 1] \\ \vdots \\ \mathsf{Pr}[\varphi_1 = 5] \\ \mathsf{Pr}[\lambda_{2,3} = 1] \\ \mathsf{Pr}[\lambda_{2,3} = 2] \end{pmatrix}$$

In particular, $\mathbf{M}_{1,2,3}$ is a 7×10 binary matrix, in which only two values in each of the first five rows are 1 and five values in each of the last two rows are 1. $\mathbf{p}_{1,2,3}$ is a length-10 column vector. We have $\mathbf{y}_{1,2,3} = \mathbf{M}_{1,2,3} \cdot \mathbf{p}_{1,2,3}$, as

$$\mathsf{Pr}[\varphi_1] = \sum_{\lambda_{2,3}} \mathsf{Pr}[\varphi_1, \lambda_{2,3}] \quad \textit{or} \quad \mathsf{Pr}[\lambda_{2,3}] = \sum_{\varphi_1} \mathsf{Pr}[\varphi_1, \lambda_{2,3}].$$

For instance, we can get the probability $\mathsf{Pr}[\varphi_1 = 1]$ *by multiplying the first row of* $\mathbf{M}_{1,2,3}$ *by* $\mathbf{p}_{1,2,3}$*, i.e.,*

$$\mathsf{Pr}[\varphi_1 = 1] = \sum_{\lambda_{2,3}} \mathsf{Pr}[\varphi_1 = 1, \lambda_{2,3}].$$

Similarly, by fitting this Lasso regression model, the data collector can obtain the joint distribution $\mathsf{Pr}[\varphi_1, \lambda_{2,3}]$.

Notice that in SAFARI, the data collector and users use the same Rule I. In Rule I, there are d attributes in each user's tuple. For each attribute A_j in $\mathcal{A}$, its domain size $|dom(A_j)|$ is only d, which is far less than $d!$. By estimating the frequency of every possible value of each attribute in $\mathcal{A}$, the data collector can obtain the distributions in $\mathcal{S}_1$, and derive the distributions in $\mathcal{S}_2$ and $\mathcal{S}_3$.

5.3 Design of Rule II

After the K-thin chain $\mathcal{T}$ is constructed, the data collector needs to collect the relative ranking distributions and interleaving distributions over $\mathcal{T}$. To this end, we design Rule II, which instructs users to transform their rankings to provide the information about these two types of distributions. The details of Rule II are given below.

Rule II. Each user u_i starts by transforming his/her ranking σ_i into a tuple t_i that contains all the attributes in an attribute set $\mathcal{A}$. Specifically, $\mathcal{A}$ consists of the subsets $\mathcal{A}^{\mathcal{L}}$ and $\mathcal{A}^{\mathcal{G}}$ ($\mathcal{A} = \mathcal{A}^{\mathcal{L}} \cup \mathcal{A}^{\mathcal{G}}$), which are defined as follows.

1) $\mathcal{A}^{\mathcal{L}} = \{A_j | 1 \leq j \leq |\mathcal{L}| - |\mathcal{L}_1|\}$

$\mathcal{A}^{\mathcal{L}}$ corresponds to the leaf item sets $\mathcal{L}$ in $\mathcal{T}$. As a subset of $\mathcal{L}$, $\mathcal{L}_1$ consists of the leaf item sets which contain only one item. For instance, $\mathcal{L}_1$ is $\{l_1\}$ in Fig. 1(a). Since the relative ranking distribution over each leaf item set in $\mathcal{L}_1$ can be easily derived (for example in Fig. 1(a), $\mathsf{Pr}[\sigma^{l_1}] = \mathsf{Pr}[\langle \mathbf{P} \rangle] = 1$), it is unnecessary for users to provide the information about $\mathcal{L}_1$.

Each attribute A_j in $\mathcal{A}^{\mathcal{L}}$ corresponds to a leaf item set l_k in $\mathcal{L} - \mathcal{L}_1$. The domain of A_j consists of all possible relative rankings of l_k. When K is 1, all leaf item sets contain only one item, which leads to $\mathcal{A}^{\mathcal{L}} = \emptyset$.

2) $\mathcal{A}^{\mathcal{G}} = \{A_j | 1 + |\mathcal{A}^{\mathcal{L}}| \leq j \leq |\mathcal{A}^{\mathcal{L}}| + |\mathcal{G}|\}$

Each attribute A_j in $\mathcal{A}^{\mathcal{G}}$ corresponds to an internal item set g_k in $\mathcal{G}$. $dom(A_j)$ consists of all possible interleavings of g_k. For each attribute A_j in $\mathcal{A}$, each user u_i assigns $t_i[A_j]$'s value that is derived from his/her ranking σ_i.

Example 7. *We continue with the running example to illustrate how to transform a ranking by Rule II. From the 2-thin chain in Fig. 1(a), we have* $\mathcal{A}^{\mathcal{L}} = \{A_1, A_2\}$ *and* $\mathcal{A}^{\mathcal{G}} = \{A_3, A_4\}$.

For $\mathcal{A}^{\mathcal{L}}$*, we assume that the attributes* A_1 *and* A_2 *correspond to the leaf item sets* $l_2 = \{\mathbf{B}, \mathbf{W}\}$ *and* $l_3 = \{\mathbf{C}, \mathbf{F}\}$*, respectively. Thus,* $dom(A_1)$ *consists of two possible values, i.e.,* $\langle \mathbf{B}, \mathbf{W} \rangle$ *and* $\langle \mathbf{W}, \mathbf{B} \rangle$*, and* $dom(A_2)$ *also consists of two possible values, i.e.,* $\langle \mathbf{C}, \mathbf{F} \rangle$ *and* $\langle \mathbf{F}, \mathbf{C} \rangle$*. After user* u_0 *transforms* $\sigma_0 = \langle \mathbf{W}, \mathbf{B}, \mathbf{C}, \mathbf{F}, \mathbf{P} \rangle$ *into a tuple* t_0 *by Rule II, we have* $t_0[A_1] = \langle \mathbf{W}, \mathbf{B} \rangle$ *and* $t_0[A_2] = \langle \mathbf{C}, \mathbf{F} \rangle$.

For $\mathcal{A}^{\mathcal{G}}$*, we assume that the attributes* A_3 *and* A_4 *correspond to the internal item sets* $g_1 = \{\mathbf{B}, \mathbf{C}, \mathbf{F}, \mathbf{P}, \mathbf{W}\}$ *and* $g_2 = \{\mathbf{B}, \mathbf{C}, \mathbf{F}, \mathbf{W}\}$*, respectively. Thus,* $dom(A_3)$ *consists of five possible values:*

$$\langle l_1, g_2, g_2, g_2, g_2 \rangle, \langle g_2, l_1, g_2, g_2, g_2 \rangle, \langle g_2, g_2, l_1, g_2, g_2 \rangle,$$
$$\langle g_2, g_2, g_2, l_1, g_2 \rangle, \langle g_2, g_2, g_2, g_2, l_1 \rangle,$$

and $dom(A_4)$ *consists of six possible values:*

$$\langle l_2, l_2, l_3, l_3 \rangle, \langle l_2, l_3, l_2, l_3 \rangle, \langle l_2, l_3, l_3, l_2 \rangle,$$
$$\langle l_3, l_2, l_2, l_3 \rangle, \langle l_3, l_2, l_3, l_2 \rangle, \langle l_3, l_3, l_2, l_2 \rangle.$$

In $\sigma_0 = \langle \mathbf{W}, \mathbf{B}, \mathbf{C}, \mathbf{F}, \mathbf{P} \rangle$*, since the item* ***P*** *in* l_1 *is ranked behind the items in* g_2*, we have* $t_0[A_3] = \langle g_2, g_2, g_2, g_2, l_1 \rangle$*. Similarly, since* $\mathbf{B}$ *and* $\mathbf{W}$ *in* l_2 *are ranked in front of* $\mathbf{C}$ *and* $\mathbf{F}$ *in* l_3*, we have* $t_0[A_4] = \langle l_2, l_2, l_3, l_3 \rangle$.

