Collecting Preference Rankings under Local Differential Privacy (technical report)

Jianyu Yang †1, Xiang Cheng †2*, Sen Su †3, Rui Chen ‡, Qiyu Ren †4, Yuhan Liu †5
†State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing, China
{jyyang1, chengxiang2, susen3, qyren4, liuyuhan5}@bupt.edu.cn
‡Samsung Research America, Mountain View, USA
rui.chen1@samsung.com

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 model, to approximate the overall distribution of users' rankings. By working on small domains instead of a large domain, SAFARI can significantly reduce the magnitude of added noise. In particular, 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 further 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.

I. 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, it has been increasingly prevalent for users to share their rankings with diverse data collectors (e.g., service providers) through mobile applications in order to receive personalized services. Collecting and analyzing users' rankings is also essential for service providers to provide 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.

*Corresponding author

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 local differential privacy. 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 local differential privacy. A straightforward solution is to consider each user's ranking as a categorical value in a domain that consists of all possible rankings and directly collect the users' rankings using existing LDP methods, such as RAPPOR [1], SH [4] and OLH [5]. 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 local differential privacy. In contrast to collecting 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 [6, 7], a well-established probabilistic graphical model in the field of machine learning to model rankings. The riffle independent 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 according to a transformation rule, named Rule I, to provide the distribution information required for learning the structure of the RI model. Then, the data collector uses **S**ampling R**A**ndomizer **F**or Multiple **A**ttributes (SAFA), a new LDP method for frequency estimation over multiple attributes, to collect information from the users' data transformed by Rule I to construct the structure of the RI model. Next, each user transforms his/her ranking into multi-attribute data according to another transformation rule, named Rule II, to provide the distribution information for learning the parameters of the 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 the 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 local differential privacy and present a novel approach SAFARI based on the riffle independent 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 further 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 [8].
- We formally prove that SAFARI guarantees ε-local differential privacy. Experimental results over real datasets demonstrate the effectiveness of SAFARI.

The remainder of this paper is organized as follows. Section II reviews the related literature. Section III provides the preliminaries and problem statement. Section IV describes three baseline approaches. Section V gives the details of SAFARI. Section VI shows our experimental results. Finally, Section VII concludes our work.

II. Related Work

The local privacy model was first formalized in [9]. To achieve LDP, there exist several canonical methods [10, 1, 4, 5], which are used as building blocks of LDP solutions. Randomized response (RR) [10], 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 [4] propose succinct histogram (SH), in which each user only needs to send one bit to the data collector. In addition, Wang *et al.* [5] introduce optimized local hashing (OLH) and provide a guideline to choose appropriate LDP methods 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 local differential privacy.

In recent years, finding heavy hitters under local differential privacy 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 analysing multi-attribute data under LDP. Nguyen *et al.* [8] propose Harmony to support mean and frequency estimates and complex machine learning tasks on multi-attribute data containing both numerical and categorical attributes. Cormode *et al.* [19] provide a set of methods, which are based on the

Fourier (Hadamard) transformation, to release marginal statistics of multi-attribute data. Ren *et.al* [20] present a scheme to estimate the joint distribution for multiple variables from multi-attribute data under LDP.

Some other works focus on the collection of locations, time series, telemetry data and graphs under LDP. Chen *et al.* [21] capture users' different privacy requirements and propose an efficient count estimation protocol to learn users' distribution over the spatial domain, which can achieve personalized local differential privacy for each user. Xiong *et al.* [22] describe a novel privacy-preserving compression scheme, called randomized re-quantization mechanism, to compress users' time series under LDP. Ding *et al.* [23] develop LDP mechanisms for repeated collection of telemetry data, which support mean estimation and histogram estimation. Qin *et al.* [24] propose LDPGen, a multi-phase approach, to gradually extract structural information from users and construct a representative graph of the underlying social network.

There also exist some works providing the theoretical understanding of the tradeoff between utility and local differential privacy. Duchi *et al.* [25] provide information theoretic converse techniques for deriving minimax bounds under LDP to compute the minimax rates for mean estimation and convex risk minimization. Kairouz *et al.* [26] develop a family of extremal privatization mechanisms, called staircase mechanisms, to maximize the utility of f-divergence utility functions under local differential privacy.

However, all the above approaches are not suitable for collecting ranking data. To our best knowledge, we are the first to apply local differential privacy to collecting preference rankings.

III. Preliminaries and Problem Statement

A. 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 **B**, **C**, **F**, **P** *and* **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.*

B. Riffle Independent Model

Riffle independent model [6, 7] 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 riffle independent 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 riffle independent model is to recursively decompose 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 a riffle independent model includes the following two phases.

1. Structure learning. In this phase, it first computes the tripletwise mutual information of each possible triplet of distinct items. The definition of tripletwise mutual information is shown as follows.