Notice that in SAFARI, the data collector and users apply the same Rule II. In Rule II, there are $O(d)$ attributes in each user's tuple. For each attribute A_j in $\mathcal{A}^{\mathcal{L}}$, its maximum domain size is $K!$, and for each attribute A_j in $\mathcal{A}^{\mathcal{G}}$, its maximum domain size is $\binom{d}{K}$. Hence the maximum domain size of attributes in $\mathcal{A}$ is $O(max(K!, \binom{d}{K}))$, which is also far less than $d!$. By estimating the frequency of every possible value of each attribute in $\mathcal{A}$, the data collector can obtain all the distributions over $\mathcal{T}$.

5.4 Sampling Randomizer for Multiple Attributes

To collect the distribution information required for constructing the RI model, the data collector needs to estimate the frequency of every possible value of each attribute in users' transformed tuples under LDP.

The data collector could use Harmony [6], the state-of-the-art method focusing on 1-way marginal distribution collection over multi-attribute data under LDP, to achieve this task. In particular, for each attribute A_j in $\mathcal{A}$, the data collector projects the domain of A_j into a binary matrix Φ_j of size $|dom(A_j)| \times |dom(A_j)|$. Then, for each user u_i, the data collector selects one attribute (say A_r) from $\mathcal{A}$ randomly and uses SH [9] to collect the value of A_r in u_i's tuple t_i.

We observe that by using either Rule I or Rule II, each of attributes in the transformed attribute set $\mathcal{A}$ has a small domain that is much less than $d!$. However, for an attribute whose domain size is small, the projection in Harmony will introduce unnecessary noise in the collected data, especially for binary data. Inspired by [10], which proves that GRR performs best when estimating frequencies of a small number of values, we propose a new LDP method, called **S**ampling **RA**ndomizer **F**or Multiple **A**ttributes (SAFA), to obtain more accurate frequency estimates for multiple attributes with small domains under LDP. Its main idea is to let each user release the value of a randomly selected attribute using a local randomizer based on GRR to the data collector.

Algorithm 1 provides the details of SAFA. It takes users' tuples $\{t_i | 1 \leq i \leq n\}$ and privacy budget ε' as inputs, and outputs a vector set $\mathcal{Z}$ of estimated frequencies. For each vector $\mathbf{z}_j$ in $\mathcal{Z}$, it has $|dom(A_j)|$ values. In particular, the k-th value in $\mathbf{z}_j$ (i.e., $\mathbf{z}_j[k]$) denotes the frequency of the k-th value in $dom(A_j)$. In Algorithm 1, the data collector (represented by *DC* in the pseudo code) first initializes all vectors in the set $\mathcal{Z}$ by assigning the values in each vector to zeros. Then, for each user u_i, the data collector draws an index j uniformly at random from $\{1, \ldots, |\mathcal{A}|\}$

Algorithm 1 Sampling Randomizer for Multiple Attributes (SAFA)

Input: Users' tuples $\{t_i | 1 \leq i \leq n\}$
Input: Privacy budget ε'
Output: Frequency vector set $\mathcal{Z} = \{\mathbf{z}_j | 1 \leq j \leq |\mathcal{A}|\}$
1: DC initializes all vectors in $\mathcal{Z}$;
2: **for** each user u_i **do**
3: DC draws an index j randomly from $\{1, \ldots, |\mathcal{A}|\}$
4: DC sends j to u_i ;
5: u_i returns index $\widetilde{k}_i^j = LR(j, t_i, \varepsilon')$ to DC ;
6: DC increases $\mathbf{z}_j[\widetilde{k}_i^j]$ by 1 ;
7: **for** each j-th vector $\mathbf{z}_j$ in $\mathcal{Z}$ **do**
8: DC sets the probability $p_j = \frac{e^{\varepsilon'}}{e^{\varepsilon'} + |dom(A_j)| - 1}$;
9: DC sets the probability $q_j = \frac{1}{e^{\varepsilon'} + |dom(A_j)| - 1}$;
10: **for** each k-th value $\mathbf{z}_j[k]$ in $\mathbf{z}_j$ **do**
11: DC updates

$$\mathbf{z}_j[k] = \frac{1}{n} \cdot \frac{|\mathcal{A}| \cdot \mathbf{z}_j[k] - nq_j}{p_j - q_j} ;$$

12: **return** $\mathcal{Z}$;

Algorithm 2 Local Randomizer (LR)

Input: Attribute index j
Input: User u_i's tuple t_i
Input: Privacy budget ε'
Output: Perturbed value index $\widetilde{k}_i^j$
1: Generate a perturbed value index $\widetilde{k}_i^j$ such that

$$\mathsf{Pr}\left[\widetilde{k}_i^j = k\right] = \begin{cases} \frac{e^{\varepsilon'}}{e^{\varepsilon'} + |dom(A_j)| - 1}, & if\ k = I(t_i[A_j]) \\ \frac{1}{e^{\varepsilon'} + |dom(A_j)| - 1}, & if\ k \neq I(t_i[A_j]) \end{cases},$$

where $k \in \{1, \ldots, |dom(A_j)|\}$.
2: **return** $\widetilde{k}_i^j$;

and sends j to u_i. After that, u_i generates a perturbed value index $\widetilde{k}_i^j$ by a local randomizer whose detail is given in Algorithm 2.

Local Randomizer. To protect users' privacy at the user side in the collection process, we design a local randomizer (LR) based on GRR. Its use in our setting is described in Algorithm 2. At the beginning of LR, it generates a perturbed value index $\widetilde{k}_i^j$. More specifically, with probability $\frac{e^{\varepsilon'}}{e^{\varepsilon'} + |dom(A_j)| - 1}$, $\widetilde{k}_i^j$ equals $I(t_i[A_j])$, the index of $t_i[A_j]$ in $dom(A_j)$, which ranges from 1 to $|dom(A_j)|$; with probability $\frac{1}{e^{\varepsilon'} + |dom(A_j)| - 1}$, $\widetilde{k}_i^j$ equals any value index $k \in \{1, \ldots, |dom(A_j)|\}$ other than $I(t_i[A_j])$. At the end of LR, it returns $\widetilde{k}_i^j$ to the data collector.

With the received $\widetilde{k}_i^j$, the data collector increases $\mathbf{z}_j[\widetilde{k}_i^j]$ by 1 (Line 6 of Algorithm 1). Notice that after the data collector finishes the interactions with all users, the values in each vector in $\mathcal{Z}$ are biased, since each user reports the information of one attribute instead of $|\mathcal{A}|$ attributes and uses LR to perform the random perturbation (Line 1 of Algorithm 2). To get unbiased estimates, the data collector performs the following steps. For each j-th vector $\mathbf{z}_j$ in $\mathcal{Z}$, the data collector sets the probability p_j and q_j (Line 9-10 of Algorithm 1). With p_j and q_j, the data collector updates each value in $\mathbf{z}_j$ (Line 11-13 of Algorithm 1).