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:*

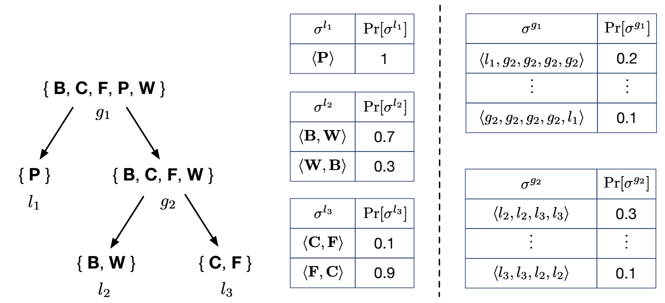
$$\mathcal{I}(x_{k_1}, x_{k_2}, x_{k_3}) = \sum_{\varphi_{k_1}} \sum_{\lambda_{k_2,k_3}} \Pr[\varphi_{k_1}, \lambda_{k_2,k_3}] \log \frac{\Pr[\varphi_{k_1}, \lambda_{k_2,k_3}]}{\Pr[\varphi_{k_1}] \Pr[\lambda_{k_2,k_3}]},$$

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

Then, it hierarchically splits the original item set into riffle independent subsets and constructs the structure of the riffle independent model, named K-thin chain. We refer readers to [7] for further details of the decomposition algorithm.

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, i.e., $|l_j| \leq K$ $(1 \leq j \leq |\mathcal{L}|)$. In other words, the size of the smaller subset at each split is no more than K.

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



{ B, C, F, P, W }
g_1
{ P }
l_1
{ B, C, F, W }
g_2
{ B, W }
l_2
{ C, F }
l_3
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

(a) 2-thin chain. (b) Distributions.

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

distributions, while those over internal item sets are interleaving distributions.

Example 2. *Continuing with Example 1, Fig. 1 shows an example of riffle independent 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 riffle independent 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 $\Pr[\sigma^{l_2}]$) 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 $\Pr[\sigma^{l_3}]$) 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 $\Pr[\sigma^{g_2}]$) 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 riffle independent model, we can obtain the probability of each ranking. For instance, the probability of ranking $\sigma_0 = \langle \boldsymbol{W}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{F}, \boldsymbol{P} \rangle$ is

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

C. Local Differential Privacy

Local differential privacy [9] is defined as follows.

TABLE I: 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 a leaf item set in $\mathcal{T}$

Definition 2 (ε-Local Differential Privacy)**.** *Given a privacy budget ε, a randomized algorithm $\mathcal{R}$ satisfies ε-local differential privacy (or 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*

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

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

For a sequence of locally differentially private algorithms, the sequential composition property [27] 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.*

D. 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 the d items in $\mathcal{X}$. Our goal is to design an approach such that an untrusted data collector is able to collect the rankings from the n users while satisfying local differential privacy.

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 local differential privacy 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 I summarizes notations that will be frequently used in this paper.

IV. Baseline Approaches

As a first attempt, we design three baseline approaches based on three existing LDP methods, namely RAPPOR [1], SH [4] and OLH [5]. These three first-cut solutions help us establish a better understanding of the problem and set up a baseline for our SAFARI approach.

A. RAPPOR

RAPPOR [1] studies the problem of estimating the frequencies of categorical values under LDP. To collect rankings by RAPPOR, we consider each user's ranking as a categorical value in a domain consisting of all possible $d!$ rankings.

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. Finally, each user independently applies RR with a flip probability $p = \frac{e^{\varepsilon/2}}{1+e^{\varepsilon/2}}$ on each bit in his/her length-$d!$ bit vector to generate a perturbed vector and sends it to the data collector. After receiving all users' perturbed vectors, the data collector computes an unbiased estimate of the frequency of each of the $d!$ rankings.

B. SH

SH [4] targets the same problem setting as RAPPOR. To collect rankings by SH, we make the same assumption as that in RAPPOR.

At the beginning of 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.

C. OLH

OLH [5] focuses on the same problem setting as RAPPOR. To collect rankings by OLH, we also make the same assumption as that in RAPPOR.

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 the generalized randomized response [12] 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.

D. Drawbacks of Baselines

Obviously, all these three approaches have serious drawbacks. For RAPPOR, its communication cost is prohibitively high. In particular, each user needs to transmit $d!$ bits to the data collector, which is rather expensive even for moderate values of d. For instance, when $d = 8$, each user needs to transmit $8! \approx 4 \times 10^4$ bits to the data collector. In addition, to achieve ε-LDP for each user u_i, the RAPPOR based approach requires u_i to randomize each bit in his/her length-$d!$ vector independently by RR, which incurs considerable amount of added noise and thus results in low utility of the collected data. For SH, although the communication cost between each user and the data collector is $O(1)$, the scale of the $d! \times m$ matrix Φ is huge, which brings too much added noise and thus leads to low utility. For OLH, while the communication cost between each user and the data collector is also $O(1)$, the use of hash functions incurs a lot of information loss. Hence OLH will lead to low utility as well.

V. Safari

In this section, we present our approach SAFARI for collecting preference rankings under local differential privacy.

A. Overall Flow of SAFARI

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**ndomizer **F**or Multiple **A**ttributes with **R**iffle **I**ndependent Model (SAFARI), for collecting rankings under local differential privacy. Its main idea is to collect a set of distributions over small domains, which are carefully chosen based on the riffle independent model, to approximate the overall distribution of users' rankings. 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 to provide the information about the distributions required for computing the tripletwise mutual information according to a transformation rule, called Rule I. The details of Rule I are given in Section V-B.

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 V-D. Then the data collector computes all the tripletwise mutual information and constructs a K-thin chain $\mathcal{T}$ (K is a small value set by the data collector). Finally, the data collector broadcasts $\mathcal{T}$ to all users.

Phase 3. Each user transforms his/her ranking to provide the information about the relative ranking distributions and interleaving distributions over $\mathcal{T}$ according to another transformation rule, called Rule II, which is discussed in detail in Section V-C.

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 riffle independent model to generate a ranking dataset. To efficiently synthesize a ranking from the riffle independent 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 Rule I and Rule II in Section V-B and V-C, respectively. Then we elaborate the SAFA method used in SAFARI in Section V-D. Finally, we give a comprehensive analysis of SAFARI in Section V-E.

B. Design of Rule I

According to the definition of tripletwise mutual information shown in Section III-B, 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:

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

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 multiple 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 result in redundant information in the transformed data and increase the transformation complexity, as

$$\Pr[\varphi_{k_1}] = \sum_{\lambda_{k_2,k_3}} \Pr[\varphi_{k_1}, \lambda_{k_2,k_3}],$$
$$\Pr[\lambda_{k_2,k_3}] = \sum_{\varphi_{k_1}} \Pr[\varphi_{k_1}, \lambda_{k_2,k_3}].$$

In fact, the data collector only needs to collect the distributions in $\mathcal{S}_3$, from which the distributions in $\mathcal{S}_1$ and $\mathcal{S}_2$ can be derived. Thus, each user only needs to transform his/her ranking to provide the information about the distributions in $\mathcal{S}_3$. Since there are $O(d^3)$ different distributions in $\mathcal{S}_3$, the number of attributes in each user's transformed tuple is $O(d^3)$.

Unfortunately, when d is relatively large, due to the curse of dimensionality, it will also lead to a considerable amount of added noise in the collected data under LDP. To address this issue, we design Rule I, according to which each user transforms his/her ranking to provide the information about the distributions in $\mathcal{S}_1$. The data collector only needs to collect the distributions in $\mathcal{S}_1$, and then makes use of a regression model to estimate the distributions in $\mathcal{S}_2$ and $\mathcal{S}_3$. In particular, we found such estimation is a sparse linear regression problem. Therefore, we choose the Lasso regression model [28], which is effective for solving such problem. 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 the Lasso regression model.

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 $\mathcal{A} = \{A_j | 1 \le j \le d\}$. 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 his/her ranking σ_i.

Example 3. *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 $\mathcal{A} = \{A_j | 1 \le j \le 5\}$. 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 according to 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 Lasso. 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 $\Pr[\lambda_{k_2,k_3}]$ 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 $\Pr[\varphi_{k_2}]$ and $\Pr[\varphi_{k_3}]$;

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 $\Pr[\varphi_{k_2}, \varphi_{k_3}]$.

By fitting the Lasso regression model via Least angle regression [29], the data collector can estimate $\mathbf{p}_{k_2,k_3}$ and obtain the joint distribution $\Pr[\varphi_{k_2}, \varphi_{k_3}]$. From $\Pr[\varphi_{k_2}, \varphi_{k_3}]$, the data collector can easily derive the distribution $\Pr[\lambda_{k_2,k_3}]$.

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 $\Pr[\varphi_{k_1}, \lambda_{k_2,k_3}]$ 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 $\Pr[\varphi_{k_1}]$ and $\Pr[\lambda_{k_2,k_3}]$;
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 $\Pr[\varphi_{k_1}, \lambda_{k_2,k_3}]$.

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 $\Pr[\varphi_{k_1}, \lambda_{k_2,k_3}]$.

Here, we illustrate how to estimate a distribution in $\mathcal{S}_2$ and a distribution in $\mathcal{S}_3$ by Example 4.

Example 4. *Continuing with Example 3, we assume that the data collector has collected the distributions in $\mathcal{S}_1$. To obtain the distribution $\Pr[\lambda_{1,2}]$ 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*

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

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

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 $\Pr[\varphi_1 = \varphi_2]$ (e.g. $\Pr[\varphi_1 = 1, \varphi_2 = 1]$). We have $\mathbf{y}_{1,2} = \mathbf{M}_{1,2} \cdot \mathbf{p}_{1,2}$, as

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

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

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

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

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

By fitting this Lassso regression model, the data collector can obtain the joint distribution $\Pr[\varphi_1, \varphi_2]$ and easily derive the distribution $\Pr[\lambda_{1,2}]$ 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 $\Pr[\varphi_1, \lambda_{2,3}]$ 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

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

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

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

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

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

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

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

Notice that in SAFARI, the data collector and users use the same Rule I. In Rule I, the number of attributes in each user's tuple is $O(d)$. 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 $\mathcal{S}_3$.