Theoretical Analysis. We theoretically analyze the privacy and utility guarantees of SAFA. We first establish the privacy guarantee of SAFA in the following theorem.

Theorem 2. *For any user u_i with privacy budget ε', SAFA is ε'-LDP for u_i.*

Proof. By definition, for any two different tuples t_i, t_i', and any perturbed value index $\widetilde{k}_i^j$ where $j \in \{1, \ldots, |\mathcal{A}|\}$ is the attribute index selected by the data collector, we need to prove that

$$\frac{\mathsf{Pr}\left[SAFA(t_i, \varepsilon') = \widetilde{k}_i^j\right]}{\mathsf{Pr}\left[SAFA(t_i', \varepsilon') = \widetilde{k}_i^j\right]} \leq e^{\varepsilon'}.$$

Due to the random sampling of the attribute index j, we have

$$\begin{aligned} &\frac{\mathsf{Pr}\left[SAFA(t_i, \varepsilon') = \widetilde{k}_i^j\right]}{\mathsf{Pr}\left[SAFA(t_i', \varepsilon') = \widetilde{k}_i^j\right]} \\ &= \frac{\mathsf{Pr}\left[j\ is\ sampled\right] \cdot \mathsf{Pr}\left[LR(j, t_i, \varepsilon') = \widetilde{k}_i^j\right]}{\mathsf{Pr}\left[j\ is\ sampled\right] \cdot \mathsf{Pr}\left[LR(j, t_i', \varepsilon') = \widetilde{k}_i^j\right]} \\ &= \frac{\mathsf{Pr}\left[LR(j, t_i, \varepsilon') = \widetilde{k}_i^j\right]}{\mathsf{Pr}\left[LR(j, t_i', \varepsilon') = \widetilde{k}_i^j\right]} \\ &= \frac{\mathsf{Pr}\left[\widetilde{k}_i^j | I(t_i[A_j])\right]}{\mathsf{Pr}\left[\widetilde{k}_i^j | I(t_i'[A_j])\right]}. \end{aligned} \tag{2}$$

We discuss Equation (2) in all four possible cases:

Case 1: if $I(t_i[A_j]) = \widetilde{k}_i^j$ and $I(t_i'[A_j]) = \widetilde{k}_i^j$, $\frac{\mathsf{Pr}\left[\widetilde{k}_i^j | I(t_i[A_j])\right]}{\mathsf{Pr}\left[\widetilde{k}_i^j | I(t_i'[A_j])\right]} = \frac{e^{\varepsilon'}}{e^{\varepsilon'} + |dom(A_j)| - 1} \Big/ \frac{e^{\varepsilon'}}{e^{\varepsilon'} + |dom(A_j)| - 1} = 1$;

Case 2: if $I(t_i[A_j]) \neq \widetilde{k}_i^j$ and $I(t_i'[A_j]) = \widetilde{k}_i^j$, $\frac{\mathsf{Pr}\left[\widetilde{k}_i^j | I(t_i[A_j])\right]}{\mathsf{Pr}\left[\widetilde{k}_i^j | I(t_i'[A_j])\right]} = \frac{1}{e^{\varepsilon'} + |dom(A_j)| - 1} \Big/ \frac{e^{\varepsilon'}}{e^{\varepsilon'} + |dom(A_j)| - 1} = e^{-\varepsilon'}$;

Case 3: if $I(t_i[A_j]) = \widetilde{k}_i^j$ and $I(t_i'[A_j]) \neq \widetilde{k}_i^j$, $\frac{\mathsf{Pr}\left[\widetilde{k}_i^j | I(t_i[A_j])\right]}{\mathsf{Pr}\left[\widetilde{k}_i^j | I(t_i'[A_j])\right]} = \frac{e^{\varepsilon'}}{e^{\varepsilon'} + |dom(A_j)| - 1} \Big/ \frac{1}{e^{\varepsilon'} + |dom(A_j)| - 1} = e^{\varepsilon'}$;

Case 4: if $I(t_i[A_j]) \neq \widetilde{k}_i^j$ and $I(t_i'[A_j]) \neq \widetilde{k}_i^j$, $\frac{\mathsf{Pr}\left[\widetilde{k}_i^j | I(t_i[A_j])\right]}{\mathsf{Pr}\left[\widetilde{k}_i^j | I(t_i'[A_j])\right]} = \frac{1}{e^{\varepsilon'} + |dom(A_j)| - 1} \Big/ \frac{1}{e^{\varepsilon'} + |dom(A_j)| - 1} = 1$.

Therefore, we have $\frac{\mathsf{Pr}\left[SAFA(t_i, \varepsilon') = \widetilde{k}_i^j\right]}{\mathsf{Pr}\left[SAFA(t_i', \varepsilon') = \widetilde{k}_i^j\right]} \leq e^{\varepsilon'}$. As such, $SAFA$ is ε'-LDP for u_i. □

In what follows, we give the utility guarantee of SAFA. In particular, we have the following theorems.

Theorem 3. *Let $\mathbf{f}_j[k]$ be the true frequency of the k-th value in $dom(A_j)$ for n users. Then, for any attribute index $j \in \{1, \ldots, |\mathcal{A}|\}$ and value index $k \in \{1, \ldots, |dom(A_j)|\}$, we have*

$$\mathbb{E}\left[\mathbf{z}_j[k]\right] = \mathbf{f}_j[k].$$

Proof. To start with, we define a function

$$\mathbb{Y}_j^k(i) = \begin{cases} 1, & if\ DC\ sends\ j\ to\ u_i\ and\ \widetilde{k}_i^j = k \\ 0, & else \end{cases}.$$

Then, we have

$$\begin{aligned}&\mathbb{E}\left[\mathbf{z}_j[k]\right]\\&=\mathbb{E}\left[\frac{1}{n}\cdot\frac{|\mathcal{A}|\sum_i^n \mathbb{Y}_j^k(i)-nq_j}{p_j-q_j}\right]\\&=\frac{1}{p_j-q_j}\cdot\left[\frac{|\mathcal{A}|}{n}\cdot\mathbb{E}\left[\sum_i^n \mathbb{Y}_j^k(i)\right]-q_j\right].\end{aligned}\tag{3}$$

Due to the random sampling of the attribute index j, the attribute A_j is selected with probability $\frac{1}{|\mathcal{A}|}$. Hence, we have

$$\begin{aligned}&\mathbb{E}\left[\sum_i^n \mathbb{Y}_j^k(i)\right]\\&=\frac{n}{|\mathcal{A}|}\cdot\left[\mathbf{f}_j[k]\cdot p_j+(1-\mathbf{f}_j[k])\cdot q_j\right]\\&=\frac{n}{|\mathcal{A}|}\cdot\left[\mathbf{f}_j[k]\cdot(p_j-q_j)+q_j\right].\end{aligned}\tag{4}$$

By substituting Equation (4) into Equation (3), we obtain $\mathbb{E}\left[\mathbf{z}_j[k]\right]=\mathbf{f}_j[k]$. This completes the proof. □

Theorem 3 shows that SAFA is an unbiased estimator and explains why the untrusted data collector can learn useful information regarding the true frequency of every possible value of each attribute in $\mathcal{A}$. The following theorem (i.e., Theorem 4) shows the variation of the estimated frequency of every possible value of each attribute in $\mathcal{A}$.

Theorem 4. *Let $\mathbf{f}_j[k]$ be the true frequency of the k-th value in $dom(A_j)$ for n users. Then, for any attribute index $j\in\{1,\ldots,|\mathcal{A}|\}$ and value index $k\in\{1,\ldots,|dom(A_j)|\}$, the variance of $\mathbf{z}_j[k]$ is*

$$Var\left[\mathbf{z}_j[k]\right]\approx\frac{\left(e^{\varepsilon'}+|dom(A_j)|-1\right)\cdot|\mathcal{A}|-1}{n\cdot\left(e^{\varepsilon'}-1\right)^2}.$$

Proof. Initially, we have

$$\begin{aligned}&Var\left[\mathbf{z}_j[k]\right]\\&=Var\left[\frac{1}{n}\cdot\frac{|\mathcal{A}|\sum_i^n \mathbb{Y}_j^k(i)-nq_j}{p_j-q_j}\right]\\&=\frac{|\mathcal{A}|^2}{n^2\cdot\left(p_j-q_j\right)^2}\cdot Var\left[\sum_i^n \mathbb{Y}_j^k(i)\right].\end{aligned}\tag{5}$$

The random variable $\sum_i^n \mathbb{Y}_j^k(i)$ is the summation of n independent random variables drawn from the Bernoulli distribution. For n users, $n\cdot\mathbf{f}_j[k]$ (resp. $n\cdot(1-\mathbf{f}_j[k])$) of these random variables are from the Bernoulli distribution with parameter $\frac{p_j}{|\mathcal{A}|}$ (resp. $\frac{q_j}{|\mathcal{A}|}$). Thus, we have

$$\begin{aligned}&Var\left[\sum_i^n \mathbb{Y}_j^k(i)\right]\\&=n\cdot\mathbf{f}_j[k]\cdot\left[\frac{p_j}{|\mathcal{A}|}\cdot\left(1-\frac{p_j}{|\mathcal{A}|}\right)\right]\\&+n\cdot(1-\mathbf{f}_j[k])\cdot\left[\frac{q_j}{|\mathcal{A}|}\cdot\left(1-\frac{q_j}{|\mathcal{A}|}\right)\right].\end{aligned}\tag{6}$$

By substituting Equation (6) into Equation (5), we obtain

$$\begin{aligned}&Var\left[\mathbf{z}_j[k]\right]\\&=\frac{\mathbf{f}_j[k]\cdot[p_j\cdot(|\mathcal{A}|-p_j)]+(1-\mathbf{f}_j[k])\cdot[q_j\cdot(|\mathcal{A}|-q_j)]}{n\cdot\left(p_j-q_j\right)^2}\\&=\frac{q_j\cdot(|\mathcal{A}|-q_j)}{n\cdot\left(p_j-q_j\right)^2}+\frac{\mathbf{f}_j[k]\cdot\left[|\mathcal{A}|\cdot(p_j-q_j)-\left({p_j}^2-{q_j}^2\right)\right]}{n\cdot\left(p_j-q_j\right)^2}\\&\approx\frac{q_j\cdot(|\mathcal{A}|-q_j)}{n\cdot\left(p_j-q_j\right)^2}\\&=\frac{\left(e^{\varepsilon'}+|dom(A_j)|-1\right)\cdot|\mathcal{A}|-1}{n\cdot\left(e^{\varepsilon'}-1\right)^2}.\end{aligned}\tag{7}$$

This completes the proof. □

Theorem 5. *For any attribute index $j\in\{1,\ldots,|\mathcal{A}|\}$, compared with Harmony, SAFA can achieve higher accuracy of the frequency of every possible value of A_j when*

$$|dom(A_j)|<\frac{(2|\mathcal{A}|-1)\cdot\left(e^{\varepsilon'}+1\right)^2+1}{|\mathcal{A}|}-e^{\varepsilon'}+1.$$

Proof. Based on the analysis of Binary Local Hashing in [10], we can derive that the variance of $\mathbf{z}_j[k]$ collected by Harmony is

$$Var_H\left[\mathbf{z}_j[k]\right]\approx\frac{(2|\mathcal{A}|-1)\cdot\left(e^{\varepsilon'}+1\right)^2}{n\cdot\left(e^{\varepsilon'}-1\right)^2}.\tag{8}$$

Let Equation (7) $<$ Equation (8), then we have

$$|dom(A_j)|<\frac{(2|\mathcal{A}|-1)\cdot\left(e^{\varepsilon'}+1\right)^2+1}{|\mathcal{A}|}-e^{\varepsilon'}+1.\tag{9}$$

This completes the proof. □

5.5 Privacy Analysis of SAFARI

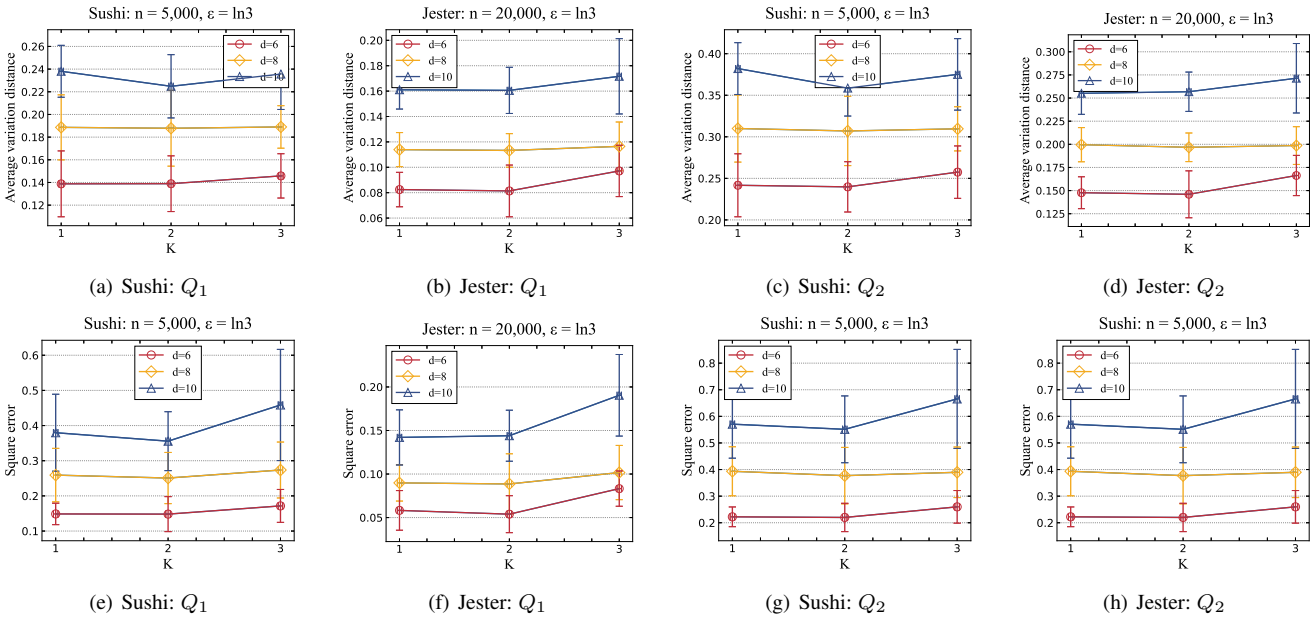
In this section, we give the privacy analysis of the proposed approach SAFARI. The following theorem establishes the privacy guarantee of SAFARI.