To verify the effectiveness of Rule I in the later experiments, here we also display a different version of Rule I, denoted by Rule I*, which is designed based on the simple transformation idea. According to Rule I*, each user transforms his/her ranking to provide the information about the distributions in $\mathcal{S}_3$. The details of Rule I* is given as follows.

Rule I*. 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

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

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, which is derived from his/her ranking 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 $\mathcal{A} = \{A_j | 1 \leq j \leq 30\}$. 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 $\mathbf{B}$ is 2 and the rank of $\mathbf{C}$ is higher than that of $\mathbf{F}$, after user u_0 transforms σ_0 into a tuple t_0 according to* Rule I, *we have $t_0[A_1] = (2, 1)$. The values of other attributes can be derived in a similar way.*

C. 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), $\Pr[\sigma^{l_1}] = \Pr[\langle \mathbf{P} \rangle] = 1$), 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 6. *We continue with the running example to illustrate how to transform a ranking according to 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 according to 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:

$$\begin{aligned} &\langle l_1, g_2, g_2, g_2, g_2 \rangle, \langle g_2, l_1, g_2, g_2, g_2 \rangle, \langle g_2, g_2, l_1, g_2, g_2 \rangle, \\ &\langle g_2, g_2, g_2, l_1, g_2 \rangle, \langle g_1, g_2, g_2, g_2, l_1 \rangle, \end{aligned}$$

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

$$\begin{aligned} &\langle l_2, l_2, l_3, l_3 \rangle, \langle l_2, l_3, l_2, l_3 \rangle, \langle l_2, l_3, l_3, l_2 \rangle, \\ &\langle l_3, l_2, l_2, l_3 \rangle, \langle l_3, l_2, l_3, l_2 \rangle, \langle l_3, l_3, l_2, l_2 \rangle. \end{aligned}$$

In $\sigma_0 = \langle \mathbf{W}, \mathbf{B}, \mathbf{C}, \mathbf{F}, \mathbf{P} \rangle$, since the item $\mathbf{P}$ 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, the number of attributes in each user's tuple is $O(d)$. 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 any attribute 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 $\mathcal{T}$.

D. Sampling Randomizer for Multiple Attributes

To collect the distribution information required for constructing the riffle independent 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 [8], the state-of-the-art method for multi-attribute data analysis 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

Algorithm 1 Sampling Randomizer for Multiple Attributes (SAFA)

```
Input: Users' tuples $\{t_i | 1 \le i \le n\}$
Input: Privacy budget $\varepsilon'$
Output: Frequency vector set $\mathcal{Z} = \{\mathbf{z}_j | 1 \le j \le |\mathcal{A}|\}$
 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: end for
 8: for each $j$-th vector $\mathbf{z}_j$ in $\mathcal{Z}$ do
 9:     DC sets the probability $p_j = \frac{e^{\varepsilon'}}{e^{\varepsilon'} + |dom(A_j)| - 1}$ ;
10:     DC sets the probability $q_j = \frac{1}{e^{\varepsilon'} + |dom(A_j)| - 1}$ ;
11:     for each $k$-th value $\mathbf{z}_j[k]$ in $\mathbf{z}_j$ do
12:         DC updates
            $$\mathbf{z}_j[k] = \frac{1}{n} \cdot \frac{|\mathcal{A}| \cdot \mathbf{z}_j[k] - nq_j}{p_j - q_j} ;$$
13:     end for
14: end for
15: return $\mathcal{Z}$ ;
```

(say A_r) from $\mathcal{A}$ randomly and uses SH [4] 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 [5], which proves that generalized randomized response [26] performs best when estimating frequencies of a small number of values, we propose a new LDP method, called **S**ampling R**A**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 generalized randomized response to the data collector.

Algorithm 1 provides the details of SAFA. It takes users' tuples $\{t_i | 1 \le i \le n\}$ 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 $\{1, \ldots, |\mathcal{A}|\}$ and sends j to u_i. After that, u_i generates a perturbed value index $\widetilde{k}_i^j$ by a local randomizer (LR) whose detail is given in Algorithm 2.