Theorem 6. *SAFARI satisfies ε-LDP.*

Proof. In SAFARI, the data collector needs to access a user's raw data exactly twice. In Phases 2 and 4, the data collector invokes SAFA using privacy budget $\varepsilon/2$, respectively. By Theorem 2, each of these two phases satisfies $\varepsilon/2$-LDP. Note that since making Rules I and II does not require the data collector to access any user's raw data, it does not lead to any privacy risk or consume any privacy budget. In particular, making Rule I depends only on the given item set, while making Rule II depends only on the differentially private distributions collected from users' data. In addition, all other steps are performed on already differentially private outputs. Therefore, by Theorem 1, SAFARI in which SAFA is sequentially invoked twice satisfies ε-LDP. This completes the proof. □

6 Experiments

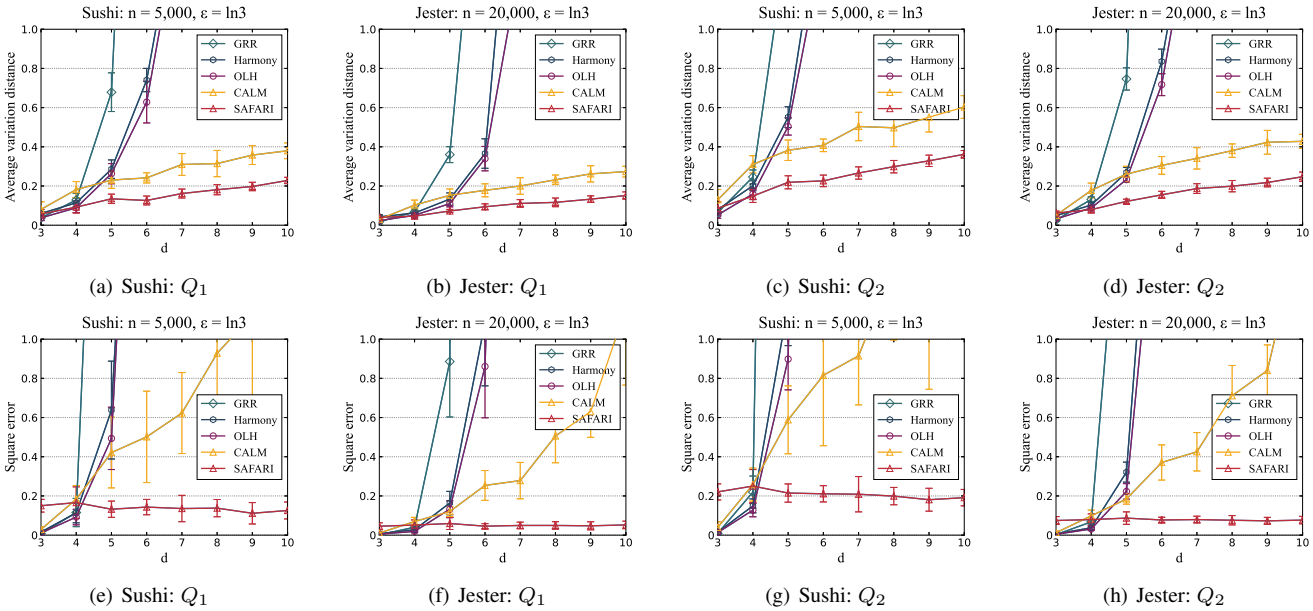
In this section, we first evaluate the performance of SAFARI and then evaluate the effectiveness of SAFA, LADE and the Lasso regression model.



(a) Sushi: Q_1 (b) Jester: Q_1 (c) Sushi: Q_2 (d) Jester: Q_2

(e) Sushi: Q_1 (f) Jester: Q_1 (g) Sushi: Q_2 (h) Jester: Q_2

Fig. 2: Impact of K



(a) Sushi: Q_1 (b) Jester: Q_1 (c) Sushi: Q_2 (d) Jester: Q_2

(e) Sushi: Q_1 (f) Jester: Q_1 (g) Sushi: Q_2 (h) Jester: Q_2

Fig. 3: Impact of d

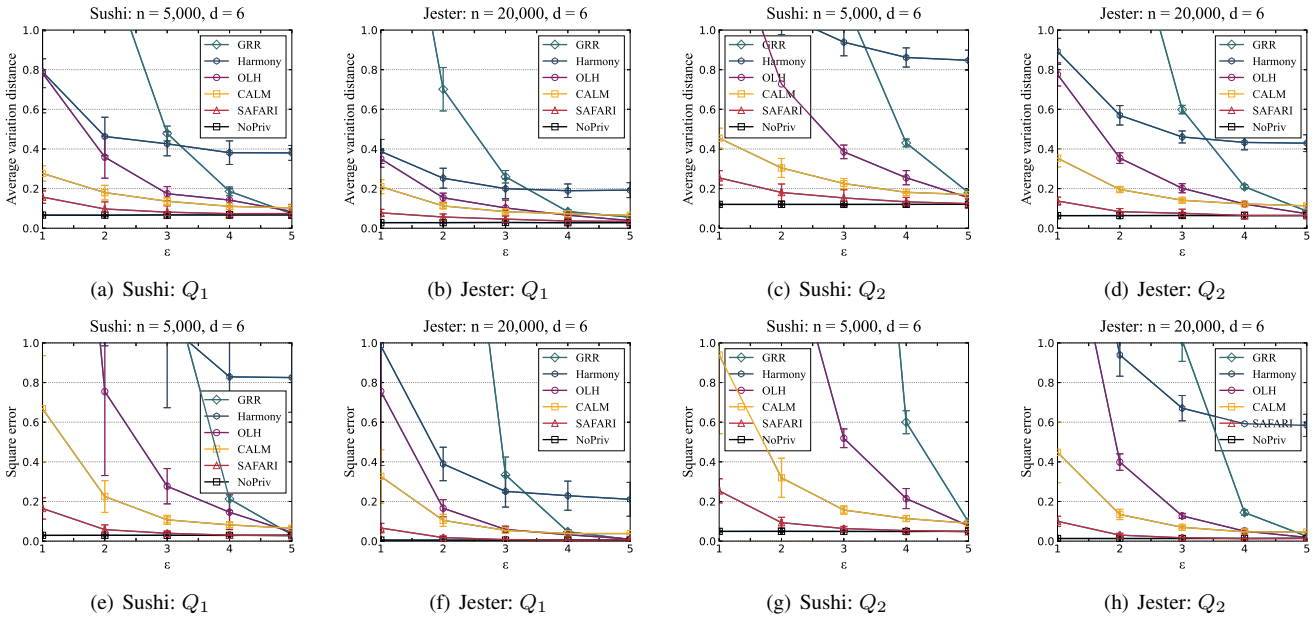
6.1 Experimental Settings

We make use of the following two real ranking datasets in our experiments.

- Sushi [41]: It consists of rankings of 10 kinds of sushi from 5000 voters surveyed across Japan.
- Jester [42]: It includes 1.7 Million continuous ratings (-10.00 to 10.00) of jokes from 50,692 users, which were collected from 2006 to 2012. To convert the dataset into rankings, we order jokes according to their real-valued ratings.

To experiment with different numbers of items, we generate multiple test datasets from these two datasets. From the Sushi dataset, we generate test datasets with 5,000 users and the number of items ranging from 3 to 10. Similarly, from the Jester dataset, we generate test datasets with 20,000 users and the number of items ranging from 3 to 10.