Algorithm 2 Local Randomizer (LR)

```
Input: Attribute index $j$
Input: User $u_i$'s tuple $t_i$
Input: Privacy budget $\varepsilon'$
Output: Perturbed value index $\widetilde{k}_i^j$
 1: Generate a perturbed value index $\widetilde{k}_i^j$ such that
    $$\Pr[\widetilde{k}_i^j = k] = \begin{cases} \frac{e^{\varepsilon'}}{e^{\varepsilon'} + |dom(A_j)| - 1}, & if\, k = I(t_i[A_j]) \\ \frac{1}{e^{\varepsilon'} + |dom(A_j)| - 1}, & if\, k \neq I(t_i[A_j]) \end{cases},$$
    where $k \in \{1, \ldots, |dom(A_j)|\}$.
 2: return $\widetilde{k}_i^j$ ;
```

Local Randomizer. To protect users' privacy at the user side in the collection process, we design a local randomizer (LR) based on generalized randomized response. 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

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

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

$$\begin{aligned}
&\frac{\Pr[SAFA(t_i,\varepsilon') = \widetilde{k}_i^j]}{\Pr[SAFA(t_i{}',\varepsilon') = \widetilde{k}_i^j]}\\
&= \frac{\Pr[\,j\ is\ sampled] \cdot \Pr[LR(j,t_i,\varepsilon') = \widetilde{k}_i^j]}{\Pr[\,j\ is\ sampled] \cdot \Pr[LR(j,t_i{}',\varepsilon') = \widetilde{k}_i^j]}\\
&= \frac{\Pr[LR(j,t_i,\varepsilon') = \widetilde{k}_i^j]}{\Pr[LR(j,t_i{}',\varepsilon') = \widetilde{k}_i^j]}\\
&= \frac{\Pr[\widetilde{k}_i^j | I(t_i[A_j])]}{\Pr[\widetilde{k}_i^j | I(t_i{}'[A_j])]}. \qquad (1)
\end{aligned}$$

We discuss (1) in all four possible cases:

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

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

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

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

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

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

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

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

By substituting (3) into (2), 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 $\mathcal{A}$.

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

$$\begin{aligned}
&Var\left[\mathbf{z}_j[k]\right]\\
&= Var\left[\frac{1}{n}\cdot\frac{|\mathcal{A}|\sum_i^n \mathbb{Y}_j^k(i) - nq_j}{p_j - q_j}\right]\\
&= \frac{|\mathcal{A}|^2}{n^2\cdot(p_j-q_j)^2}\cdot Var\left[\sum_i^n \mathbb{Y}_j^k(i)\right]. \qquad (4)
\end{aligned}$$

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. $n\cdot\left(1-\mathbf{f}_j[k]\right)$) of these random variables are from the Bernoulli distribution with parameter $\frac{p_j}{|A|}$ (resp. $\frac{q_j}{|A|}$). Thus, we have

$$Var\left[\sum_{i}^{n}\mathbb{Y}_j^k(i)\right] = n\cdot \mathbf{f}_j[k]\cdot\left[\frac{p_j}{|\mathcal{A}|}\cdot\left(1-\frac{p_j}{|\mathcal{A}|}\right)\right] + n\cdot(1-\mathbf{f}_j[k])\cdot\left[\frac{q_j}{|\mathcal{A}|}\cdot\left(1-\frac{q_j}{|\mathcal{A}|}\right)\right]. \tag{5}$$

By substituting (5) into (4), we obtain

$$\begin{aligned} &Var\left[\mathbf{z}_j[k]\right] \\ &= \frac{\mathbf{f}_j[k]\cdot[p_j\cdot(|\mathcal{A}|-p_j)]+(1-\mathbf{f}_j[k])\cdot[q_j\cdot(|\mathcal{A}|-q_j)]}{n\cdot(p_j-q_j)^2} \\ &= \frac{q_j\cdot(|\mathcal{A}|-q_j)}{n\cdot(p_j-q_j)^2}+\frac{\mathbf{f}_j[k]\cdot[|\mathcal{A}|\cdot(p_j-q_j)-({p_j}^2-{q_j}^2)]}{n\cdot(p_j-q_j)^2} \\ &\approx \frac{q_j\cdot(|\mathcal{A}|-q_j)}{n\cdot(p_j-q_j)^2} \\ &= \frac{\left(e^{\varepsilon'}+|dom(A_j)|-1\right)\cdot|A|-1}{n\cdot(e^{\varepsilon'}-1)^2}. \end{aligned} \tag{6}$$

This completes the proof. □

E. Analysis of SAFARI

In this section, we give a comprehensive analysis of SAFARI including privacy guarantee, communication cost and time complexity.

Privacy Guarantee. The following theorem establishes the privacy guarantee of SAFARI.

Theorem 5. *SAFARI satisfies ε-local differential privacy.*

Proof. In SAFARI, the data collector needs to access a user's raw data exactly twice. In Phase 2 and Phase 4, the data collector invokes SAFA using privacy budget $\varepsilon/2$, respectively. According to Theorem 2, both these phases satisfy $\varepsilon/2$-LDP. Note that Rule I depends only on the given item set and thus it does not infringe on users' privacy. In addition, all other steps are performed on the noisy outputs. Therefore, by sequential composition (Theorem 1), SAFARI in which SAFA is sequentially invoked twice satisfies ε-local differential privacy. This completes the proof. □

Communication Cost. In SAFARI, the data collector merely communicates with users in Phase 2 and Phase 4. In Phase 2, there are three types of messages. The first type is the random attribute index sent to each user, leading to a total cost of $O(n)$. The second type is the perturbed value index sent by each user, the total cost of this type is $O(n)$. The third type is the K-thin chain sent to each user, which takes a total cost of $O(nd)$. Therefore, the total communication cost in Phase 2 is $O(n+n+nd)=O(nd)$. In Phase 4, there are two types of messages, which are the same as the first two types in Phase 2, leading to a total cost of $O(n)$, respectively. Thus the total communication cost in Phase 4 is $O(n)$.