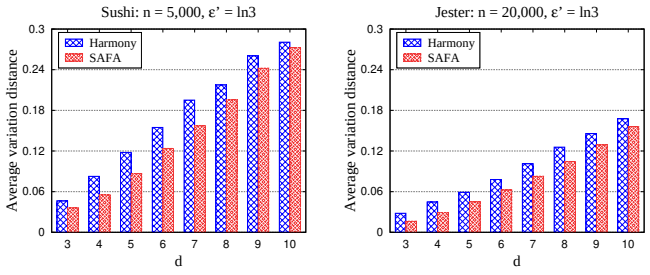
To evaluate the performance of SAFARI, we examine the accuracy of the first-order marginals and the second-order marginals [4] of the synthetic ranking dataset. For convenience, we denote Q_1 as the set of all the first-order marginals and denote Q_2 as the set of all the second-order marginals. To measure the accuracy of each noisy marginal, the average total variation distance [43] (AVD) and the sum of squared error (SSE) [] are reported. Specifically, AVD is a distance measure for probability distributions and SSE is an accuracy measure where the errors are squared, then added. Formally, on a finite probability space Ω,



(a) Sushi: Q_1 (b) Jester: Q_1 (c) Sushi: Q_2 (d) Jester: Q_2

(e) Sushi: Q_1 (f) Jester: Q_1 (g) Sushi: Q_2 (h) Jester: Q_2

Fig. 4: Impact of ε



(a) Sushi: varying d (b) Jester: varying d

Fig. 5: SAFA vs Harmony

AVD between distributions P and P'

$$AVD\left(P,P'\right)=\frac{1}{2}\left\|P-P'\right\|_1=\frac{1}{2}\sum_{\omega\in\Omega}\left|P\left(\omega\right)-P'\left(\omega\right)\right|,$$

and SSE between distributions P and P' is

$$SSE\left(P,P'\right)=\sum_{\omega\in\Omega}\left|P\left(\omega\right)-P'\left(\omega\right)\right|^2.$$

We report AVD and SSE over all marginals in Q_1 and Q_2.

We compare SAFARI against the four baseline approaches, i.e., the GRR based approach, the Harmony based approach, the OLH based approach and the CALM based approach, which are described in Section 4. For ease of presentation, we respectively denote them by GRR, Harmony, OLH and CALM in our experiments.

We implemented all the approaches in C/C++. The source code of our SAFARI approach is publicly available at [44]. All experiments are conducted on an Intel Core i5 3.30GHz PC with 8GB RAM. In our experiments, we run each approach 10 times and report the average results.

6.2 Performance of SAFARI

In general, four parameters can affect the performance of SAFARI.

The impact of K. In SAFARI, the only internal parameter is the maximum size of leaf item sets in the K-thin chain $\mathcal{T}$ (i.e., K). Fig. 2 shows the results of SAFARI under different values of K as a function of the number d of items when the privacy budget ε is fixed at $\ln 3$.

From Fig. 2, we can observe that in general $K=1$ achieves the best performance. This is because a larger K value normally makes the maximum domain size of attributes in Rule II become overly large, leading to excessive injected noise in the collected distributions over the constructed K-thin chain. The only exception occurs when $d=6$. When d is sufficiently small, the benefit of capturing more correlations by using a slightly larger K value (i.e., $K=2$) outweighs the harm of more noise. In any case, using an even larger K value ($K=3$) introduces excessive noise, canceling out the benefit of capturing more correlations.

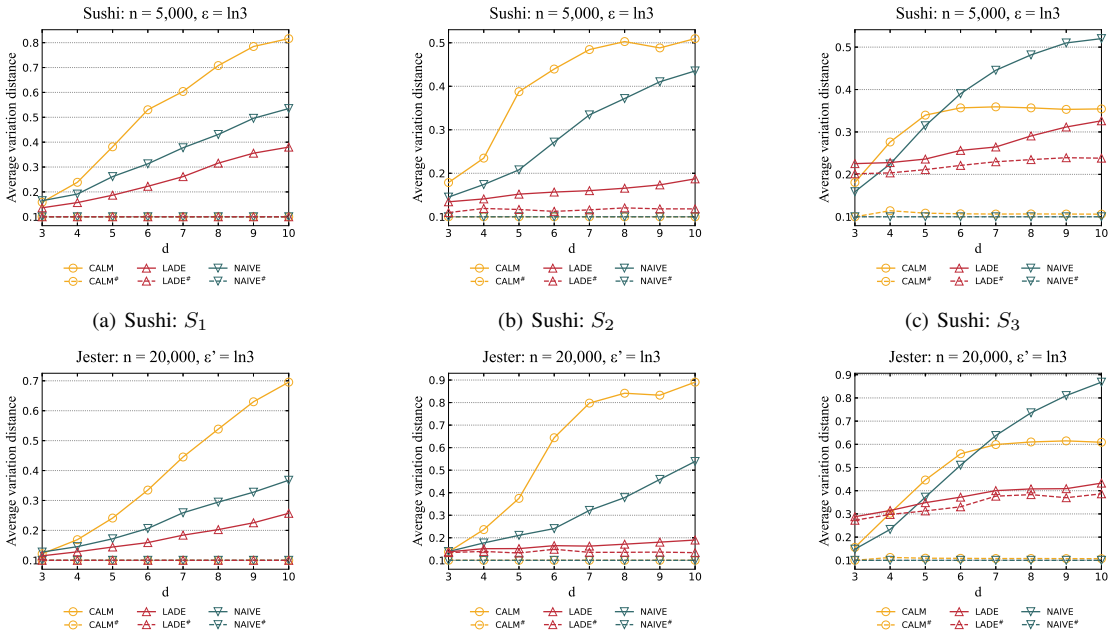
Since $K=1$ usually gives good results, we set $K=1$ in all subsequent experiments.

The impact of d. We fix $\varepsilon=\ln 3$ and vary the number d of items to study its impact on the utility of each approach. The results are shown in Fig. 3.

It should be noted that the range of AVD is normally from 0 to 1. However, as explained in [12], due to the perturbation at the user side, the estimated frequencies of an attribute's values might be negative, which could make the AVD of a baseline approach larger than 1.

Not surprisingly, for all approaches, their utilities degrade when d increases. Note that GRR, RAPPOR, SH and OLH can perform well when d is less than 4. This is because when d is small, the number of all possible rankings is still manageable. However, when d becomes relatively large, the advantage of SAFARI compared to the baseline approaches becomes much more observable, which confirms its good scalability.

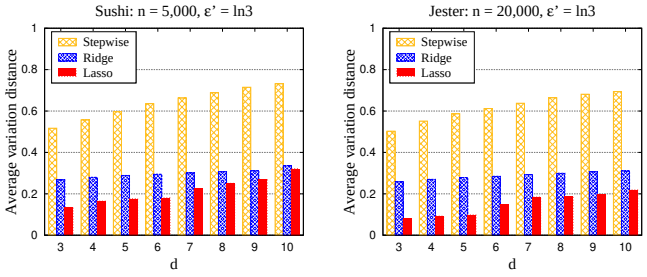
The impact of ε. We fix $d=6$ and report the AVD and SSE of each approach when varying the privacy budget ε. To show the utility loss due to privacy protection, we include a non-private



(a) Sushi: S_1 (b) Sushi: S_2 (c) Sushi: S_3

(d) Jester: S_1 (e) Jester: S_2 (f) Jester: S_3

Fig. 6: Effectiveness of LADE



(a) Sushi: varying d (b) Jester: varying d

Fig. 7: Regression model selection

version of SAFARI, denoted by NoPriv, in this set of experiments. Fig. 4 shows the AVD and SSE over the two datasets under different ε.

As expected, for all approaches, as ε increases, their errors become lower. From Fig. 4, we can observe that SAFARI clearly outperforms the baseline approaches and the relative superiority of SAFARI is more pronounced when ε decreases. The reason can be explained as follows. The K-thin chain makes the collected distributions in SAFARI more robust against noise injection, which ensures that the quality of the synthetic data does not degrade significantly as ε decreases. In contrast, the performance of the baseline approaches is highly sensitive to ε, since they incur considerable noise when ε decreases. Additionally, we find that the errors of Q_2 are much larger than those of Q_1 for the baseline approaches. In contrast, for SAFARI, the difference between them is small. The reason is that the RI model enables SAFARI to effectively capture the correlations among items. Besides, we notice that, SAFARI obtains close performance to NoPriv, especially when ε is large.

The impact of n. Fig. 2-4 indicate that under the same experimental parameter setting, for different approaches, the results on Jester are always more accurate than those on Sushi. Thus, our experiments imply that a larger population can effectively reduce the amount of added noise in the collected data and better approximate the true distribution of rankings under LDP. It can be expected that when applying SAFARI to real-world applications, we are able to achieve desirable performance.

6.3 Effectiveness of SAFA

To evaluate the effectiveness of SAFA, we compare it against Harmony [6]. We let the data collector respectively invoke SAFA and Harmony to collect the distributions in $\mathcal{S}_1$ from the users' data transformed by Rule I, and report the AVD of the obtained distributions. Similarly, we fix $\varepsilon' = \ln 3$ and vary the number d of items in this set of experiments. The results are illustrated in Fig. 5. We can observe that SAFA outperforms Harmony in all cases. This is consistent with our analysis in Section 5.4 that SAFA can improve the accuracy of the collected frequencies of attributes whose domain sizes are relatively small.

6.4 Effectiveness of LADE

To evaluate the effectiveness of LADE, we compare it against NAIVE and CALM. We report the AVD of the obtained distributions in $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$, respectively. In this set of experiments, we fix $\varepsilon' = \ln 3$ and vary the number d of items. The results are shown in Fig. 6.

From Fig. 6(a, b, d, and e), we can observe that, for both $\mathcal{S}_1$ and $\mathcal{S}_2$, LADE achieves the best performance. This is consistent with our analysis in Section 5.2.3, i.e., LADE avoids the curse of dimensionality and mitigates the accumulation of LDP noise, and thus leads to higher accuracy of $\mathcal{S}_1$ and $\mathcal{S}_2$. From Fig. 6(c and f), we find that when d is not greater than 4, both NAIVE and CALM may produce low errors in $\mathcal{S}_3$. This is because when d is small, the issues in these two methods are less damaging. However, when d

is relatively large, LADE results in significantly better results than NAIVE and CALM, which confirms its good scalability.

To evaluate the effectiveness of Lasso regression model in LADE, we also compare the effectiveness of NAIVE, CALM and LADE before adding LDP noise, which are denoted as NAIVE*, CALM* and LADE*. From Fig. 6, it is obvious that the estimation error of LADE* is much higher than that of NAIVE* and CALM* before adding LDP noise, which is consistent with our analysis. However, when comparing these methods' performance with the versions that satisfying LDP, we can observe that the effectiveness of LADE is outperforms the others since NAIVE and CALM suffer from the curse of the dimensionality and the accumulation of LDP noise.

6.5 Effectiveness of Lasso Regression Model

To confirm the effectiveness of Lasso regression model, we compare it with another two canonical regression models, namely Stepwise regression and Ridge regression, for solving the sparse linear regression problem. We let the data collector first use SAFA to collect the distributions in $\mathcal{S}_1$ from the users' data transformed by Rule I and then use these three regression models to estimate the distributions in $\mathcal{S}_2 \bigcup \mathcal{S}_3$ from $\mathcal{S}_1$, respectively. We report the AVD of the obtained distributions in $\mathcal{S}_3$.

In this set of experiments, we fix $\varepsilon' = \ln 3$ and vary the number d of items. Fig. 7 shows the results. We can observe that Lasso regression outperforms the other two models consistently. The reason can be explained as follows. For Stepwise regression, since it gives biased regression coefficients that need shrinkage and yields predicted values that are falsely narrow, it cannot achieve desirable performance. For Ridge regression, though it is much better than Stepwise regression and sometimes even close to Lasso regression, it involves a lot of computations in the regularization parameter adjustment which is inefficient. Therefore, we choose the Lasso regression model for estimating the distributions in $\mathcal{S}_2 \bigcup \mathcal{S}_3$ from $\mathcal{S}_1$.

7 Conclusion

In this paper, we present SAFARI, a novel approach for collecting preference rankings under LDP. In SAFARI, we design two transformation rules to instruct users to transform their rankings into multi-attribute data in which each attribute has a small domain. We propose LADE, a new method to precisely estimate the required distributions used to learn the structure of the RI model. We also propose SAFA, a new LDP method to accurately estimate the frequency of every possible value of each attribute in users' transformed data. We show that SAFARI guarantees ε-LDP. Our results demonstrate the effectiveness of SAFARI. We believe that our work provides an effective means of data collection in many real-world applications while guaranteeing LDP.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 61872045, No. 62072052, and No. 62072136), the National Key R&D Program of China under Grant No. 2020YFB1710200, the Foundation for Innovative Research Groups of the National Natural Science Foundation of China (Grant No. 61921003), and the Fundamental Research Funds for the Central Universities under Grant No. 3072020CFT2402.