Time Complexity. We discuss the time complexity of SAFARI on the user side and the data collector, respectively. On the user side, each user needs to transform his/her ranking using $O(d)$ time in Phase 1 and $O(d)$ time in Phase 3. In addition, each user needs to generate a perturbed value index by the local randomizer using $O(1)$ time in Phase 2 and $O(1)$ time in Phase 4. Hence the total time complexity on the user side is $O(d)$.

On the data collector side, the data collector participates in Phases 2, 4 and 5. In Phase 2, the data collector first invokes SAFA to collect the distributions in $\mathcal{S}_1$. In SAFA, for each user, the data collector needs to compute the random attribute index and increase the corresponding value in a vector in $\mathcal{Z}$ by 1. This can be done in $O(n)$ time. Then the data collector needs to take $O(d^2)$ time to get unbiased estimates. Thus the total time complexity of SAFA is $O(n+d^2)$. Then the data collector needs to take $O(d^5)$ time to estimate the distributions in $\mathcal{S}_2$ and $\mathcal{S}_3$ by Lasso. After that, the data collector takes $O(d^3)$ time to compute all the tripletwise mutual information and $O(d^3)$ time to construct a K-thin chain $\mathcal{T}$, respectively. In Phase 4, the data collector invokes SAFA to collect the distributions over $\mathcal{T}$, which has a time complexity of $O(n+d^2)$. In Phase 5, the data collector needs to synthesize n rankings, which can be done in $O(nd)$ time. Thus the total time complexity on the data collector side is $O(nd+d^5)$.

VI. Experiments

In this section, we first evaluate the performance of SAFARI and then evaluate the effectiveness of Rule I and SAFA used in SAFARI, respectively.

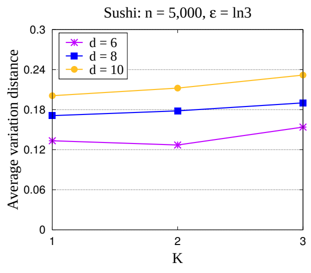
A. Experimental Settings

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

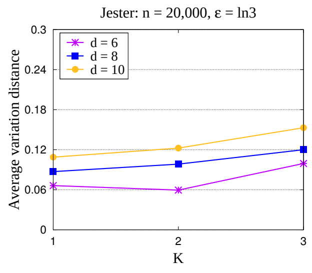
- Sushi [30]: It consists of rankings of 10 kinds of sushi from 5000 voters surveyed across Japan.
- Jester [31]: It includes 1.7 Million continuous ratings (−10.00 to 10.00) of 150 jokes from 50,692 users, which were collected from 2006 to 2012. To convert the dataset into rankings, we order jokes according 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.

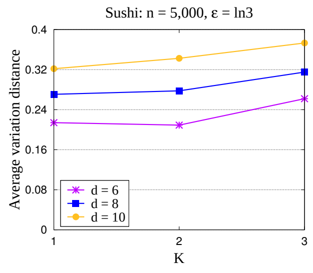
To evaluate the performance of SAFARI, we examine the accuracy of the first-order marginals and the second-order marginals [6] of the synthetic ranking dataset. For



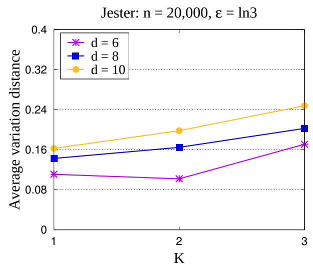
(a) Sushi: Q_1



(b) Jester: Q_1

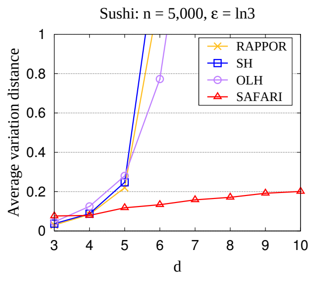


(c) Sushi: Q_2

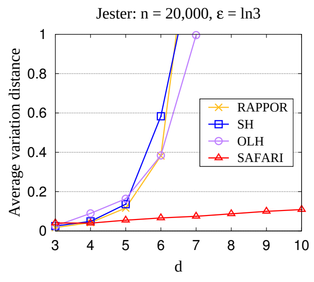


(d) Jester: Q_2

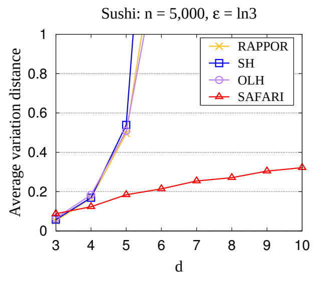
Fig. 2: Impact of K



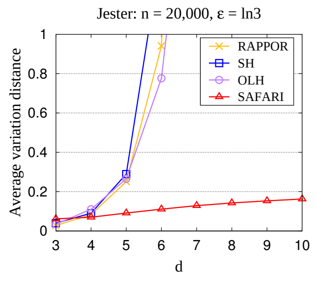
(a) Sushi: Q_1



(b) Jester: Q_1



(c) Sushi: Q_2



(d) Jester: Q_2

Fig. 3: Impact of d

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, we use total variation distance [32], a distance measure for probability distributions. Formally, on a finite probability space Ω, the total variation distance between distributions P and P' is

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

We report the average total variation distance over all marginals in Q_1 and Q_2.

We compare SAFARI against the three baseline approaches, i.e., RAPPOR [1], SH [4] and OLH [5], which are described in Section IV.

We implemented all the approaches in C/C++. The source code of our SAFARI approach is publicly available at [33]. 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.

B. *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 a leaf item set 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 $\ln 3$.

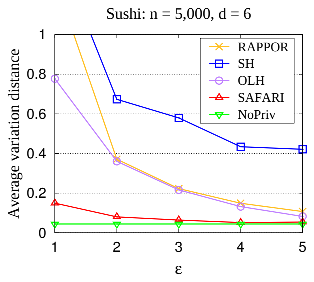
From Fig. 2, we can observe that in general $K = 1$ achieves the best performance. This is because a larger K value normally makes the maximum domain size of attributes in Rule II become overly large, leading to excessive injected noise in the collected distributions over the constructed K-thin chain. 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.

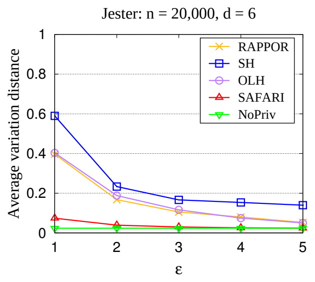
The impact of d. We fix $\varepsilon = \ln 3$ and vary the number d of items to study its impact on the utility of each approach. The results are shown in Fig. 3.

It should be noted that the range of average variation distance 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 average variation distance of a baseline approach larger than 1.

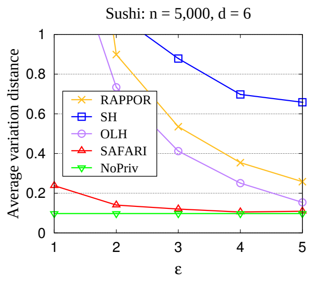
Not surprisingly, for all approaches, their utilities degrade when d increases. Note that RAPPOR, SH and OLH can perform well when d is less than 4. This is because when d is small, the number of all possible rankings is still manageable. However, when d becomes relatively large, the advantage of SAFARI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



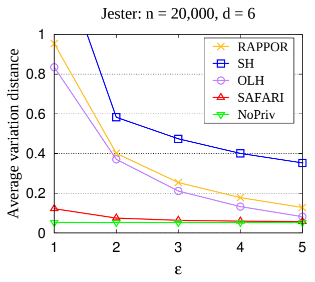
(a) Sushi: Q_1



(b) Jester: Q_1

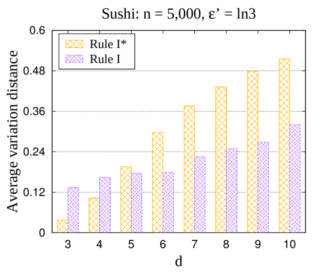


(c) Sushi: Q_2

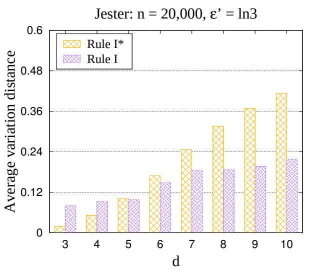


(d) Jester: Q_2

Fig. 4: Impact of ε

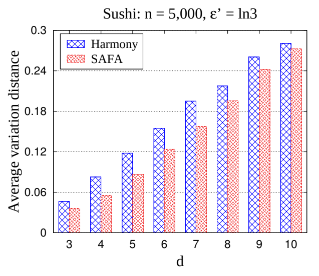


(a) Sushi: varying d

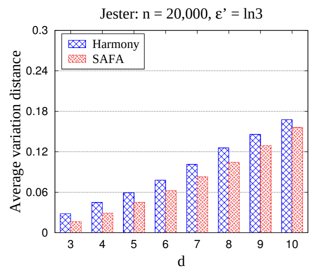


(b) Jester: varying d

Fig. 5: Rule I vs Rule I*



(a) Sushi: varying d



(b) Jester: varying d

Fig. 6: SAFA vs Harmony

which confirms its good scalability.