References

[1] Ú. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: randomized aggregatable privacy-preserving ordinal response," in *CCS*. ACM, 2014, pp. 1054–1067.
[2] J. Tang, A. Korolova, X. Bai, X. Wang, and X. Wang, "Privacy loss in apple's implementation of differential privacy on macos 10.12," *CoRR*, vol. abs/1709.02753, 2017.
[3] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith, "Calibrating noise to sensitivity in private data analysis," in *TCC*, vol. 3876. Springer, 2006, pp. 265–284.
[4] J. Huang and C. Guestrin, "Riffled independence for ranked data," in *NIPS*. Curran Associates, Inc., 2009, pp. 799–807.
[5] ——, "Learning hierarchical riffle independent groupings from rankings," in *ICML*. Omnipress, 2010, pp. 455–462.
[6] T. T. Nguyên, X. Xiao, Y. Yang, S. C. Hui, H. Shin, and J. Shin, "Collecting and analyzing data from smart device users with local differential privacy," *CoRR*, vol. abs/1606.05053, 2016.
[7] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. D. Smith, "What can we learn privately?" in *FOCS*. IEEE Computer Society, 2008, pp. 531–540.
[8] S. L. Warner, "Randomized response: A survey technique for eliminating evasive answer bias," *Journal of the American Statistical Association*, vol. 60, no. 309, pp. 63–69, 1965.
[9] R. Bassily and A. D. Smith, "Local, private, efficient protocols for succinct histograms," in *STOC*. ACM, 2015, pp. 127–135.
[10] T. Wang, J. Blocki, N. Li, and S. Jha, "Locally differentially private protocols for frequency estimation," in *USENIX Security*. USENIX Association, 2017, pp. 729–745.
[11] G. C. Fanti, V. Pihur, and Ú. Erlingsson, "Building a RAPPOR with the unknown: Privacy-preserving learning of associations and data dictionaries," *Proc. Priv. Enhancing Technol.*, vol. 2016, no. 3, pp. 41–61, 2016.
[12] P. Kairouz, K. Bonawitz, and D. Ramage, "Discrete distribution estimation under local privacy," in *ICML*, vol. 48. JMLR.org, 2016, pp. 2436–2444.
[13] J. Hsu, S. Khanna, and A. Roth, "Distributed private heavy hitters," in *ICALP*, vol. 7391. Springer, 2012, pp. 461–472.
[14] R. Bassily, K. Nissim, U. Stemmer, and A. G. Thakurta, "Practical locally private heavy hitters," in *NIPS*, 2017, pp. 2288–2296.
[15] M. Bun, J. Nelson, and U. Stemmer, "Heavy hitters and the structure of local privacy," in *PODS*. ACM, 2018, pp. 435–447.
[16] Z. Qin, Y. Yang, T. Yu, I. Khalil, X. Xiao, and K. Ren, "Heavy hitter estimation over set-valued data with local differential privacy," in *CCS*. ACM, 2016, pp. 192–203.
[17] T. Wang, N. Li, and S. Jha, "Locally differentially private frequent itemset mining," in *SP*. IEEE Computer Society, 2018, pp. 127–143.
[18] N. Wang, X. Xiao, Y. Yang, T. D. Hoang, H. Shin, J. Shin, and G. Yu, "Privtrie: Effective frequent term discovery under local differential privacy," in *ICDE*. IEEE Computer Society, 2018, pp. 821–832.
[19] X. Ren, C. Yu, W. Yu, S. Yang, X. Yang, J. A. McCann, and P. S. Yu, "Lopub: High-dimensional crowdsourced data publication with local differential privacy," *IEEE TIFS*, vol. 13, no. 9, pp. 2151–2166, 2018.
[20] G. Cormode, T. Kulkarni, and D. Srivastava, "Marginal release under local differential privacy," in *SIGMOD*. ACM, 2018, pp. 131–146.
[21] Z. Zhang, T. Wang, N. Li, S. He, and J. Chen, "CALM: consistent adaptive local marginal for marginal release under local differential privacy," in *CCS*. ACM, 2018, pp. 212–229.
[22] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *FOCS*. IEEE Computer Society, 2013, pp. 429–438.
[23] P. Kairouz, S. Oh, and P. Viswanath, "Extremal mechanisms for local differential privacy," in *NIPS*, 2014, pp. 2879–2887.
[24] M. E. Gursoy, A. Tamersoy, S. Truex, W. Wei, and L. Liu, "Secure and utility-aware data collection with condensed local differential privacy," *TDSC*, 2019.
[25] R. Chen, H. Li, A. K. Qin, S. P. Kasiviswanathan, and H. Jin, "Private spatial data aggregation in the local setting," in *ICDE*. IEEE Computer Society, 2016, pp. 289–300.
[26] Q. Ye, H. Hu, X. Meng, and H. Zheng, "Privkv: Key-value data collection with local differential privacy," in *SP*. IEEE, 2019, pp. 317–331.
[27] X. Gu, M. Li, Y. Cheng, L. Xiong, and Y. Cao, "PCKV: locally differentially private correlated key-value data collection with optimized utility," in *USENIX Security*. USENIX Association, 2020, pp. 967–984.
[28] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren, "Generating synthetic decentralized social graphs with local differential privacy," in *CCS*. ACM, 2017, pp. 425–438.

[29] H. Sun, X. Xiao, I. Khalil, Y. Yang, Z. Qin, W. H. Wang, and T. Yu, "Analyzing subgraph statistics from extended local views with decentralized differential privacy," in *CCS*. ACM, 2019, pp. 703–717.
[30] C. Wei, S. Ji, C. Liu, W. Chen, and T. Wang, "Asgldp: Collecting and generating decentralized attributed graphs with local differential privacy," *IEEE TIFS*, vol. 15, pp. 3239–3254, 2020.
[31] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," in *NIPS*, 2017, pp. 3571–3580.
[32] S. Xiong, A. D. Sarwate, and N. B. Mandayam, "Randomized requantization with local differential privacy," in *ICASSP*. IEEE, 2016, pp. 2189–2193.
[33] M. Joseph, A. Roth, J. Ullman, and B. Waggoner, "Local differential privacy for evolving data," in *NIPS*, 2018, pp. 2381–2390.
[34] S. Truex, L. Liu, K. H. Chow, M. E. Gursoy, and W. Wei, "Ldp-fed: federated learning with local differential privacy," in *EdgeSys@EuroSys*. ACM, 2020, pp. 61–66.
[35] M. A. P. Chamikara, P. Bertók, I. Khalil, D. Liu, S. Camtepe, and M. Atiquzzaman, "Local differential privacy for deep learning," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5827–5842, 2020.
[36] C. Xu, J. Ren, L. She, Y. Zhang, Z. Qin, and K. Ren, "Edgesanitizer: Locally differentially private deep inference at the edge for mobile data analytics," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5140–5151, 2019.
[37] J. Yang, X. Cheng, S. Su, R. Chen, Q. Ren, and Y. Liu, "Collecting preference rankings under local differential privacy," in *ICDE*. IEEE, 2019, pp. 1598–1601.
[38] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.
[39] F. McSherry, "Privacy integrated queries: an extensible platform for privacy-preserving data analysis," in *SIGMOD*. ACM, 2009, pp. 19–30.
[40] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani *et al.*, "Least angle regression," *Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.
[41] "Sushi data," http://www.kamishima.net/sushi.
[42] "Jester data," http://www.ieor.berkeley.edu/∼goldberg/jester-data.
[43] A. B. Tsybakov, *Introduction to Nonparametric Estimation*, ser. Springer series in statistics. Springer, 2009.
[44] "Source code of safari," [Online], 2021, https://github.com/cheng-lab-at-bupt/SAFARI.

Xiang Cheng received his PhD degree in computer science from the Beijing University of Posts and Telecommunications, China, in 2013. He is currently an associate professor at the Beijing University of Posts and Telecommunications. His research interests include data mining and data privacy.

Jianyu Yang is working toward his PhD degree at the Beijing University of Posts and Telecommunications in China. His major is computer science. His research interests include data mining and data privacy.

Yufei Wang is working toward his PhD degree at the Beijing University of Posts and Telecommunications in China. His major is computer science. His research interests include data mining and data privacy.

Sen Su received his PhD degree in computer science from the University of Electronic Science and Technology, China, in 1998. He is currently a professor at the Beijing University of Posts and Telecommunications. His research interests include service computing, cloud computing and data privacy.

Rui Chen received the Ph.D. degree in Computer Science from Concordia University. He is a professor in the College of Computer Science and Technology, Harbin Engineering University. His research interests include machine learning, data privacy and databases. He has published more than 50 technical papers in top venues, including CSUR, VLDBJ, PVLDB, TKDE, TDSC, NeurIPS, KDD, CCS, IJCAI, and ICDE, and won CIKM 2015 Best Paper Runner Up, the Best Papers of ICDE 2016 and the Best Papers of ICDM 2018. He has served as program committee member for leading conferences, including KDD, AAAI, IJCAI, ICDM, and CIKM, and as reviewer for numerous flagship journals, including VLDBJ, PVLDB, TKDE, TDSC, TIFS, and TOPS.

Yuejia Li is working toward her master's degree at the Beijing University of Posts and Telecommunications in China. Her major is computer science. His research interests include data mining and data privacy.