The impact of ε. We fix $d = 6$ and report the average variation distance of each approach when varying the privacy budget ε. To show the utility loss due to privacy protection, we include a non-private version of SAFARI, denoted by NoPriv, in this set of experiments. Fig. 4 shows the average variation distance 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 is the following. 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 riffle independent model enables SAFARI to effectively capture the correlations among items. Besides, we notice that, SAFARI obtains close performance to NoPiv, 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. Therefore, 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 local differential privacy. It can be expected that when applying SAFARI to real-world applications where the number of users is large, we are able to generate desirable performance.

C. *Effectiveness of Rule I*

To evaluate the effectiveness of Rule I, we compare it against Rule I* displayed in Section V-B. In Rule I*, each user directly transforms his/her ranking to provide the information about the distributions in $\mathcal{S}_3$. We let the data collector use SAFA to collect the distributions from the users' data respectively transformed by Rule I and Rule I*, and report the average variation distance 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. The results are shown in Fig. 5. We can see that, when d is not greater than 4, Rule I* leads to better performance than Rule I. This is because when d is small, the benefit of using Lasso to estimate the distributions in $\mathcal{S}_3$ cannot outweigh the harm of information loss. However, when d is relatively large, Rule I results in significantly better results. This confirms the superiority of Rule I for a relatively large d.

D. *Effectiveness of SAFA*

To evaluate the effectiveness of SAFA, we compare it against Harmony [8]. 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 average variation distance 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. 6. We can observe that SAFA outperforms Harmony in all cases. This is consistent with our analysis in Section V-D that SAFA can improve the accuracy of the collected frequencies of attributes whose domain sizes are relatively small.

VII. Conclusion

In this paper, we present SAFARI, a novel approach for collecting preference rankings under local differential privacy. 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 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-local differential privacy. 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 providing local differential privacy guarantee.

References

[1] Ú. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: randomized aggregatable privacy-preserving ordinal response," in *CCS*, 2014.

[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*, 2006.

[4] R. Bassily and A. D. Smith, "Local, private, efficient protocols for succinct histograms," in *STOC*, 2015.

[5] T. Wang, J. Blocki, N. Li, and S. Jha, "Locally differentially private protocols for frequency estimation," in *USENIX*, 2017.

[6] J. Huang and C. Guestrin, "Riffled independence for ranked data," in *NIPS*, 2009.

[7] ——, "Learning hierarchical riffle independent groupings from rankings," in *ICML*, 2010.

[8] 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.

[9] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. D. Smith, "What can we learn privately?" in *FOCS*, 2008.

[10] 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.

[11] G. C. Fanti, V. Pihur, and Ú. Erlingsson, "Building a RAPPOR with the unknown: Privacy-preserving learning of associations and data dictionaries," *PoPETs*, 2016.

[12] P. Kairouz, K. Bonawitz, and D. Ramage, "Discrete distribution estimation under local privacy," in *ICML*, 2016.

[13] J. Hsu, S. Khanna, and A. Roth, "Distributed private heavy hitters," in *ICALP*, 2012.

[14] R. Bassily, K. Nissim, U. Stemmer, and A. G. Thakurta, "Practical locally private heavy hitters," in *NIPS*, 2017.

[15] M. Bun, J. Nelson, and U. Stemmer, "Heavy hitters and the structure of local privacy," in *PODS*, 2018.

[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*, 2016.

[17] T. Wang, N. Li, and S. Jha, "Locally differentially private frequent itemset mining," in *SP*, 2018.

[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*, 2018.

[19] G. Cormode, T. Kulkarni, and D. Srivastava, "Marginal release under local differnential privacy," in *SIGMOD*, 2018.

[20] 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 Trans. Information Forensics and Security*, 2018.

[21] R. Chen, H. Li, A. K. Qin, S. P. Kasiviswanathan, and H. Jin, "Private spatial data aggregation in the local setting," in *ICDE*, 2016.

[22] S. Xiong, A. D. Sarwate, and N. B. Mandayam, "Randomized requantization with local differential privacy," in *ICASSP*, 2016.

[23] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," in *NIPS*, 2017.

[24] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren, "Generating synthetic decentralized social graphs with local differential privacy," in *CCS*, 2017.

[25] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *FOCS*, 2013.

[26] P. Kairouz, S. Oh, and P. Viswanath, "Extremal mechanisms for local differential privacy," in *NIPS*, 2014.

[27] F. McSherry, "Privacy integrated queries: an extensible platform for privacy-preserving data analysis," in *SIGMOD*, 2009.

[28] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, 1996.

[29] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *The Annals of statistics*, 2004.

[30] http://www.kamishima.net/sushi.

[31] http://www.ieor.berkeley.edu/~goldberg/jester-data.

[32] A. B. Tsybakov, *Introduction to Nonparametric Estimation*, ser. Springer series in statistics. Springer, 2009.

[33] "Collecting preference rankings under local differential privacy (technical report)," 2018. [Online]. Available: https://github.com/cheng-lab-at-bupt/SAFARI